



# ITCS447

## Lecture 1

# Introduction to Internet of Things (IoT)

## IoT Applications and Architectures

Asst. Prof. Dr. Thitinan Tantidham



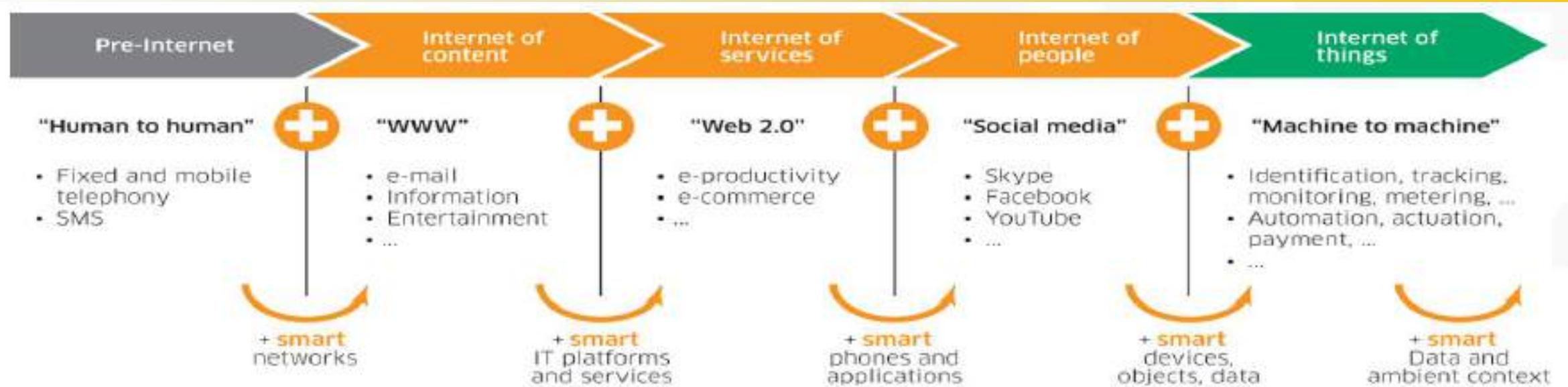


ห้ามมิให้นักศึกษาทำข้า ดัดแปลง หรือใช้ประโยชน์จากการอั่นเมดิชสิทธิ์ปราการอยู่ในระบบการศึกษาอิเล็กทรอนิกส์ (E-Learning) ของมหาวิทยาลัย ไม่ว่าจะทั้งหมดหรือบางส่วน โดยไม่ได้รับอนุญาตจากมหาวิทยาลัย นอกเหนือจากการศึกษาส่วนบุคคล ทั้งนี้ การทำข้า ดัดแปลง หรือเผยแพร่ต่อสาธารณะขึ้นซึ่งงานอั่นเมดิชสิทธิ์ จะมีโทษปรับตั้งแต่ 20,000 บาท ถึง 200,000 บาท และหากเป็นการกระทำเพื่อการค้า จะมีโทษจำคุกตั้งแต่ 6 เดือน ถึง 10 ปี หรือปรับตั้งแต่ 100,000 บาท ถึง 800,000 บาท หรือทั้งจำทั้งปรับ

Unless allowed by Mahidol University, the User shall not copy, modify, or exploit in part or in whole of the copyrighted materials on the Platform, other than for your own individual study. The copy, modify or communication to public of the copyrighted materials shall be inflicted with a fine from 20,000 Baht up to 200,000 Baht. If the offence is committed with the commercial purpose, the offender shall be inflicted with imprisonment for a term from 6 months up to 10 years or a fine from 100,000 Baht up to 800,000 Baht or both imprisonment and fine.

- Internet of Things (IoT) Evolution
- History of the IoT
- IoT Definitions
- IoT Applications
- IoT Architectures
- IoT Technologies

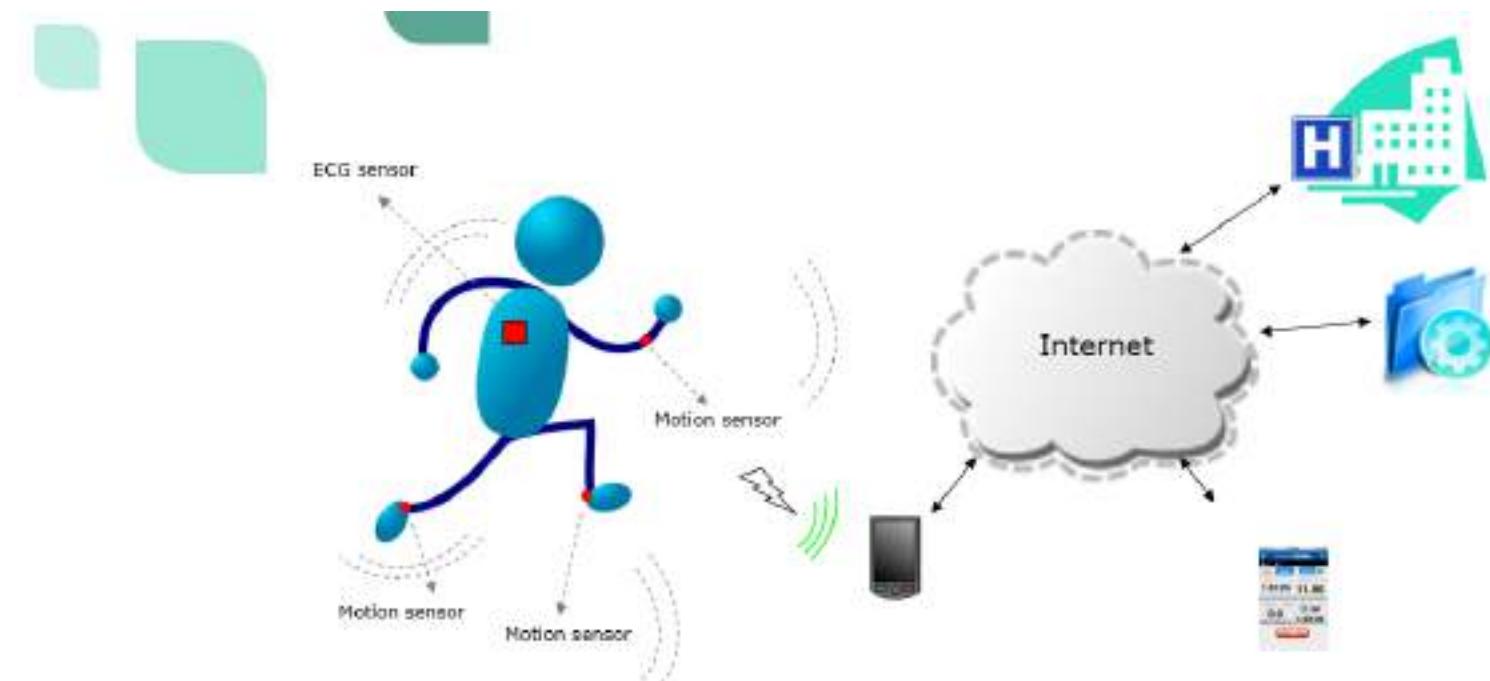
# Internet of Things (IoT) Evolution



- Move from Internet of People to Internet of Things
  - Internet appears everywhere in the world
  - It is primarily connection between people
- IoT has roots in several earlier technologies: pervasive information systems, sensor networks, and embedded computing.

Slide from Dr. Kayarvizhy N.

## People Connecting with Things

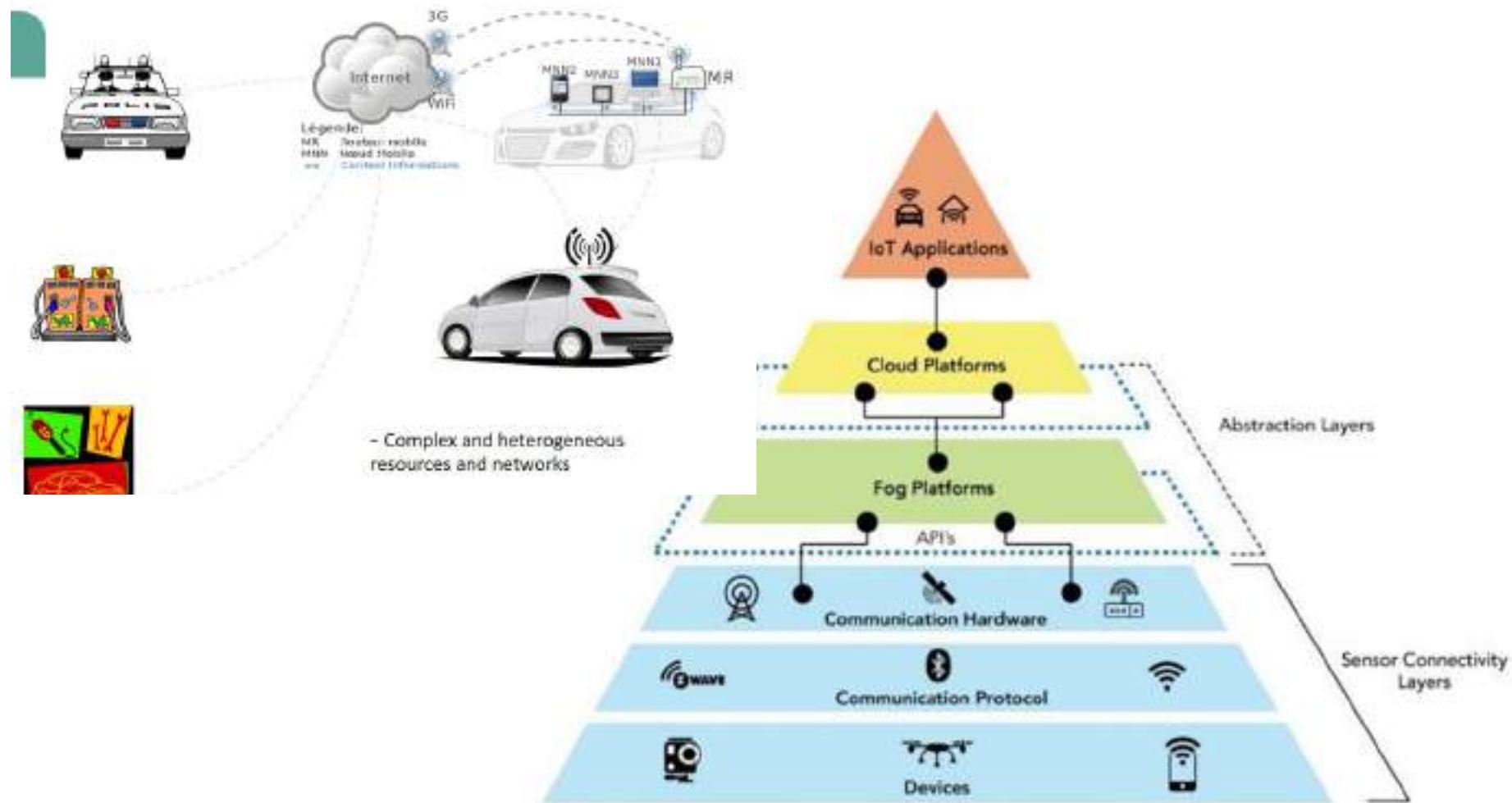


Slide from Dr. Kayarvizhy N.

# Internet of Things (IoT) Evolution



## Things Connecting with Things



Slide from Dr. Kayarvizhy N.

cloudtp.com

# History of the IoT

## Kevin Ashton [1]

- The term "IoT" can most likely be attributed to Kevin Ashton in 1997
- His work at Procter and Gamble using RFID tags to manage supply chains.
- The work brought him to MIT in 1999 where he and a group of like-minded individuals started the Auto-ID Center research consortium.
- The term "Internet of Things" is coined by Kevin Ashton 1999.

"If we had computers that knew **everything** there was to know about **things** - using data they gathered without any help from us - we would be able to track and count everything, and greatly reduce waste, loss and cost.  
**We would know when things needed replacing, repairing or recalling, and whether they were fresh or past their best".** - Kevin Ashton

[https://en.wikipedia.org/wiki/Kevin\\_Ashton](https://en.wikipedia.org/wiki/Kevin_Ashton)

<https://www.c-span.org/video/?325061-1/communicators-kevin-ashton>

<https://www.smithsonianmag.com/innovation/kevin-ashton-describes-the-internet-of-things-180953749/>

<https://newsroom.cisco.com/feature-content?type=webcontent&articleId=1558161>

## Kevin Ashton Career Timeline

**EDUCATION, 1980s**

"I spent some time in Norway and fell in love with Harald Bratt. I found out that I could study Bratt at the University of London. The Scandinavian Studies department at the University of London doesn't get many in-bound immigrants. I don't think."

**WORK EXPERIENCE**

**1986**  
Involved principally in starting noodle bar Wagamama. Offered a brand job at P&G in the U.K.

**1987**  
P&G Oil of Olay lipstick brand manager. Promoted RFID in supply chain management.

**1990**  
Co-Authored "Internet of Things" as part of an RFID power-point presentation. Loaned by P&G to MIT to start the Auto-ID center, turned a broom closet office into a global organization.

**2004**  
Joined start-up ThingMagic as VP of Marketing.

**2007**  
Joined cleantech company EnerNex as VP of Marketing.

**2009**  
Started Zemni, President and CEO. Zemni acquired by Belkin International.

**2010**  
General Manager for Belkin.

**2013**  
Full-time Writer.

**2015**  
Release of first book, *How to Fly a Horse: The Secret History of Creation, Invention, and Discovery in America*.

**LIFE LESSONS**

"The best time to do a startup actually is when everyone thinks it's the worst time to do a startup, when everyone's down and gloomy."

"It's improbable that there's only one store that has a problem and the brand manager happens to go into that. It's more likely that you're noticing a pattern. It's the same thing, by the way, when your kids take home a product and it catches on fire in the bathroom. That's not that the only one that does that; you've got a major problem."

**Listen to the full Kevin Ashton interview**



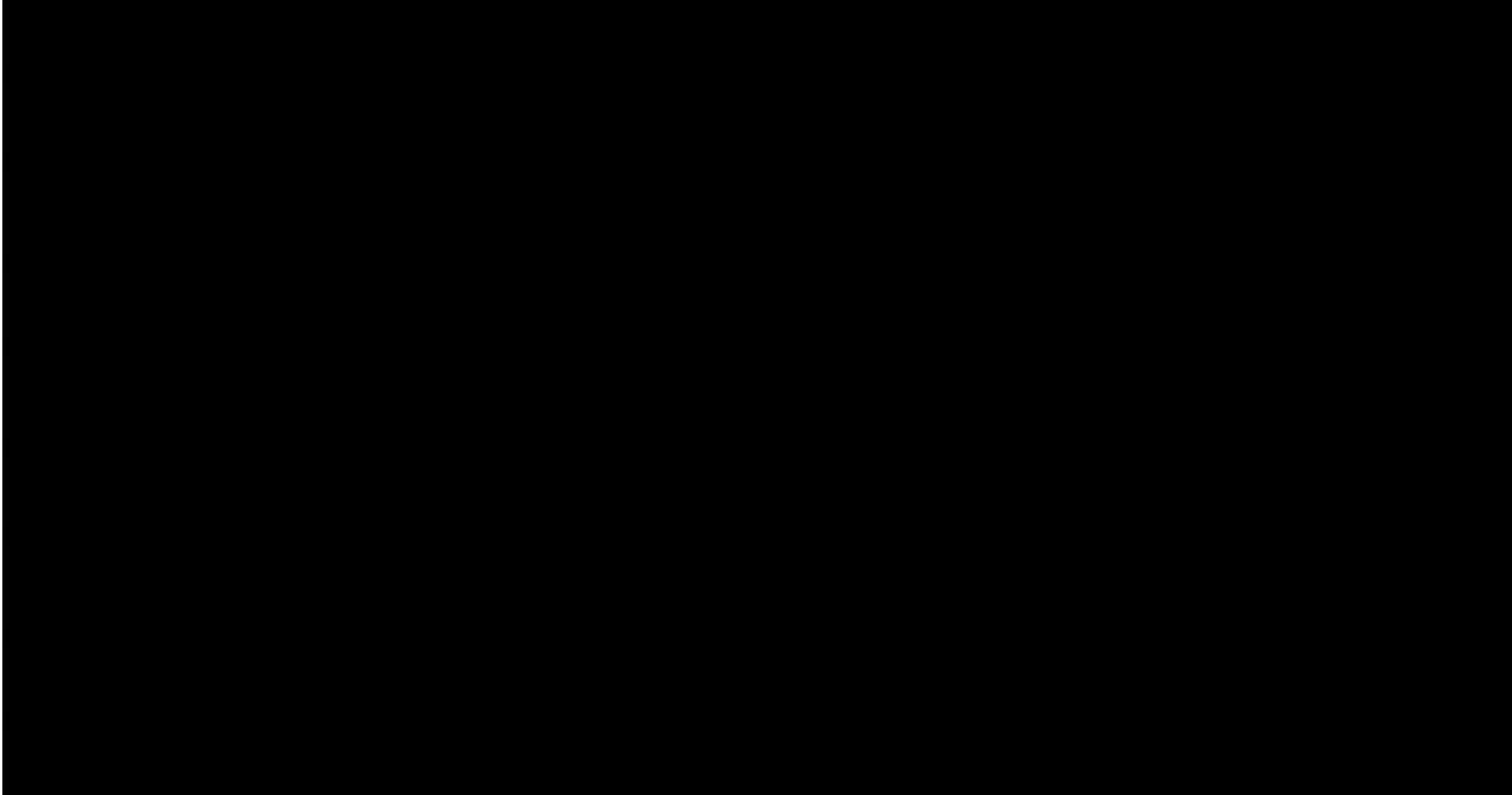


# History of the IoT

## Kevin Ashton [2]

What is the Internet of Things?

Video from <https://www.youtube.com/watch?v=uUrb4sOEIH4>



# History of the IoT



# Kevin Ashton [3]

## Examples of Internet of Things in 90 seconds:

Video from <https://www.youtube.com/watch?v=ML9DmK0IpjI&feature=youtu.be>

```
if (m_head == NULL) {
    m_head = new Node();
    m_head->next = m_head;
}
else if (m_head->size == MAX_ELEMENT_SIZE) {
    Node* newHead = new Node();
    newHead->size = 0;
    newHead->next = m_head;
    m_head = newHead;
}

Node* findIntersectionPoint(const Line1 & l1, const Line2 & l2) {
    if (l1.size == 0 || l2.size == 0)
        return NULL;

    Node* head = m_head;
    Node* tail = m_head->next;
    Node* current = m_head->next->next;
    Node* previous = m_head->next;
    Node* nextNode = m_head->next->next;
    Node* result = NULL;

    while (current != m_head) {
        if (isIntersection(l1, l2, current)) {
            result = current;
            break;
        }
        previous = tail;
        tail = current;
        current = nextNode;
        nextNode = current->next;
    }

    return result;
}

bool isIntersection(const Line1 & l1, const Line2 & l2, Node * node) {
    Point p1 = l1.line1->P1;
    Point p2 = l1.line1->P2;
    Point p3 = l1.line2->P1;
    Point p4 = l1.line2->P2;
    Point p5 = l2.line1->P1;
    Point p6 = l2.line1->P2;
    Point p7 = l2.line2->P1;
    Point p8 = l2.line2->P2;

    double d1 = signed_area(p1, p2, p3);
    double d2 = signed_area(p1, p2, p4);
    double d3 = signed_area(p5, p6, p7);
    double d4 = signed_area(p5, p6, p8);

    if ((d1 < 0 & d2 > 0) || (d1 > 0 & d2 < 0) || (d3 < 0 & d4 > 0) || (d3 > 0 & d4 < 0))
        return true;
    else
        return false;
}

double signed_area(const Point & p1, const Point & p2, const Point & p3) {
    return (p2.x - p1.x)*(p3.y - p1.y) - (p3.x - p1.x)*(p2.y - p1.y);
}

Point intersectionPoint(const Line1 & l1, const Line2 & l2) {
    Node* result = findIntersectionPoint(l1, l2);
    if (result == NULL)
        return Point(0, 0);
    else
        return result->point;
}
```

# History of the IoT [Lea20]



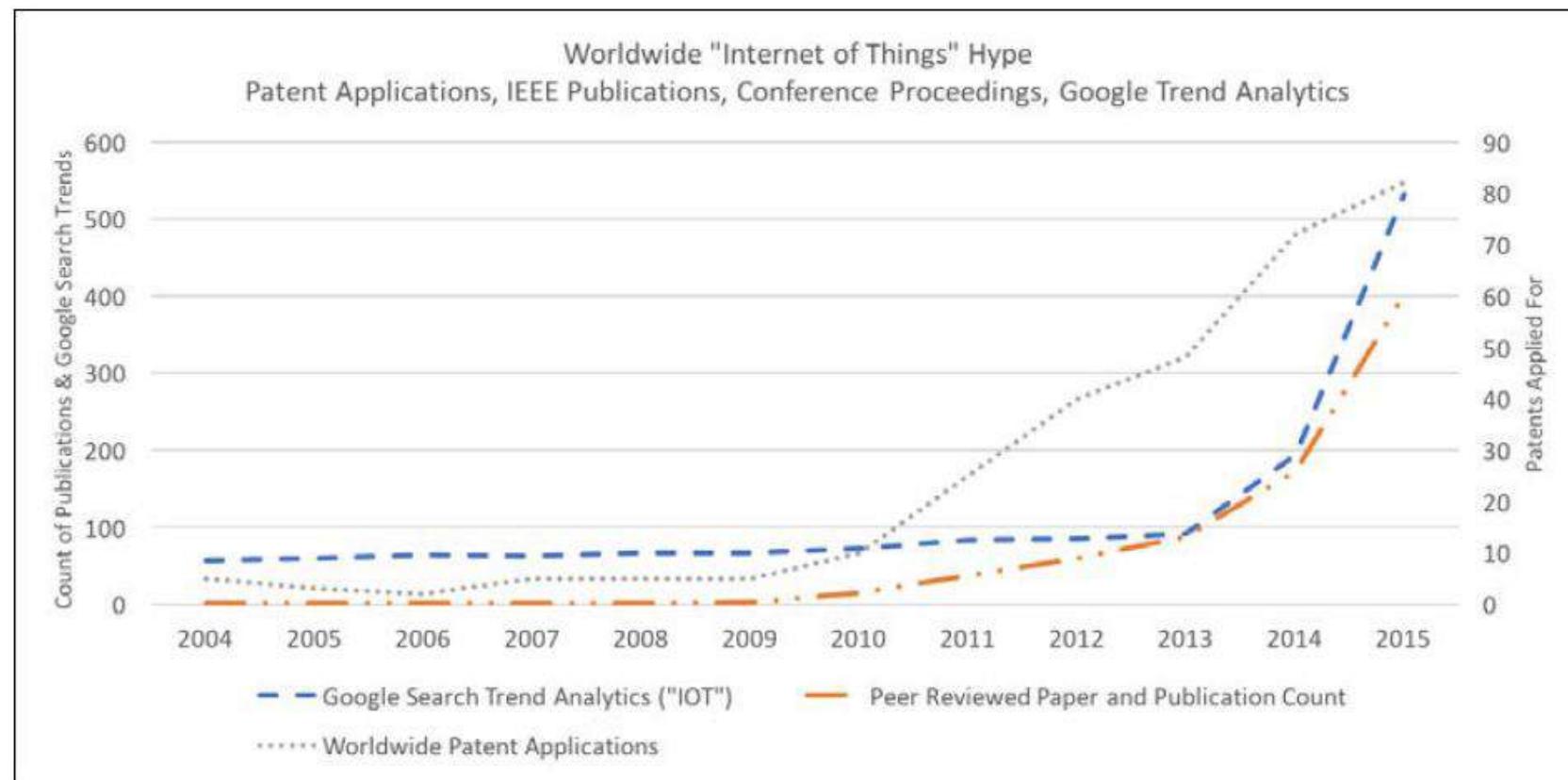
The following timeline shows the slow progress in connecting things to the Internet.

Year	Device	Reference
1969	The Internet starting with ARPANET	
1973	Mario W. Cardullo receives the patent for first RFID tag.	US Patent US 3713148 A
1982	Carnegie Mellon Internet-connected soda machine.	<a href="https://www.cs.cmu.edu/~coke/history_long.txt">https://www.cs.cmu.edu/~coke/history_long.txt</a>
1989	Internet-connected toaster at Interop'89.	IEEE Consumer Electronics Magazine (Volume: 6, Issue: 1, Jan. 2017)
1991	HP introduces HP LaserJet IISi: the first Ethernet-connected network printer.	<a href="http://hpmuseum.net/display_item.php?hw=350">http://hpmuseum.net/display_item.php?hw=350</a>
1993	Internet-connected coffee pot at University of Cambridge (the first Internet-connected camera).	<a href="https://www.cl.cam.ac.uk/coffee/qsf/coffee.html">https://www.cl.cam.ac.uk/coffee/qsf/coffee.html</a>
1996	General Motors OnStar (2001 remote diagnostics).	<a href="https://en.wikipedia.org/wiki/OnStar">https://en.wikipedia.org/wiki/OnStar</a>
1998	Bluetooth Special Interest Group (SIG) formed.	<a href="https://www.bluetooth.com/about-us/our-history">https://www.bluetooth.com/about-us/our-history</a>
1999	LG Internet Digital DIOS refrigerator.	<a href="https://www.telecompaper.com/news/lg-unveils-internetreadyrefrigerator--221266">https://www.telecompaper.com/news/lg-unveils-internetreadyrefrigerator--221266</a>
2000	First instances of the Cooltown concept of pervasive computing everywhere: HP Labs, a system of computing and communication technologies that, combined, create a web-connected experience for people, places, and objects.	<a href="https://www.youtube.com/watch?v=U2AkkuIVV-I">https://www.youtube.com/watch?v=U2AkkuIVV-I</a>
2001	First Bluetooth product launched: KDDI Bluetooth-enabled mobile phone.	<a href="http://edition.cnn.com/2001/BUSINESS/asia/04/17/tokyo.kddibluetooth/index.html">http://edition.cnn.com/2001/BUSINESS/asia/04/17/tokyo.kddibluetooth/index.html</a>
2005	United Nation's International Telecommunications Union report predicting the rise of IoT for the first time.	<a href="http://www.itu.int/osg/spu/publications/internetofthings/internetofThings_summary.pdf">http://www.itu.int/osg/spu/publications/internetofthings/internetofThings_summary.pdf</a>
2008	IPSO Alliance formed to promote IP on objects, first IoT-focused alliance.	<a href="https://www.ipso-alliance.org">https://www.ipso-alliance.org</a>
2010	The concept of Smart Lighting formed after success in developing solid-state LED light bulbs.	<a href="https://www.bu.edu/smartlighting/files/2010/01/BobK.pdf">https://www.bu.edu/smartlighting/files/2010/01/BobK.pdf</a>
2014	Apple creates iBeacon protocol for beacons.	<a href="https://support.apple.com/enus/HT202880">https://support.apple.com/enus/HT202880</a>

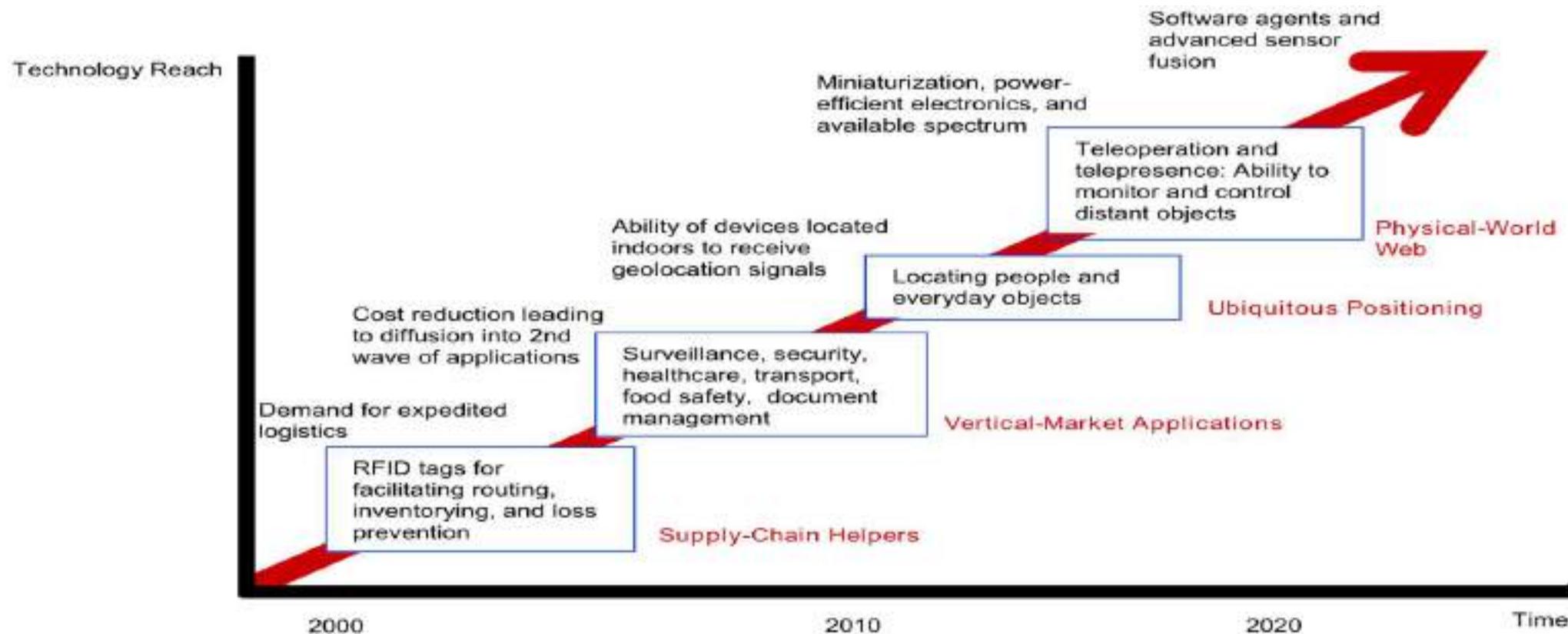
# History of the IoT [Lea20]

IoT has generated a lot of interest and hype.

- The number of patents issued (<https://www.uspto.gov>) has grown exponentially since 2010.
- The number of Google searches (<https://trends.google.com/trends/>)
- IEEE peer-reviewed paper publications hit the knee of the curve in 2013.



## IoT Road Map



Source: SRI Consulting Business Intelligence



## What is Internet of Things (IoT)?

- The Internet of Things, also called **The Internet of Objects**, is **the inter-networking of physical devices**, vehicles (also referred to as "connected devices" and "smart devices"), buildings, and other items **embedded with electronics, software, sensors, actuators, and network connectivity** which **enable these objects to collect and exchange data** with manufacturer, operator, other devices through network infrastructure—[https://en.wikipedia.org/wiki/Internet\\_of\\_things](https://en.wikipedia.org/wiki/Internet_of_things)
- **By embedding short-range mobile transceivers** into a wide array of additional gadgets and everyday items, enabling new forms of **communication between people and things**, and **between things themselves**. – WSIS 2005
- The term “Internet of Things” has come to describe a number of technologies and research disciplines that **enable the Internet to reach out into the real world of physical objects**. – IoT 2008
- Things having identities and virtual personalities operating in **smart spaces using intelligent interfaces** to connect and communicate within social, environmental, and user contexts. – IoT 2020

## IoT Functionalities



Dynamic control of Industry  
and daily life



Improve the resource  
Utilization ratio



Integrate human society  
+ Physical systems



Accessibility and  
Usability



Flexible configuration  
P&P



Better relationship between  
Human and nature



Technology Integrator



Universal transport  
& internetworking

IoT is network of physical objects

- Embedded systems with electronics, software, sensors
- Enable objects to exchange data
- Allow remote control
- Direct integration computer with physical world
- Result: Automation in all fields

Slide from Dr. Kayarvizhy N.

# IoT Applications [1]



- Consumer**
  - Smart home control (lighting, security, comfort)
  - Optimized energy use
  - Maintenance
- Retail**
  - Product tracking
  - Inventory control
  - Focused marketing
- Medical**
  - Wearable devices
  - Implanted devices
  - Telehealth services
- Military**
  - Resource allocation
  - Threat analysis
  - Troop monitoring

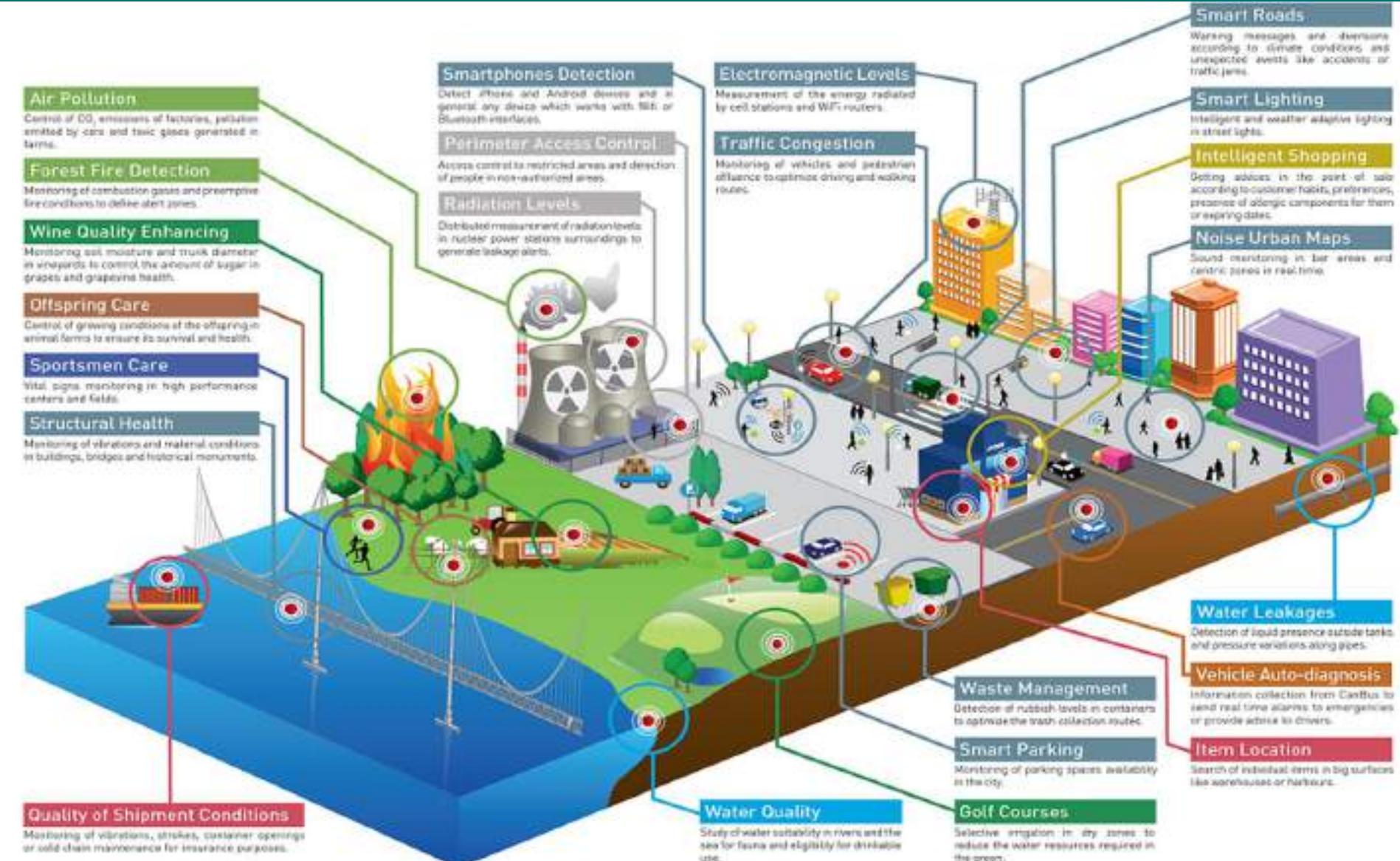


- Industrial**
  - Smart Meters
  - Wear-out sensing
  - Manufacturing control
  - Climate control
- Automotive**
  - Parking
  - Traffic flow
  - Anti-theft location
- Environmental**
  - Species tracking
  - Weather prediction
  - Resource management
- Agriculture**
  - Crop management
  - Soil analysis

CONFIDENTIAL Not for distribution. All Contents © 2014 Alfa Systems

1

# IoT Applications [2]



Slide from Dr. Kayarvizhy N.

# IoT Applications [3]

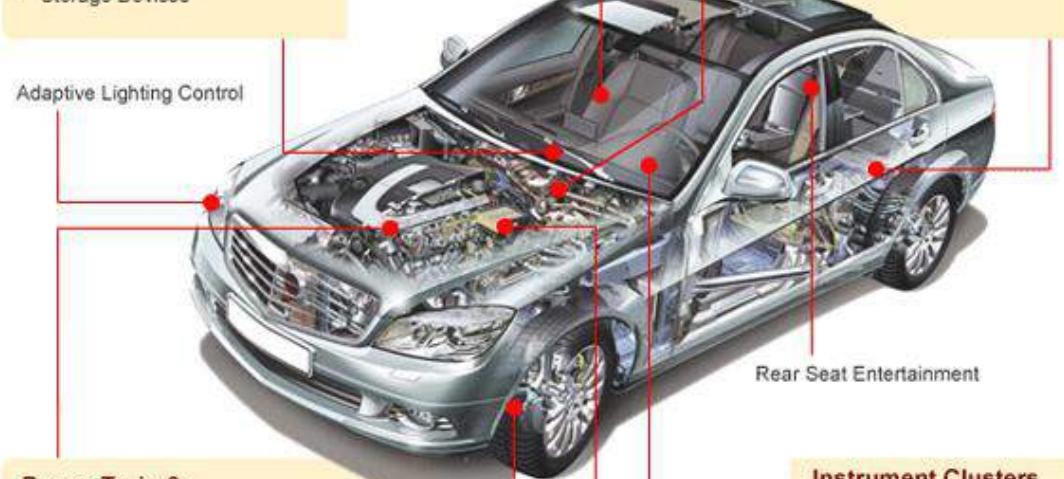


## Infotainment & Telematics

- Hands Free Telephony
- Navigation
- DAB & IBOC Receivers
- E-Calls, B-Calls, S-Calls
- Automotive Hybrid TV Receiver
- DVD Playback, Radios
- Storage Devices

## Vehicle Bus Communication

- Multimedia Bus Interface like most & 1394
- CAN, J1850, LIN, GMLAN, UART, FNOS



## Power Train & Engine Management

- Diagnostic Tests and Interfaces
- Complex and Simple IO Development
- Model Based Software Development
- Model-In-Loop
- Software-In-Loop
- Hardware-in-Loop

## HEV/EV

- Vehicle Energy Management System
- The Pressure Monitoring Systems
- Modelling of Suspension Systems

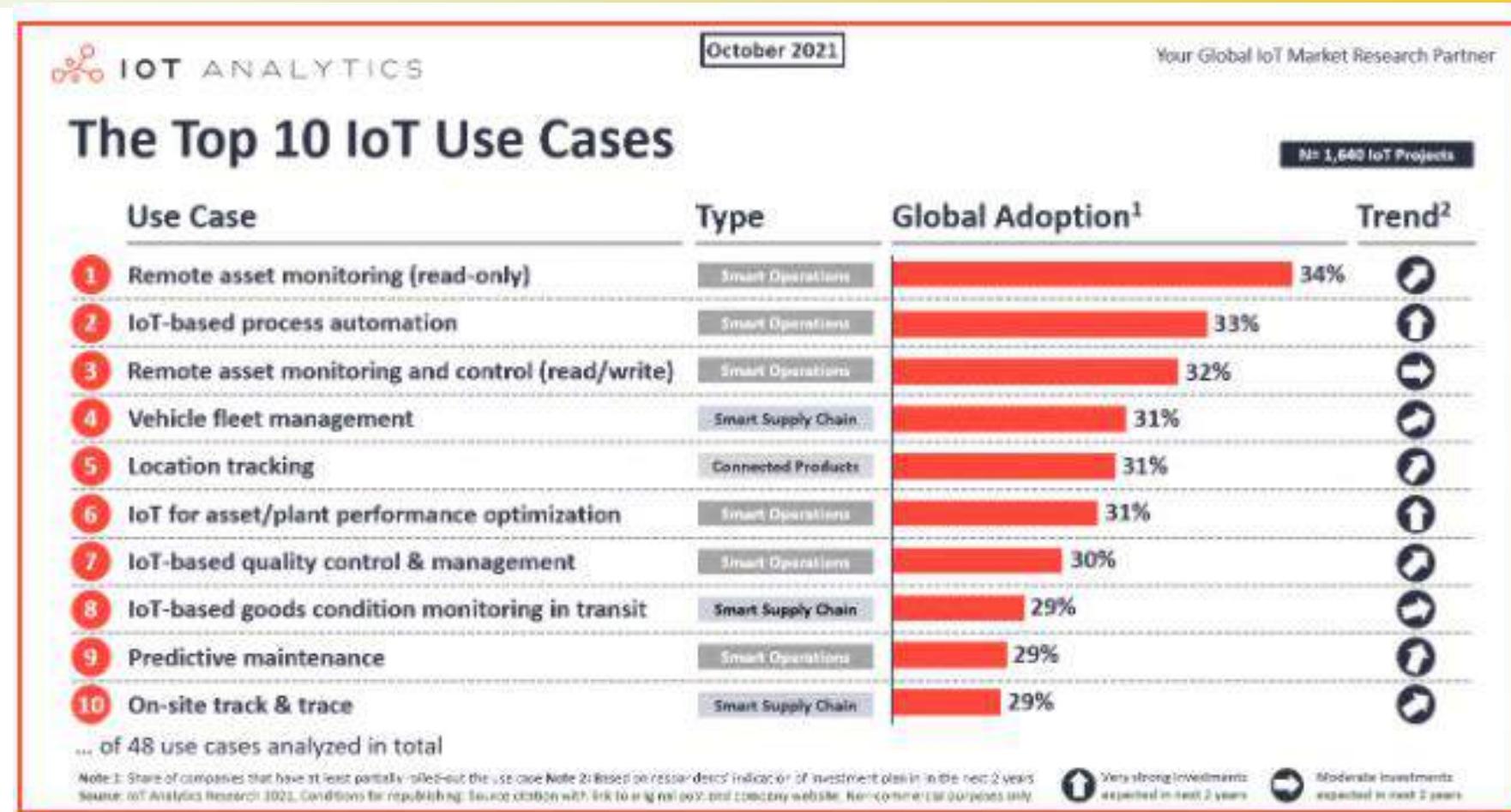
## Instrument Clusters

- Hardware Architecture
- HMI Components
- Reusable Components
- Integration and Validation
- Dot-matrix, TFT display
- Telltale



# IoT Applications [4]

## Examples of IoT Use Cases



Ref: <https://iot-analytics.com/top-10-iot-use-cases/>

Learn More:

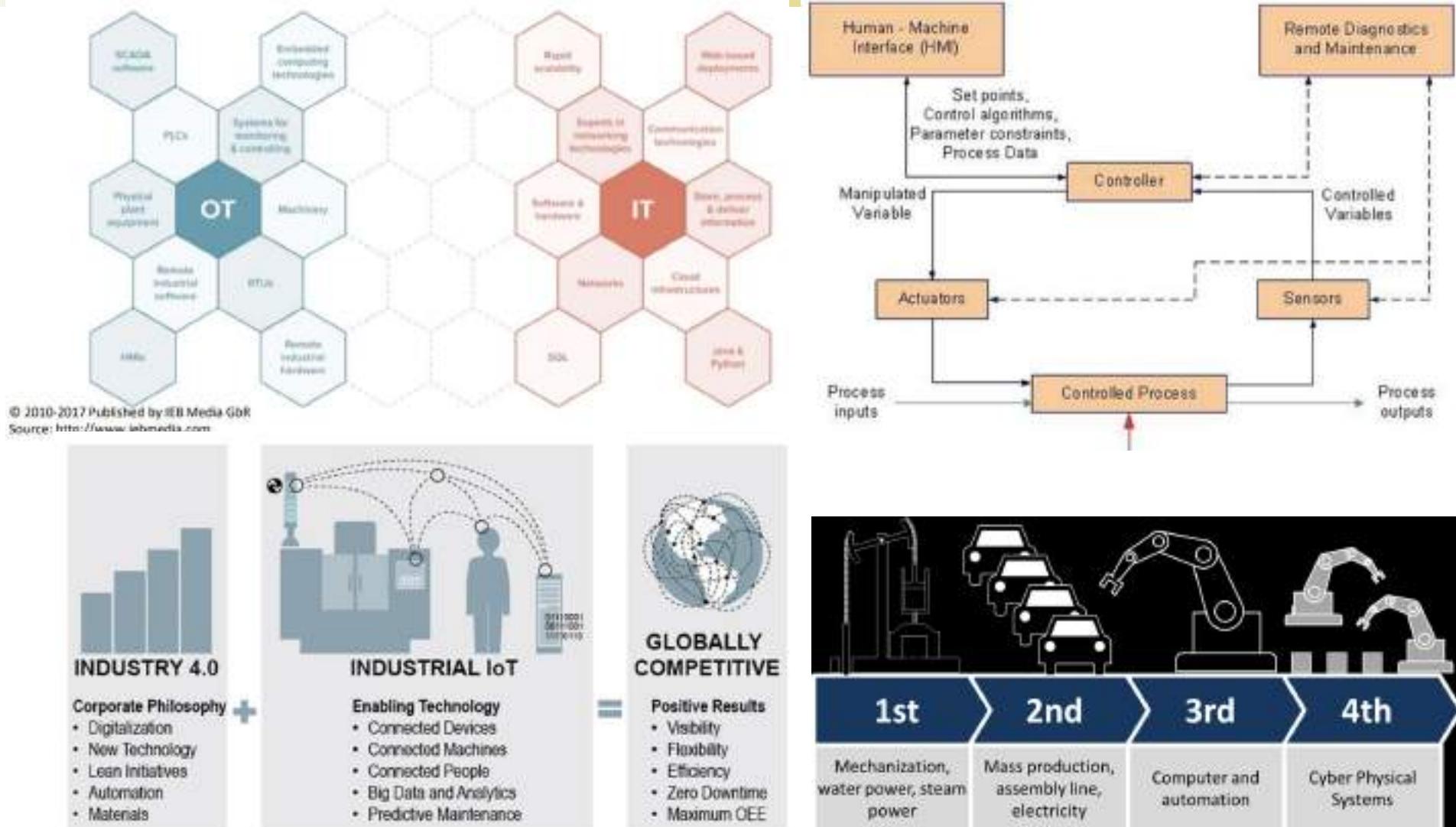
<https://pressbooks.bccampus.ca/iotbook/>

## What is Industrial IoT (IIoT)?

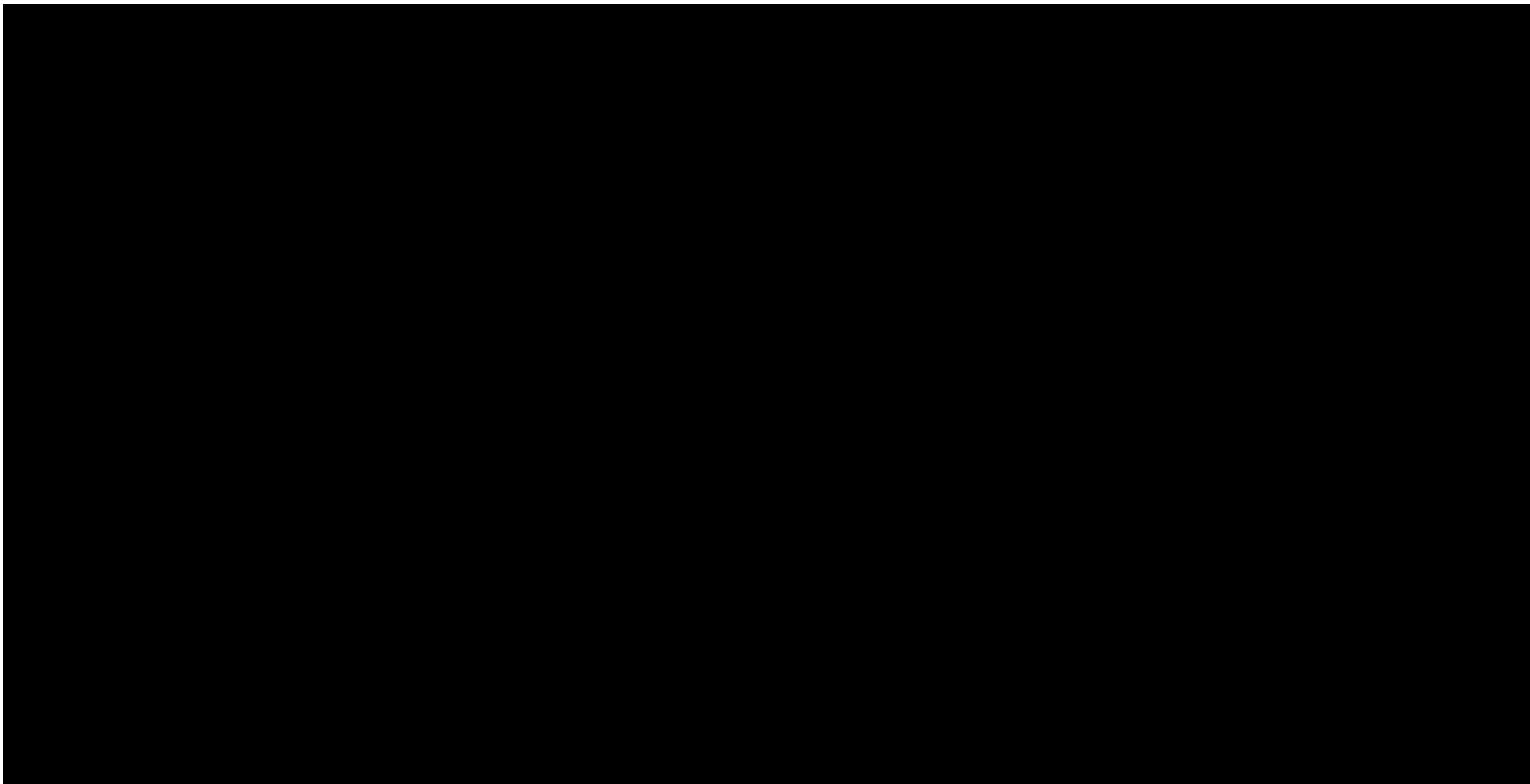
- One of the fastest-growing and largest segments in the overall IoT space by the number of connected things and the value those services bring to **manufacturing and factory automation**.
- This segment has traditionally been the world of **operations technology (OT)** which involves hardware and software tools **to monitor physical devices in real time**.
- For examples: **Supervisory Control and Data Acquisition (SCADA)** systems and **Industrial Control Systems (ICS)**
  - Typically the systems involve Programmable Logic Controllers (PLCs) that monitor or control various **sensors and actuators** on machinery.
  - SCADA systems are distributed and only recently have been connected to Internet services.
  - This is where **Industry 2.0** and the new growth of manufacturing is taking place.
  - They use **communication protocols** such as **ModBus, BACNET, and Profibus**.
- Case Studies: <https://www.facebook.com/SCADA-Thailand-151583115684091/>

# IoT Applications: IIoT

## What is Industrial IoT (IIoT)?



# IoT Applications: IIoT - SCADA

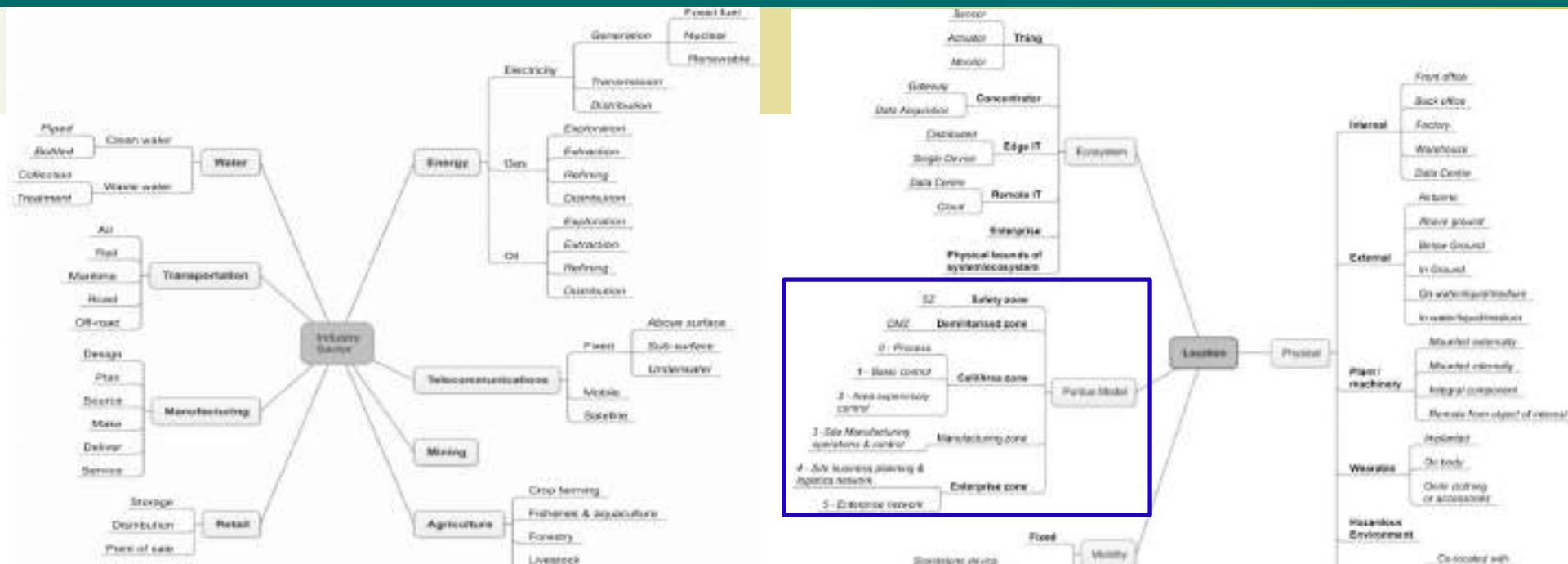


Video from <https://www.youtube.com/watch?v=n1FM1q9QPJw>

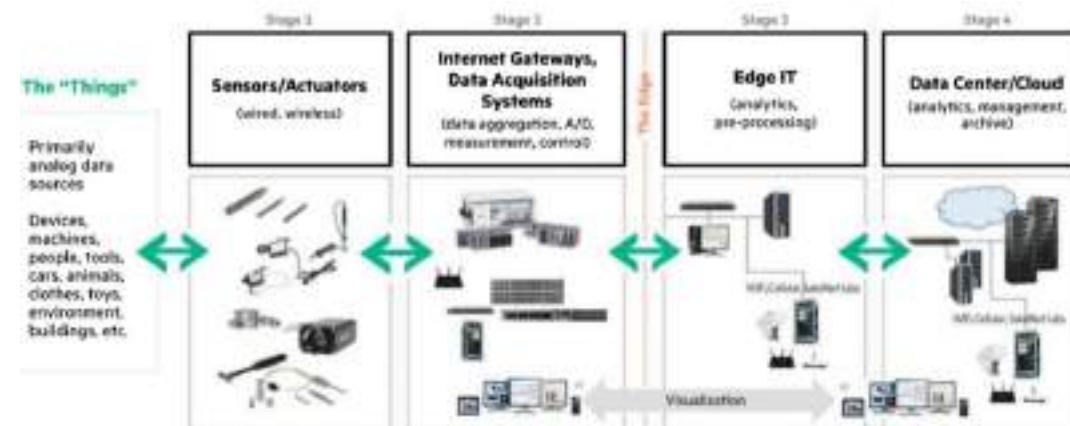
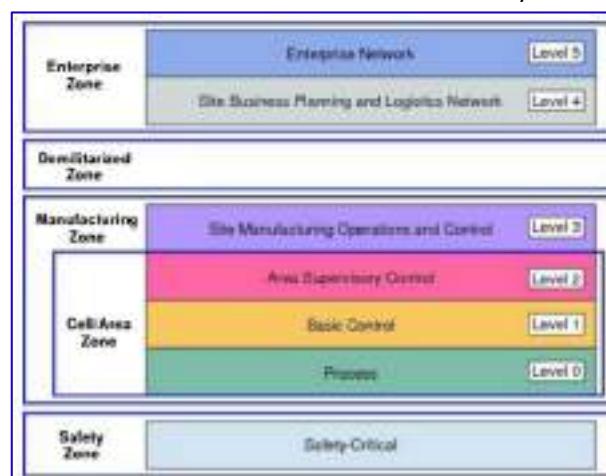
# IoT Applications: IIoT [BHC+18]



## Industrial



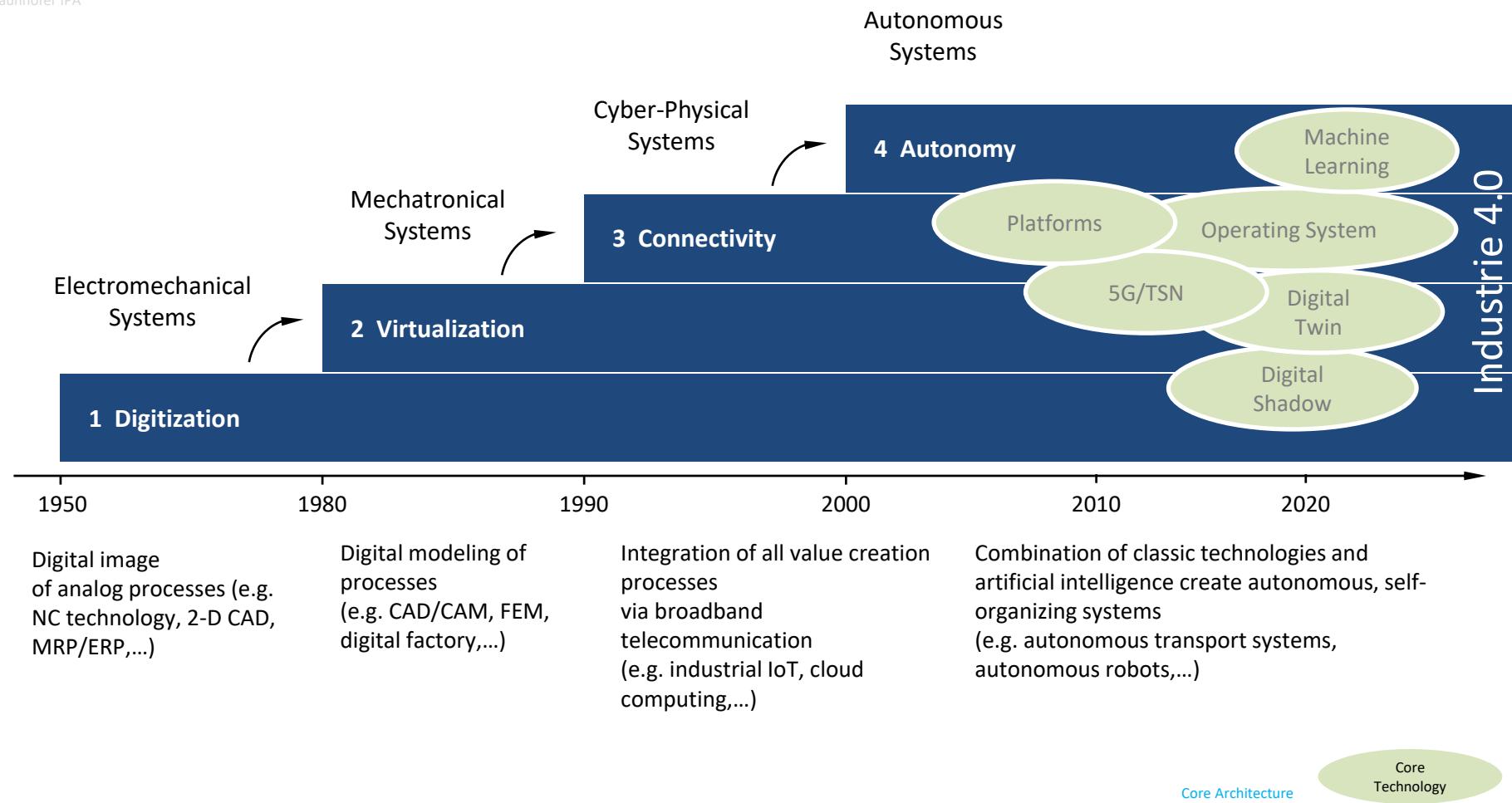
Purdue Reference Model for Control Hierarchy





## Digital Transformation Development Stages and Industrie 4.0 Levels

source: Fraunhofer IPA





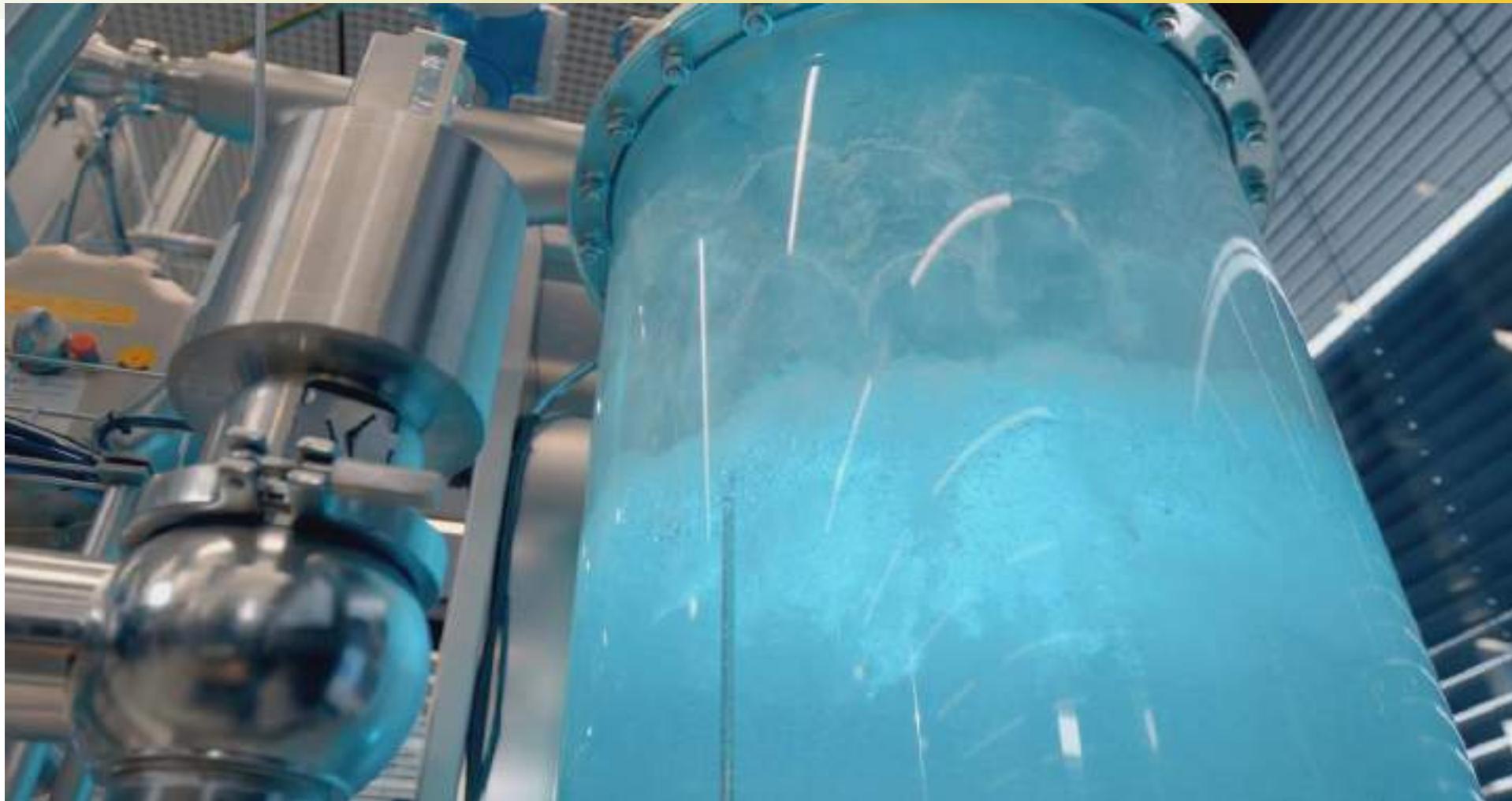
## Digital Transformation Development Stages and Industrie 4.0



Video from: <https://www.youtube.com/watch?v=TePOTewlFwk>

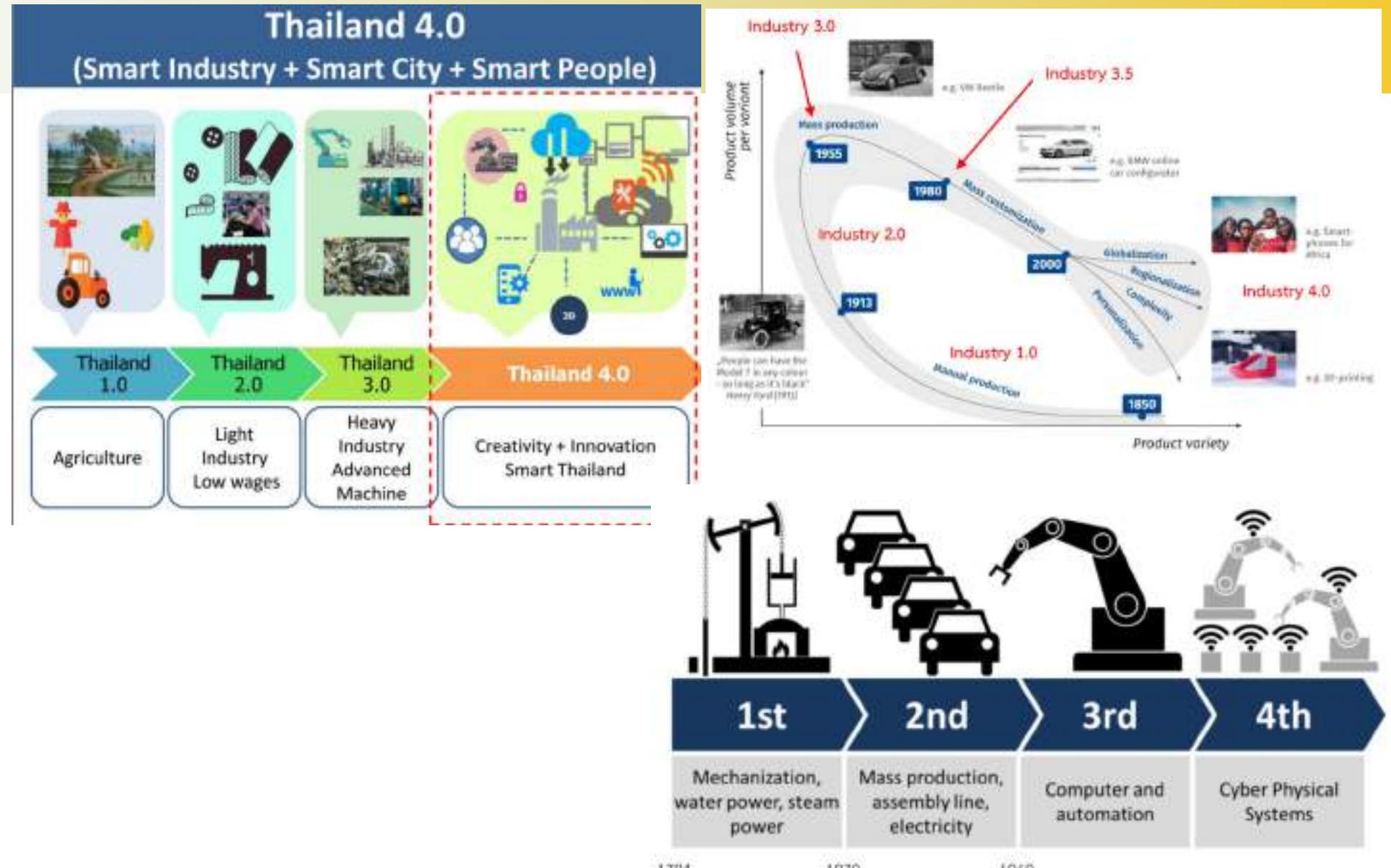


## Industrie 4.0

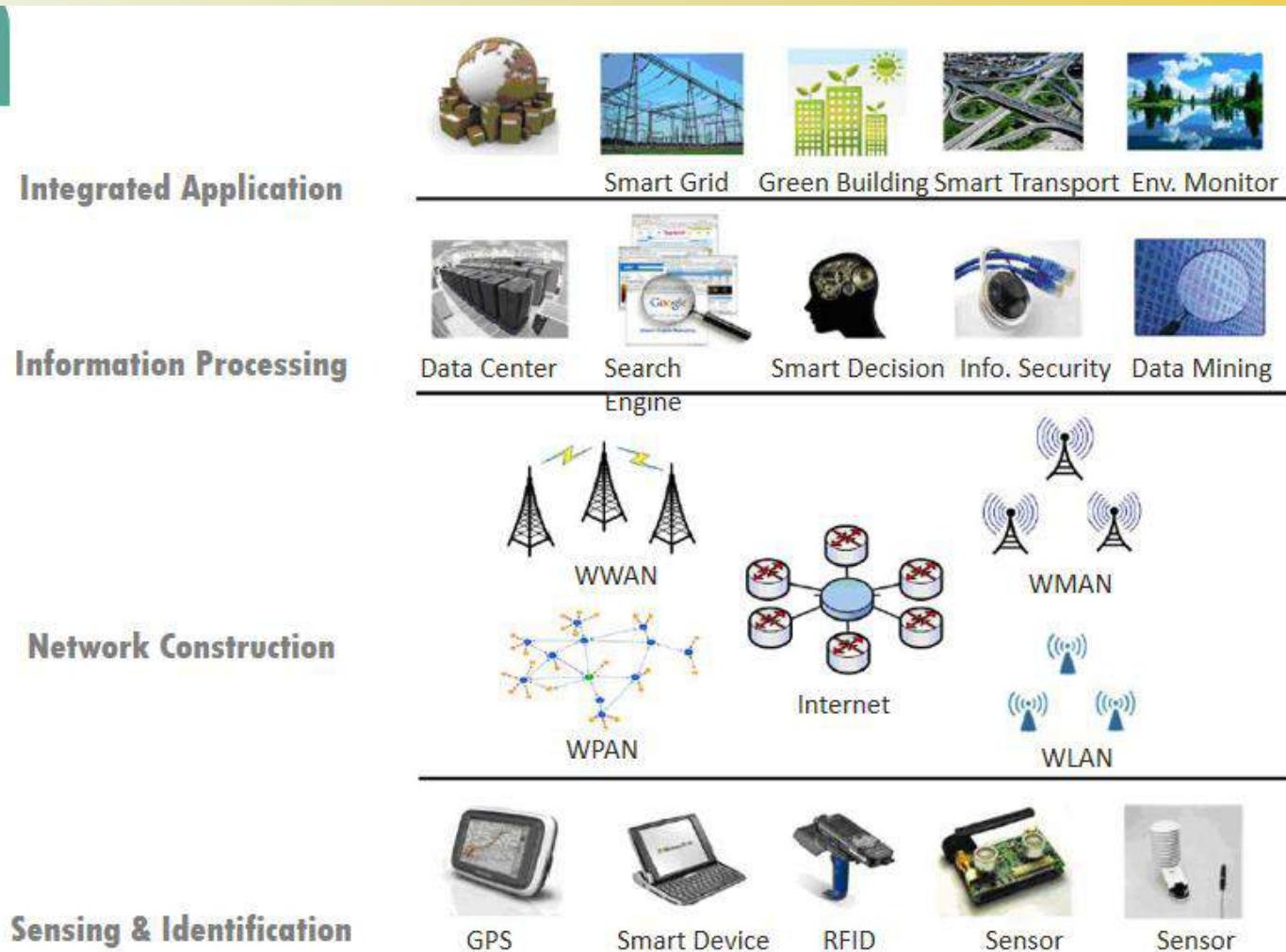


Video from <http://youtube.com/watch?v=HyyGSILAcE>

# Industry 4.0 vs Thailand 4.0

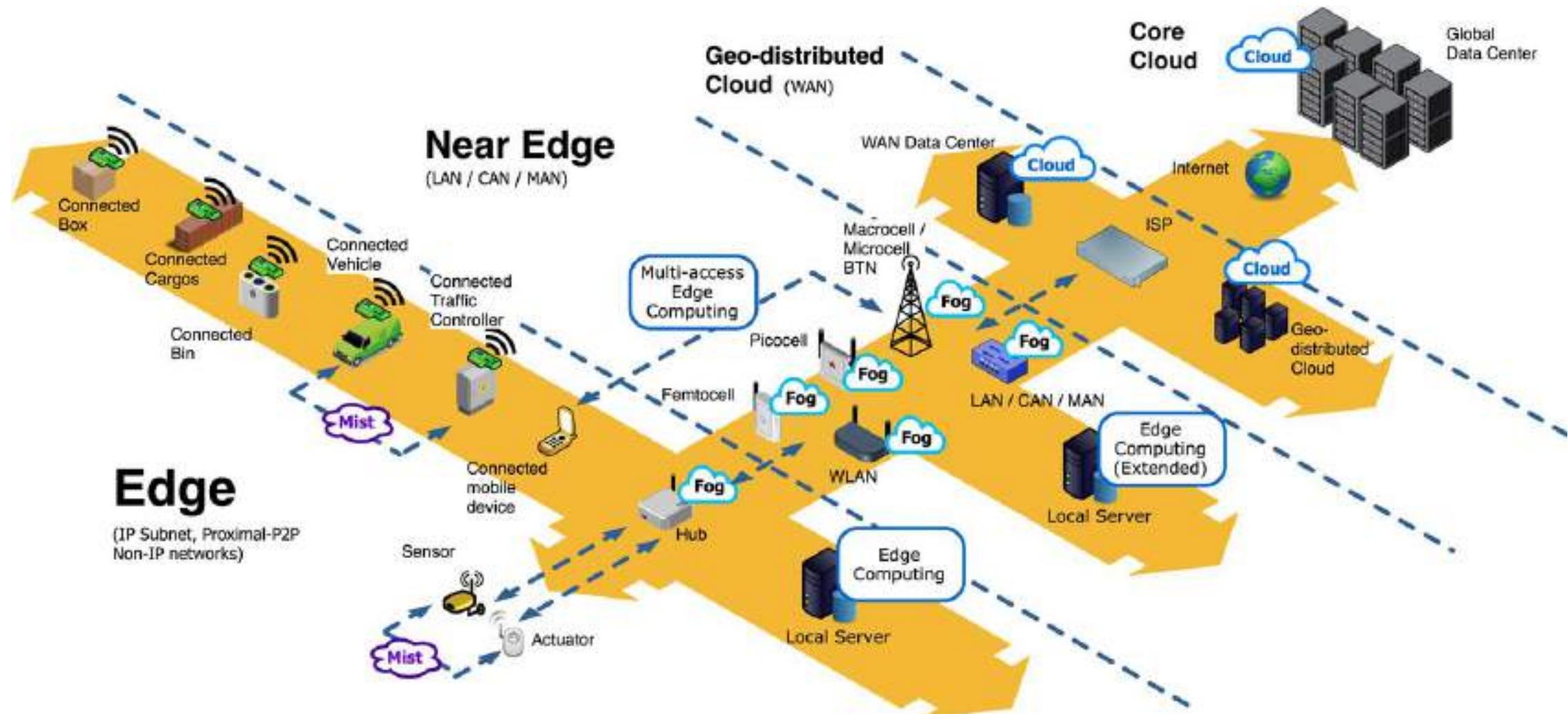


## Devices

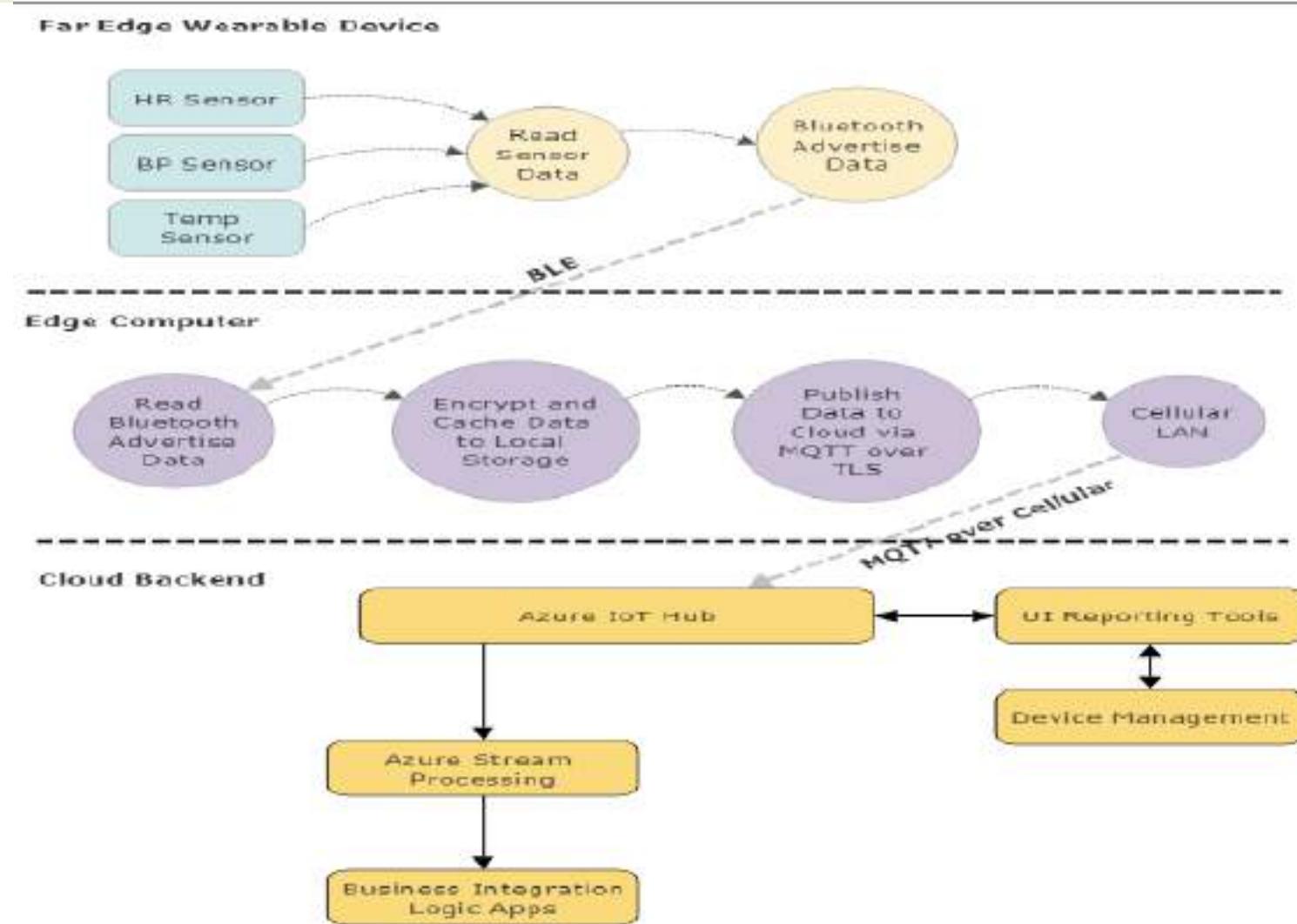


Slide from Dr. Kayarvizhy N.

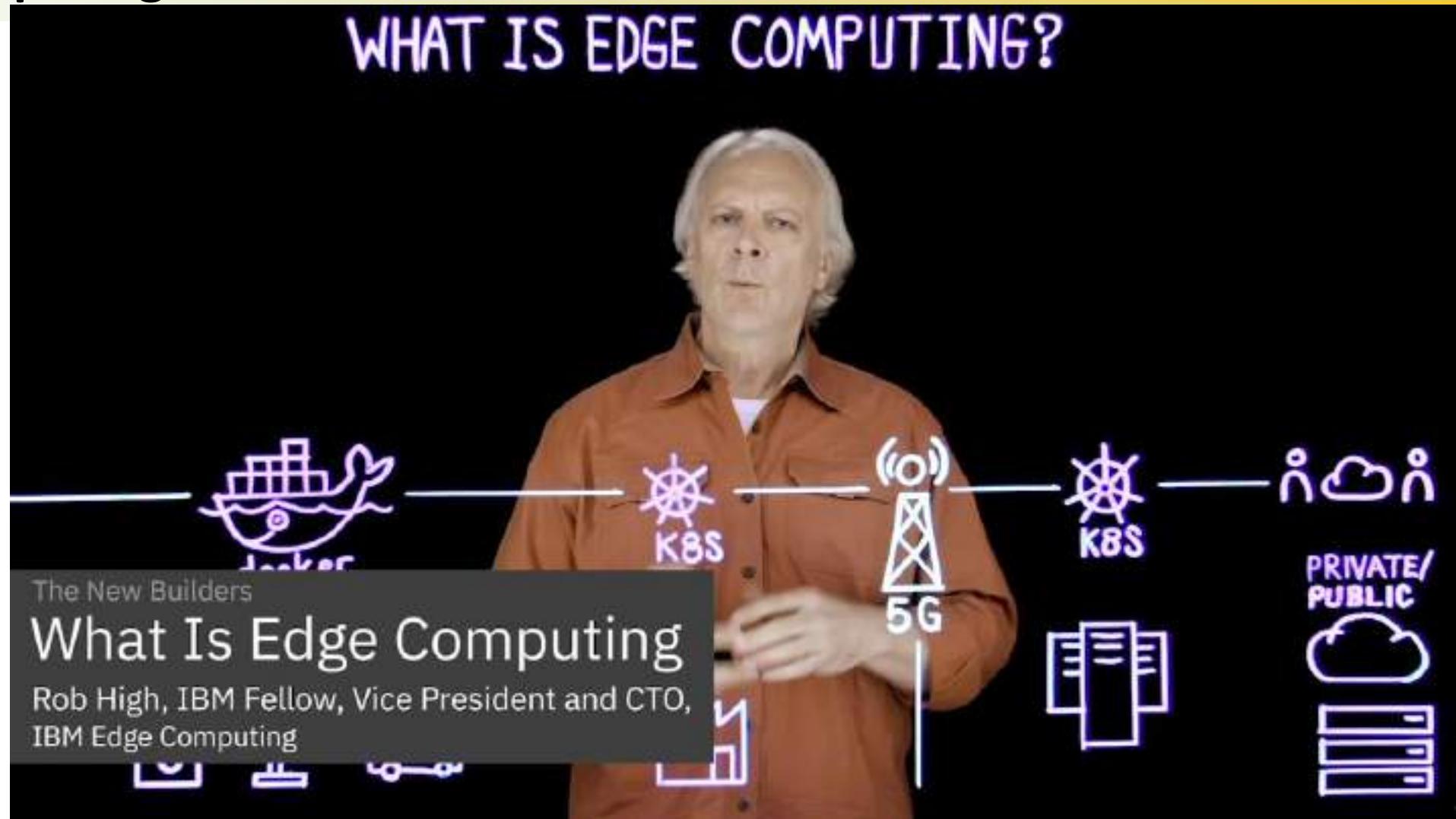
## Coverage



## Use Case



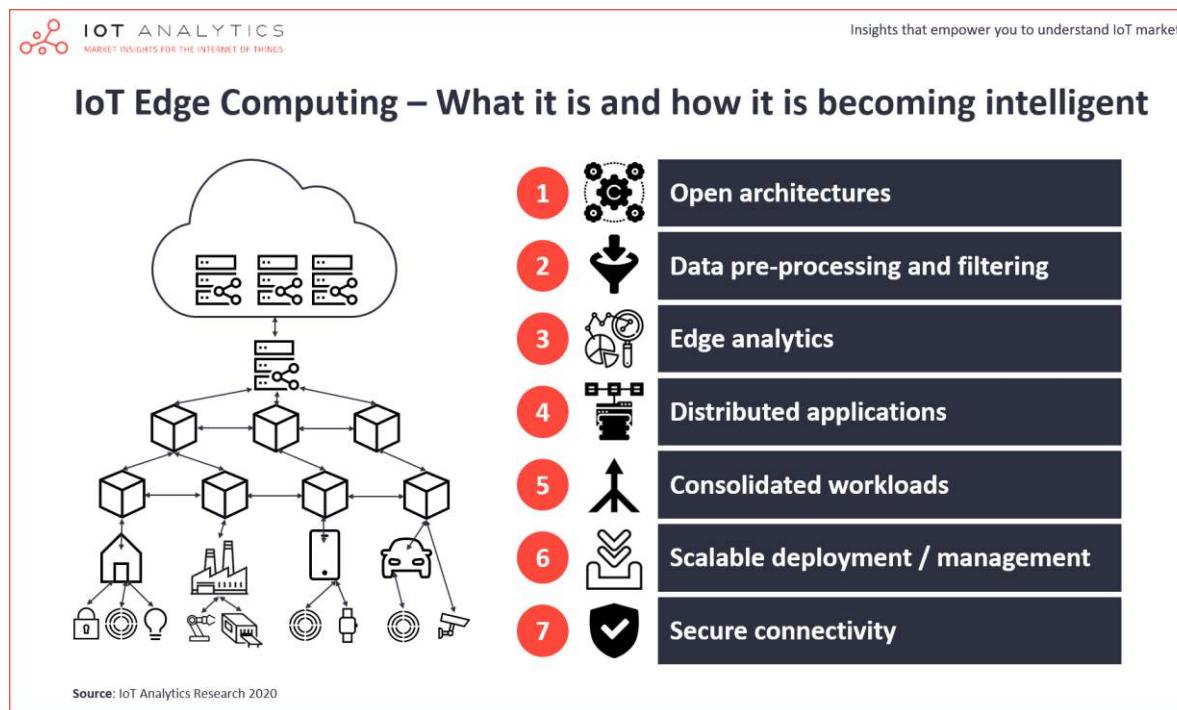
## Edge Computing



Video from: <https://www.youtube.com/watch?v=cEOUeItHDo>



## Edge Computing



Machine Learning and AI enable intelligent Edge

5G or ultrawide with low latency and powerful edge system enable real-time processing and transmission for AR/VR applications

<https://iot-analytics.com/iot-edge-computing-what-it-is-and-how-it-is-becoming-more-intelligent/>

[https://stlpartners.com/edge\\_computing/how-5g-and-edge-computing-will-transform-ar-vr-use-cases/](https://stlpartners.com/edge_computing/how-5g-and-edge-computing-will-transform-ar-vr-use-cases/)

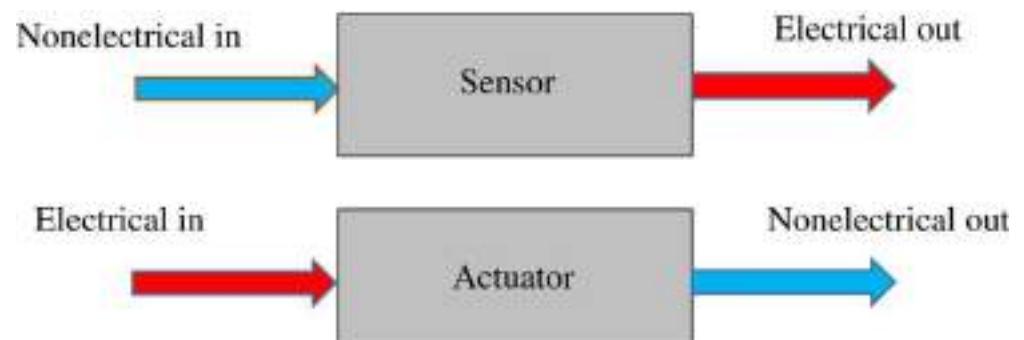
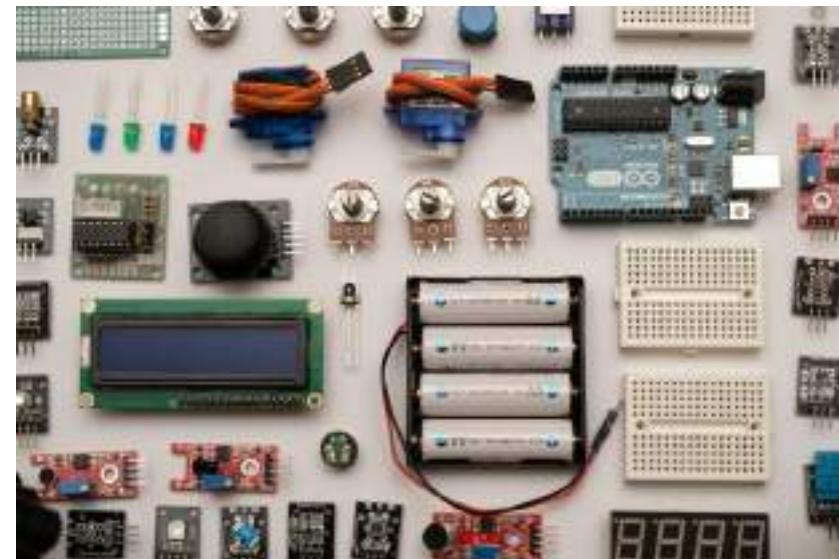
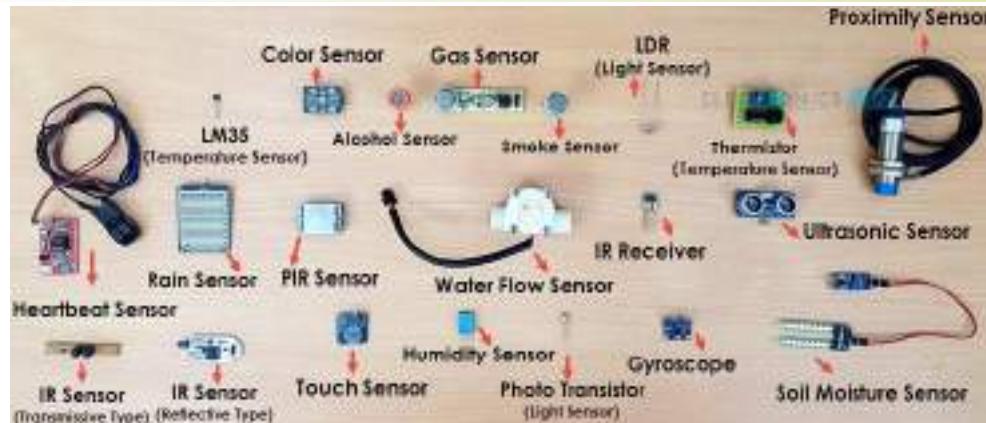
## IoT Components

- IoT Devices:  
Sensor, Actuator, Transceiver, Microcontroller
- IoT Connectivity
- Communication Protocols
- Data Storage and Processing: Edge/Fog/Cloud Platforms

## IoT Devices: Development Boards

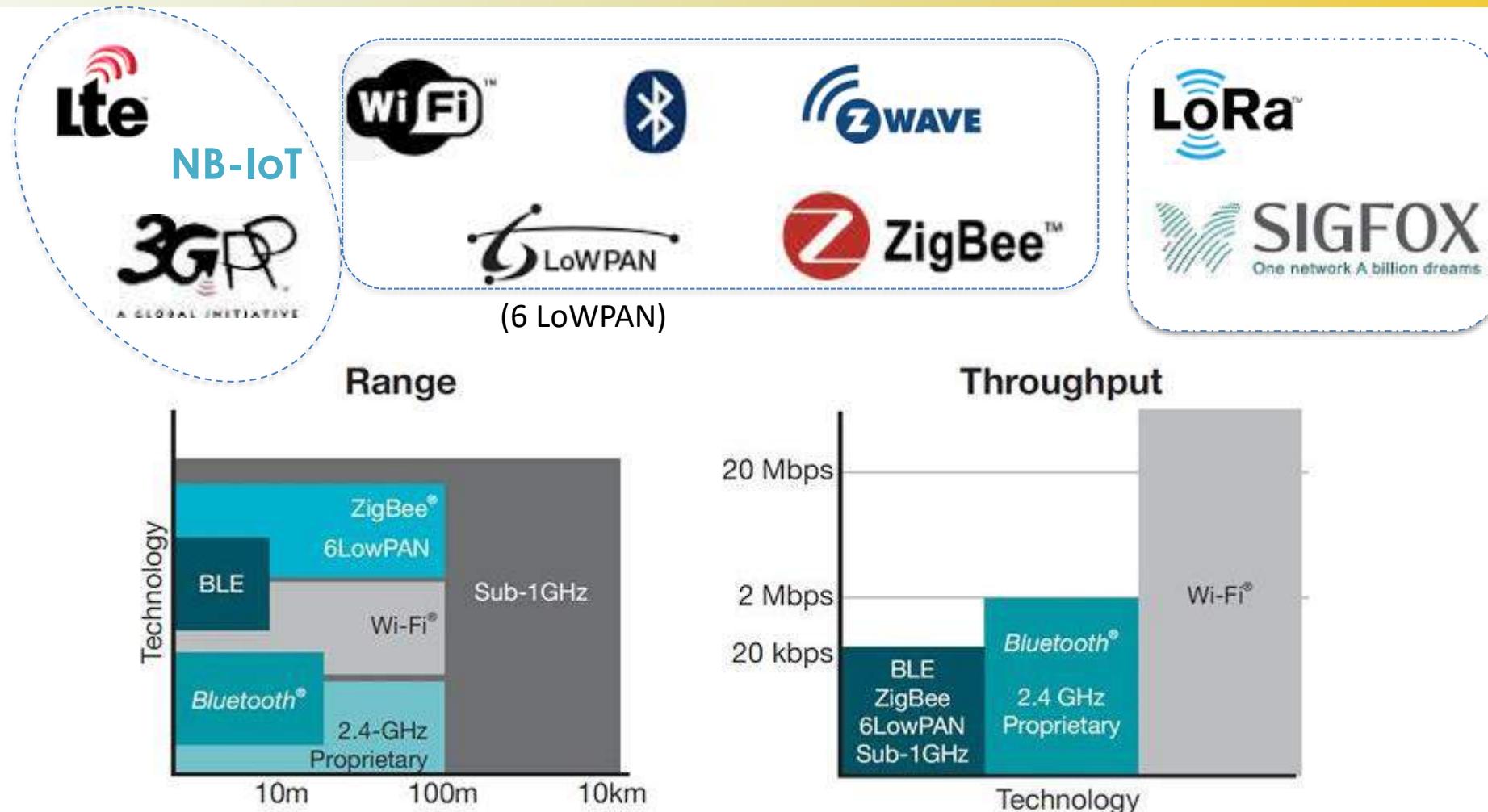


## IoT Devices: Sensors and Actuators



<https://www.electronicshub.org/different-types-sensors/>

## IoT Connectivity [1]





## IoT Connectivity [2]

Local Area Network Short Range Communication	Low Power Wide Area (LPWAN) Internet of Things	Cellular Network Traditional M2M														
<b>40%</b> Well established standards In building	<b>45%</b> Low power consumption Low cost Positioning	<b>15%</b> Existing coverage High data rate														
 Battery Life Provisioning Network cost & dependencies	High data rate Emerging standards	Autonomy Total cost of ownership														
	<table border="1"> <thead> <tr> <th></th><th>Europe</th><th>North America</th><th>China</th><th>Korea</th><th>Japan</th><th>India</th></tr> </thead> <tbody> <tr> <td>Frequency band</td><td>867-869MHz</td><td>902-928MHz</td><td>470-510MHz</td><td>920-925MHz</td><td>920-925MHz</td><td>865-867MHz</td></tr> </tbody> </table>		Europe	North America	China	Korea	Japan	India	Frequency band	867-869MHz	902-928MHz	470-510MHz	920-925MHz	920-925MHz	865-867MHz	
	Europe	North America	China	Korea	Japan	India										
Frequency band	867-869MHz	902-928MHz	470-510MHz	920-925MHz	920-925MHz	865-867MHz										

References:

<https://www.lora-alliance.org>

<https://web.facebook.com/nbiotthailand>

<https://web.facebook.com/groups/1643387719051153>

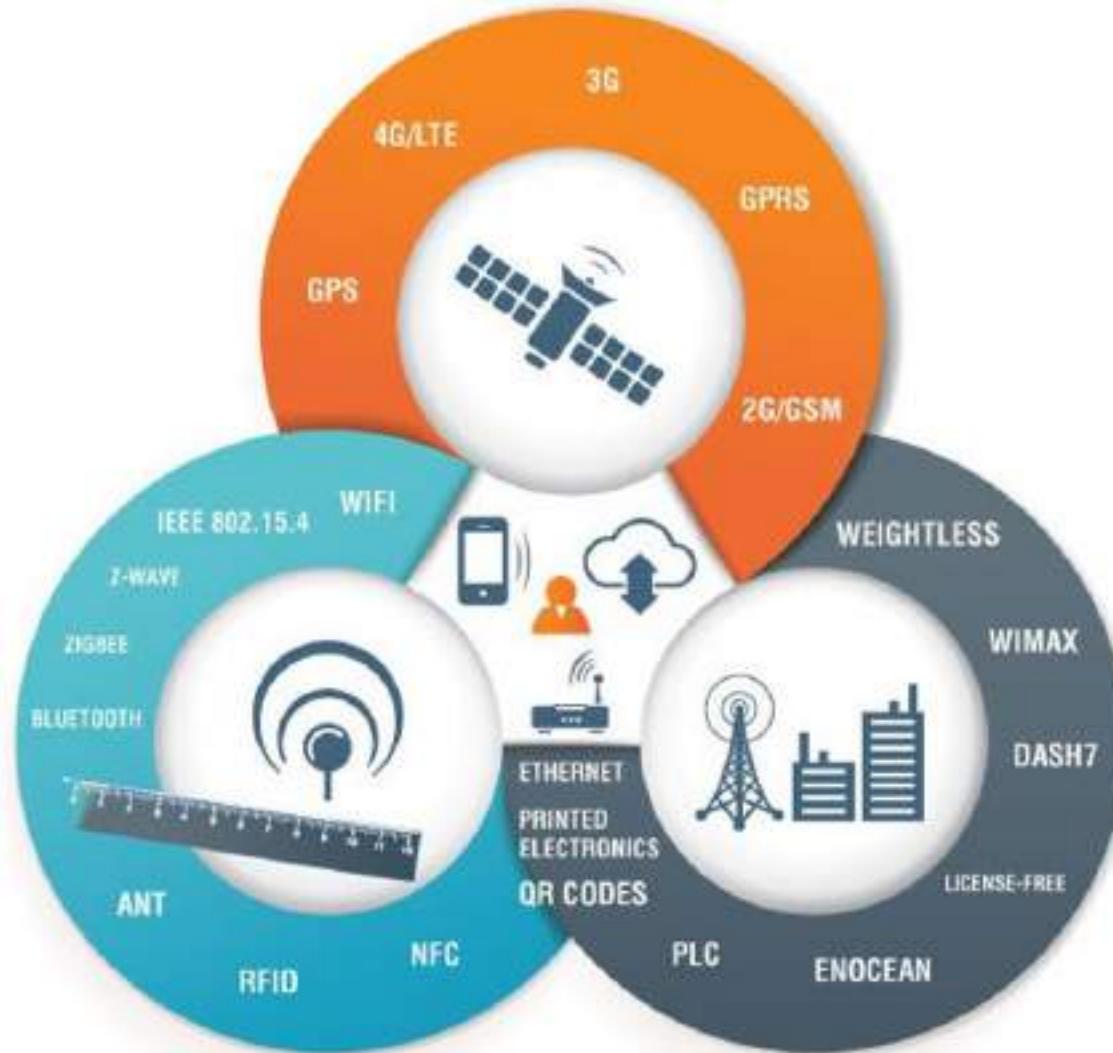
<https://web.facebook.com/ChiangMaiTheThingsNetwork>

<https://loraiot.cattelecom.com/site/usecase>

Thailand	433.05–434.79 MHz	EU433
	920–925 MHz	AS923

[https://lora-alliance.org/sites/default/files/2020-06/rp\\_2-1.0.1.pdf](https://lora-alliance.org/sites/default/files/2020-06/rp_2-1.0.1.pdf)

## IoT Connectivity [3]



## Communication Protocols (1/7)



### Common Architecture

Devices, gateways, cloud server, applications  
Standard interfaces, protocols, and open APIs

→ Scalability

### Common Service Functions

Registration, notifications, charging, DM...  
Cellular interworking, 3<sup>rd</sup> party platforms...

→ Manageability

### Key Standards Support

OneM2M and ETSI TC M2M  
IETF, 3GPP, OMA and BBF

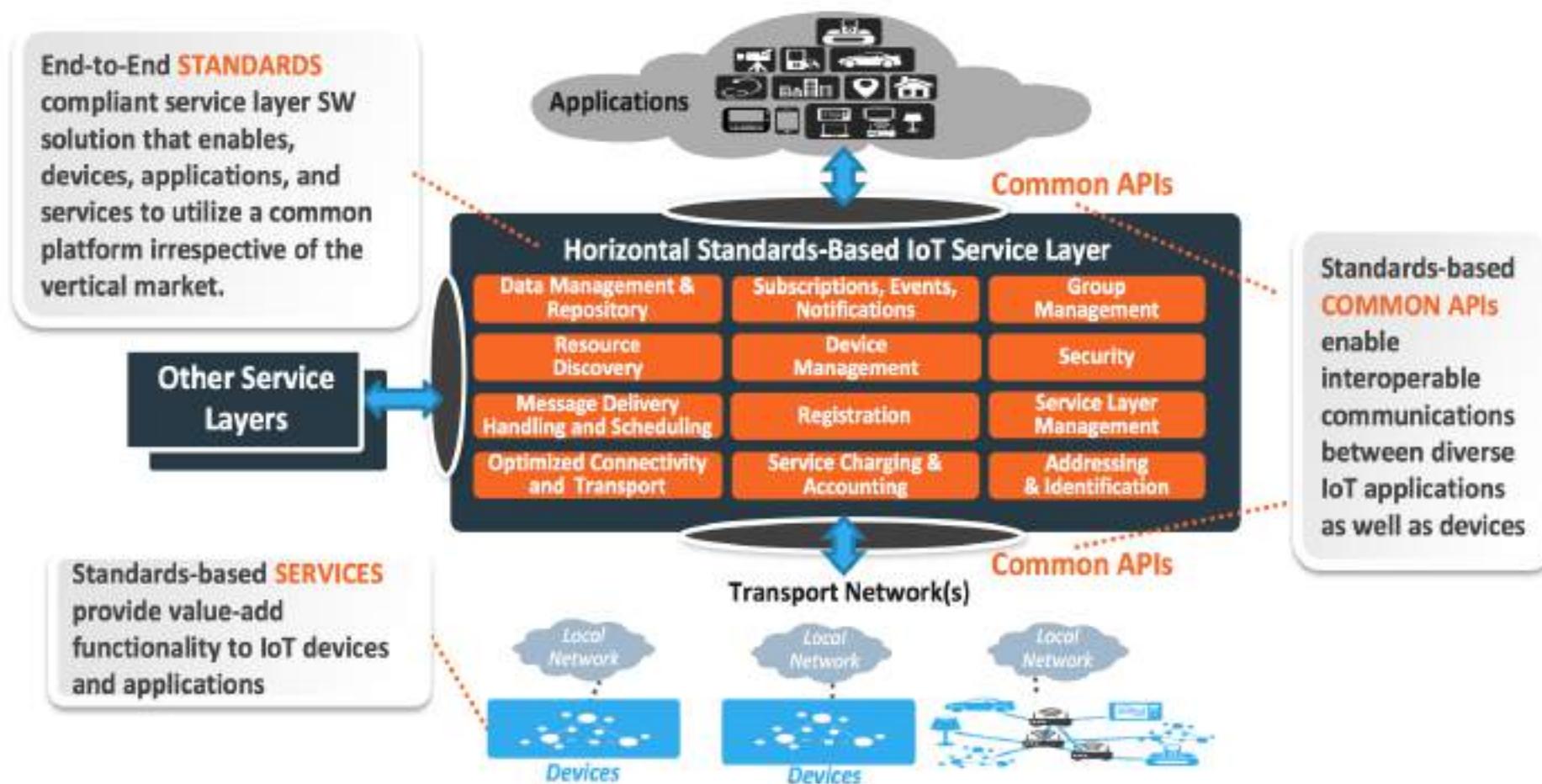
→ Inoperability

### Cross-Domain Interworking

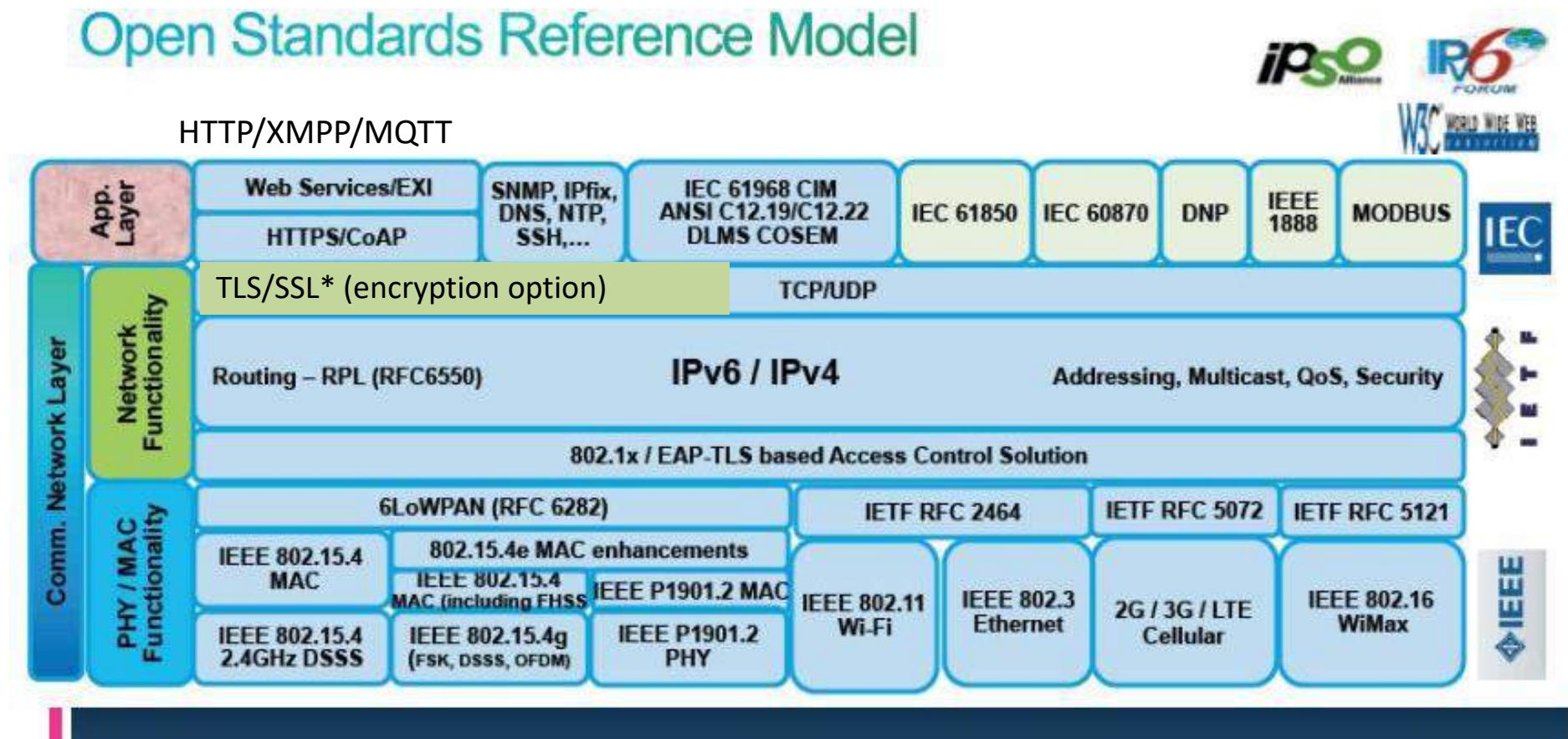
Cross-verticals (health, home, transportation)  
Various devices (cellular, Wi-Fi, ZigBee,...)

→ Versatility

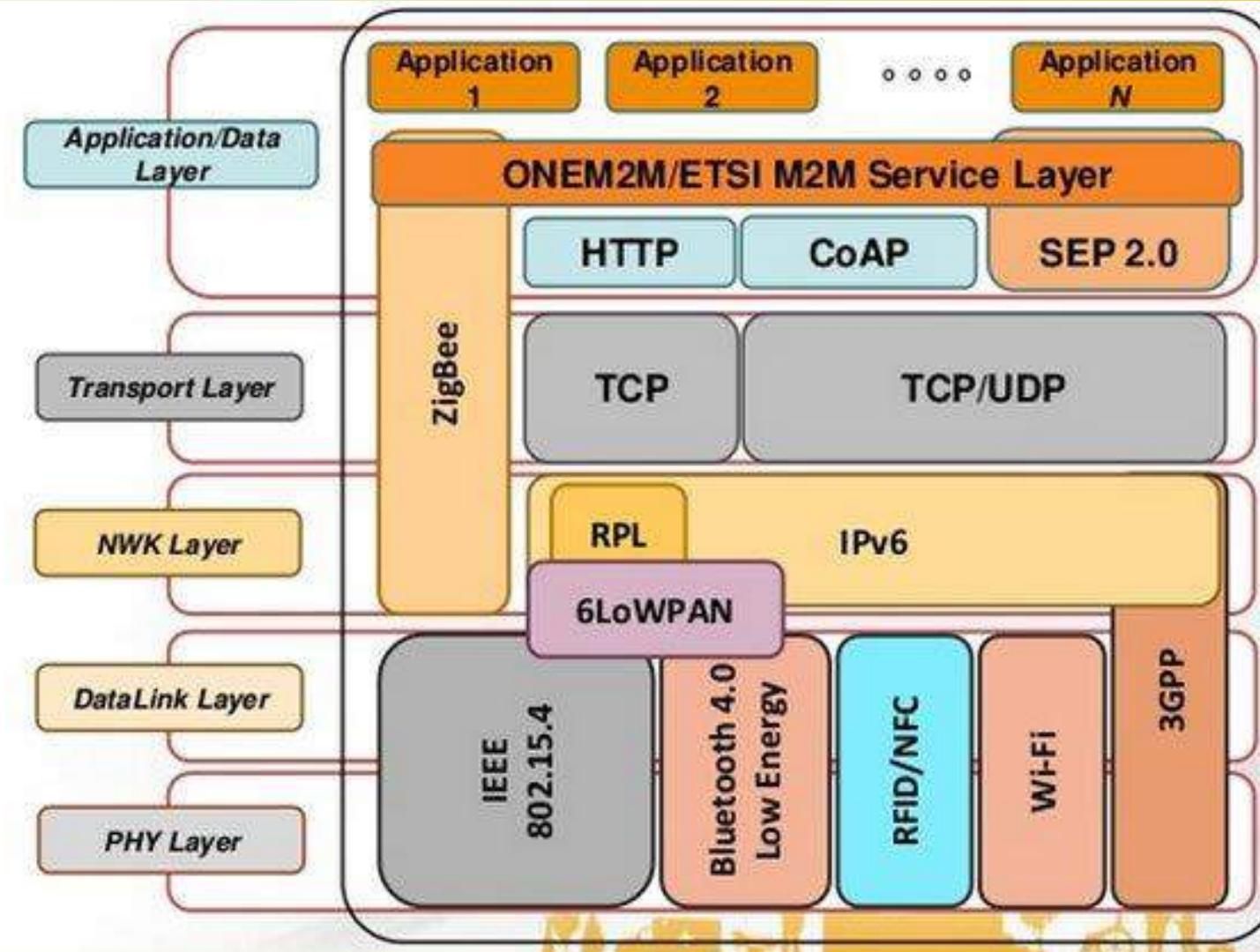
## Communication Protocols (2/7)



## Communication Protocols (3/7)



## Communication Protocols (4/7)





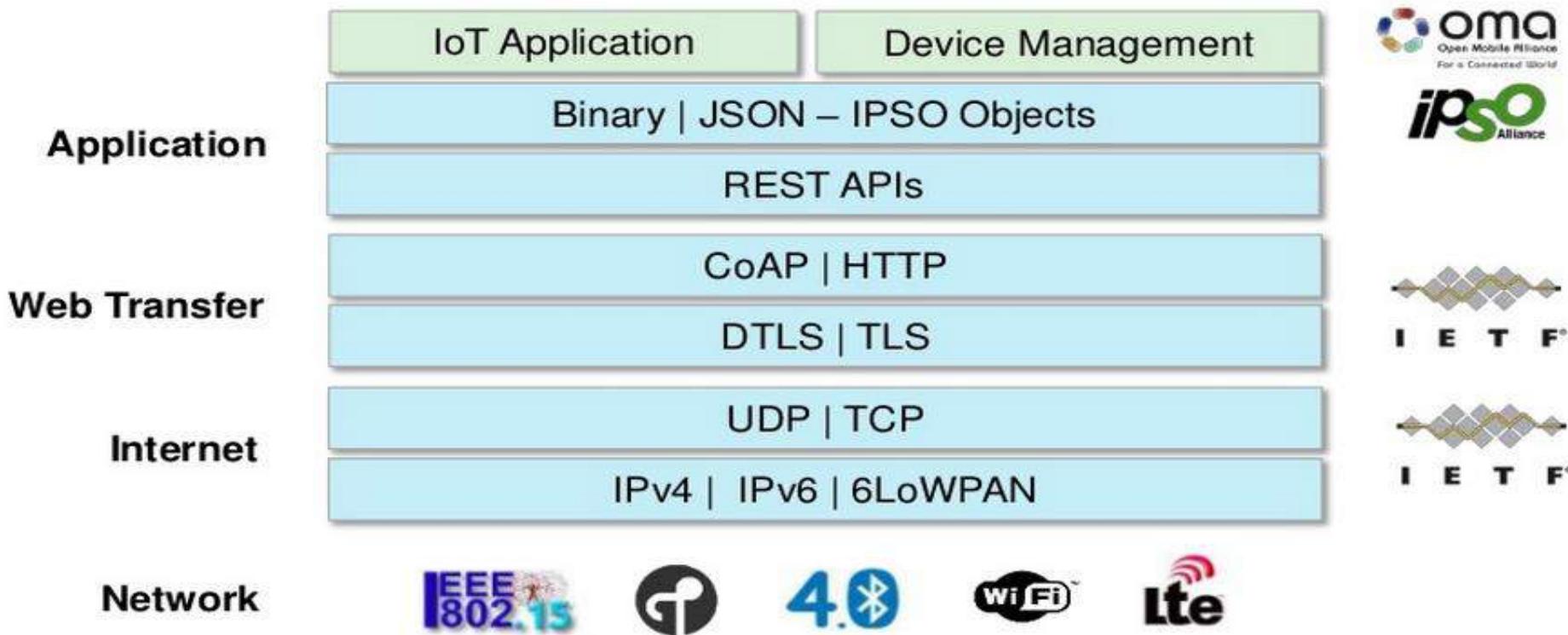
## Communication Protocols (5/7)

IoT Ecosystem [Ref: ces570-15@www.cse.wustl.edu/]

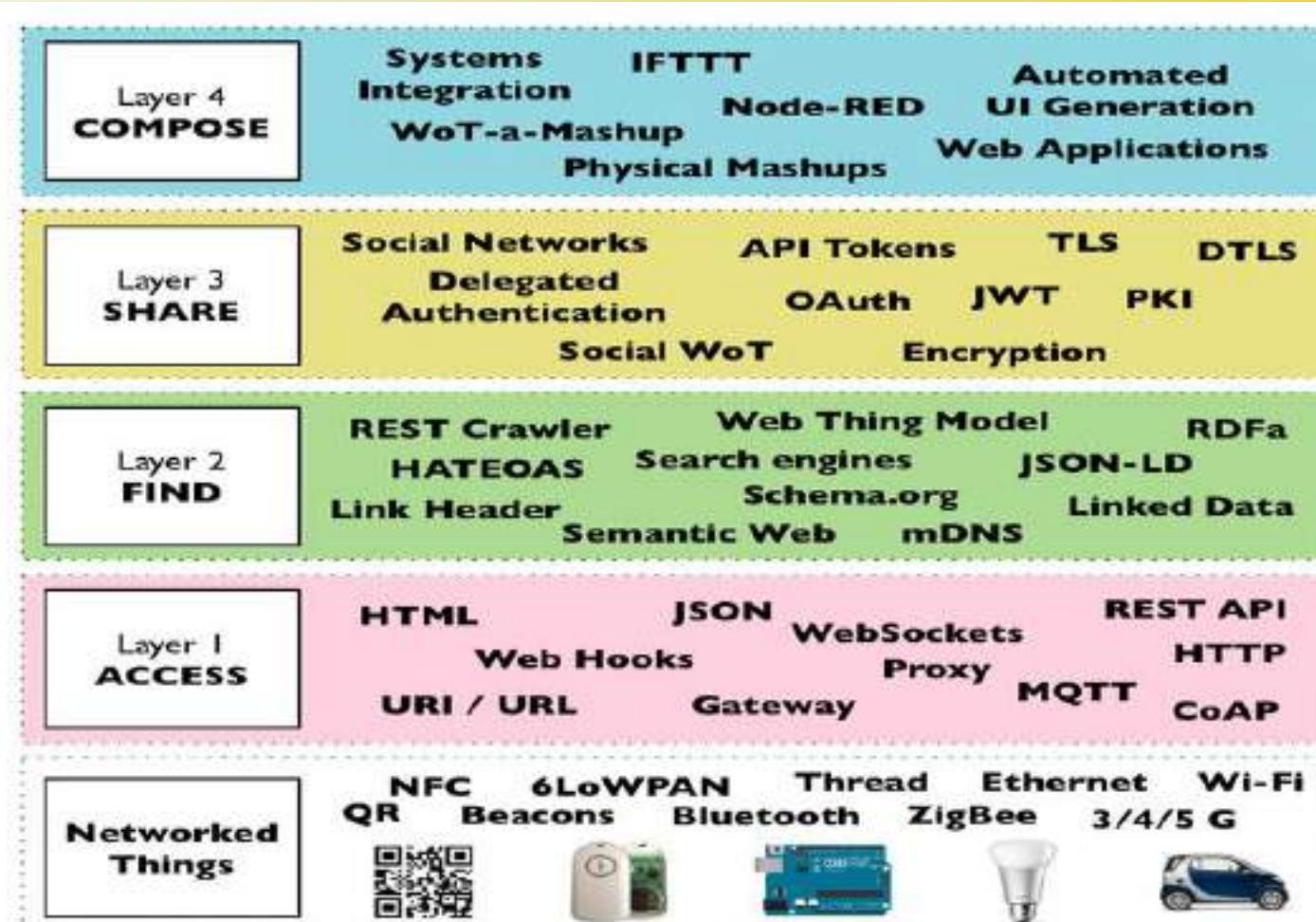
Applications	Smart Health, Smart Home, Smart Grid Smart Transport, Smart Workspaces, ...	Security	Management
Session	<b>MQTT</b> , CoRE, DDS, AMQP , ...	TCG, Oath 2.0, <b>SMACK</b> , SASL, ISASecure, ace, CoAP, DTLS, Dice	IEEE 1905, IEEE 1451, ...
Routing	<b>6LowPAN</b> , <b>RPL</b> , 6Lo, 6tsch, Thread, 6-to-nonIP , ...		
Datalink	WiFi, Bluetooth Smart, ZigBee Smart, Z-Wave, DECT/ULE, 3G/LTE, NFC, Weightless, <b>HomePlug GP</b> , 802.11ah, <b>802.15.4</b> , G.9959, WirelessHART, DASH7, ANT+, LoRaWAN, ...		
Software	Mbed, Homekit, AllSeen, IoTivity, ThingWorks, EVRYTHNG , ...		
Operating Systems	Linux, Android, Contiki-OS, TinyOS, ...		
Hardware	ARM, <b>Arduino</b> , Raspberry Pi, ARC-EM4, Mote, Smart Dust, Tmote Sky, ...		



## Communication Protocols (6/7)

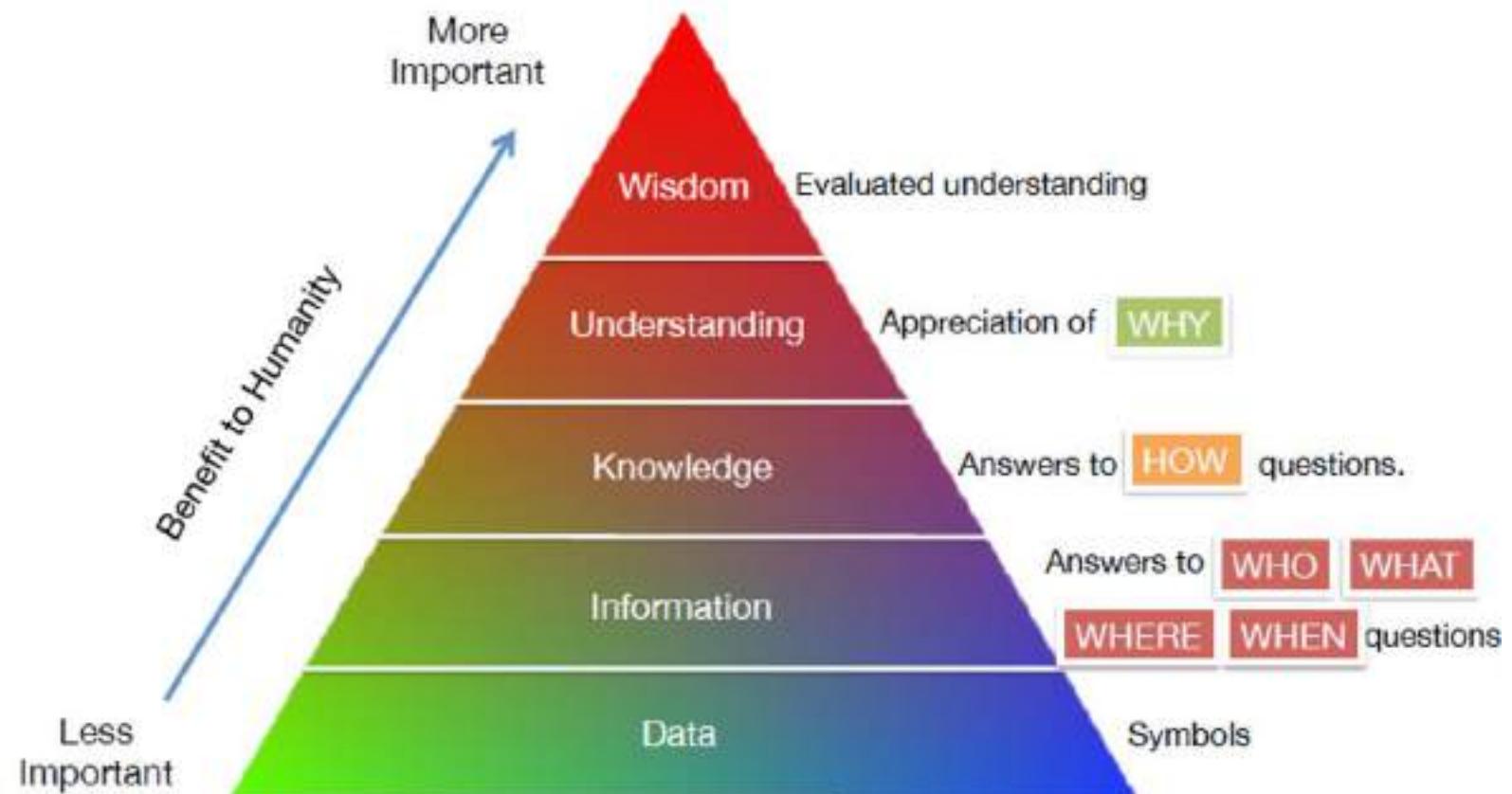


## Communication Protocols (7/7)



Source: Building the Web of Things: book.webofthings.io  
Creative Commons Attribution 4.0

## Data Analysis



The more data that is created, the better understanding and wisdom people can obtain.

## Storage Capacity

### Information from the Internet of Things:

We have gone beyond the decimal system

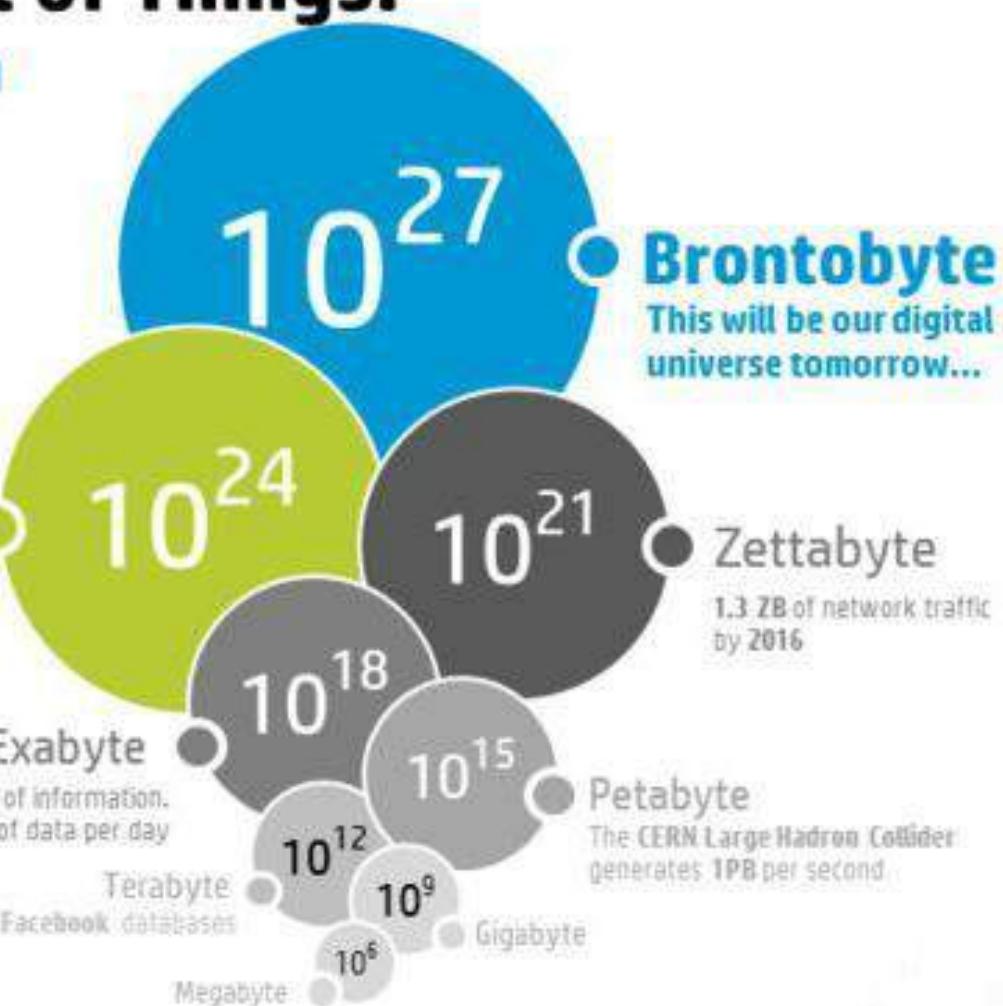
Today data scientist uses **Yottabytes** to describe how much government data the NSA or FBI have on people altogether.

In the near future, **Brontobyte** will be the measurement to describe the type of sensor data that will be generated from the IoT (Internet of Things)

**Yottabyte**  
This is our digital universe today  
= 250 trillion of DVDs

1 EB of data is created on the internet each day = 250 million DVDs worth of information.  
The proposed Square Kilometer Array telescope will generate an EB of data per day

500TB of new data per day are ingested in Facebook databases



## Realtime Database + Big Data Platforms

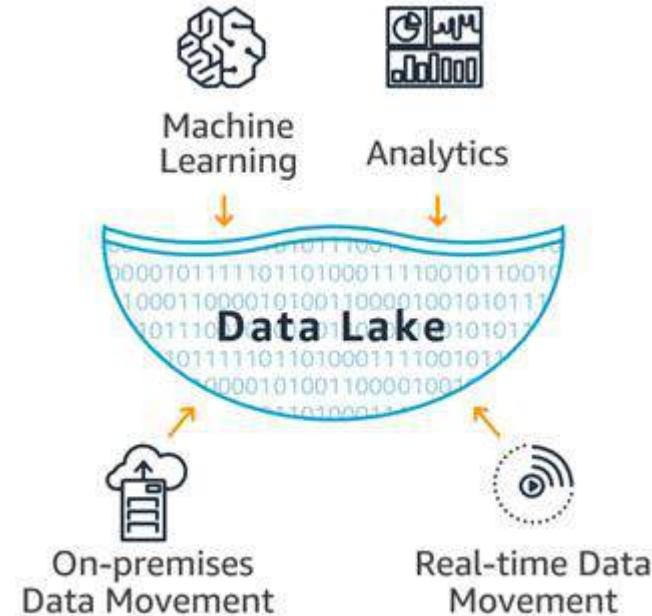
NoSQL DB



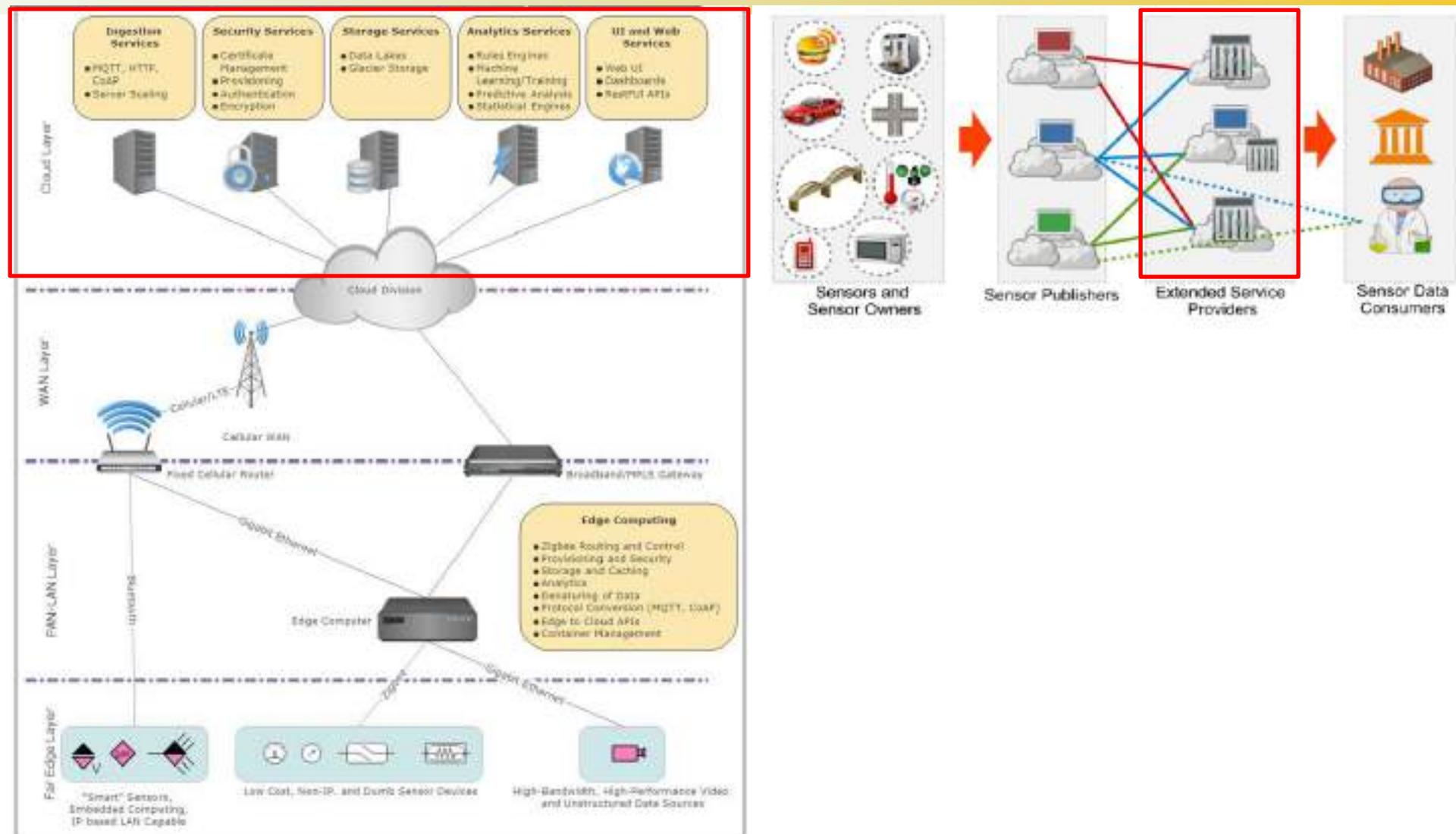
*cassandra*



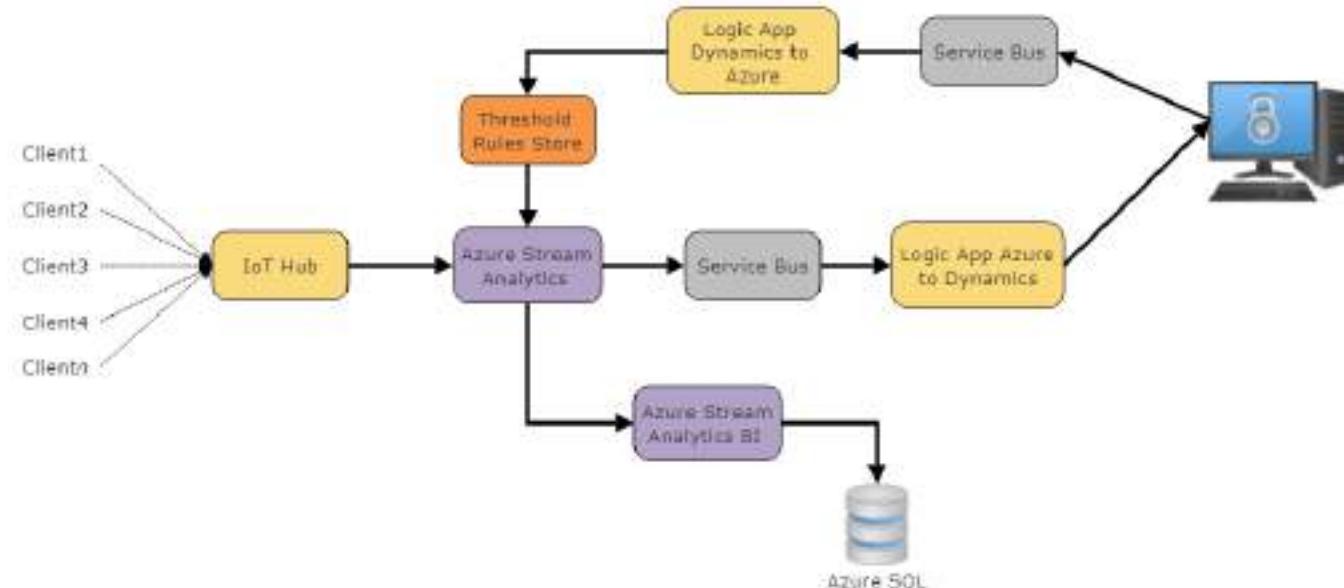
Big Data Cluster



## Cloud (Sensing as-a-service Model)

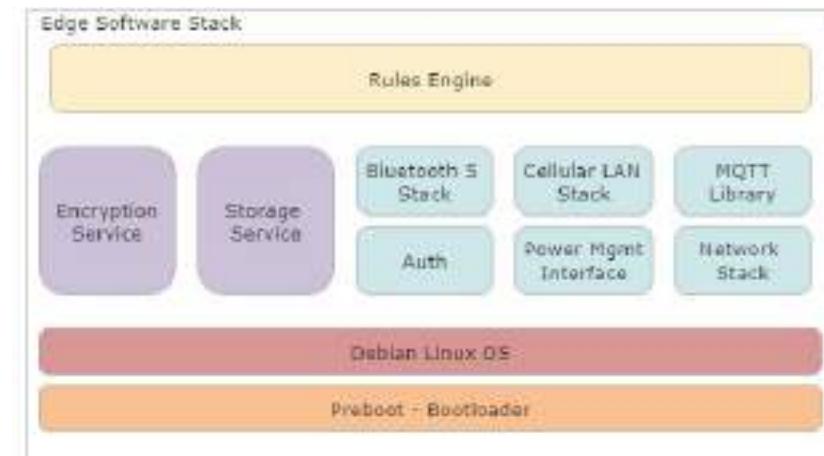


## Cloud (Sensing as-a-service Model)

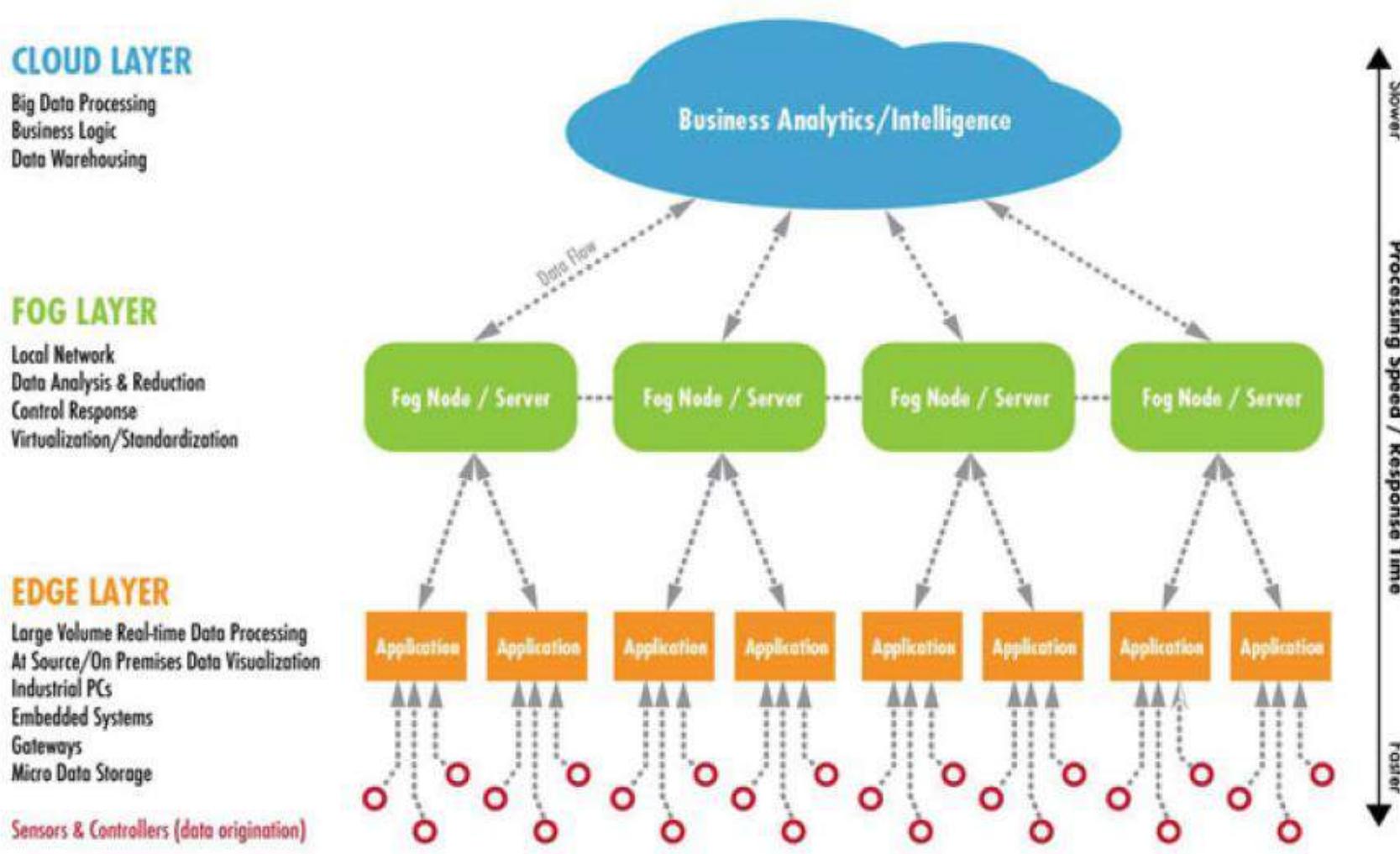


### Cloud Providers

- IBM BlueMix
- AWS IoT
- Azure IoT
- NETPIE

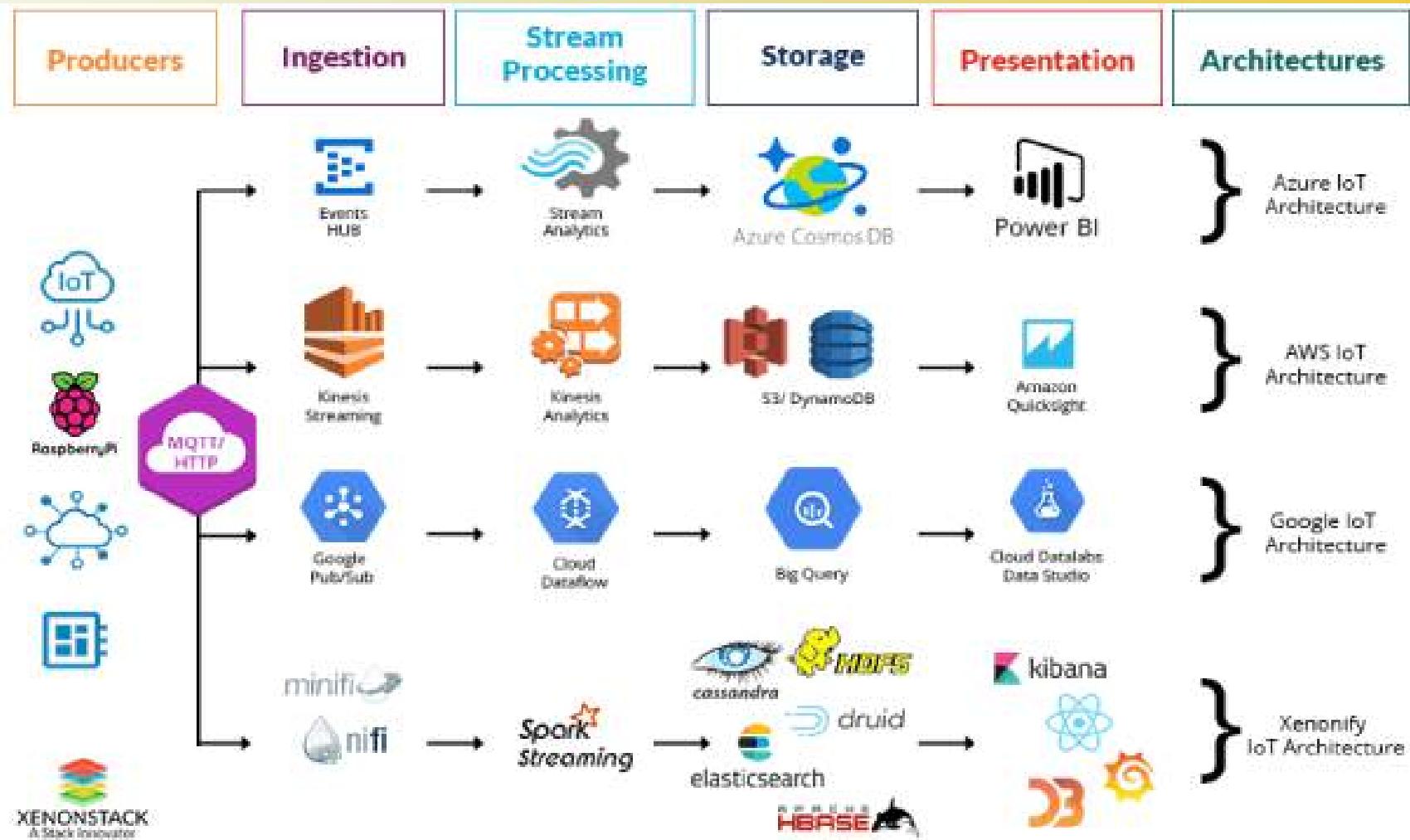


## Cloud vs Fog vs Edge Layers



<https://www.analyticsinsight.net/top-edge-computing-predictions-to-look-out-for-in-2022/>

## IoT Analytics Platform for Real-Time Data Ingestion



<https://www.xenonstack.com/blog/iot-analytics-platform>

- Technological Standardization in most areas still remains fragmented.
- Managing and fostering rapid innovation is a challenge for governments
- Privacy and security of IoT System and Application
- Absence of governance
- Vulnerability to internet attack
- Resource constraints e.g. energy or storage capacity
- Lack of interoperability

Lack of interoperability causes a major technological issue:

- Impossibility to plug noninteroperable IoT devices into heterogeneous IoT platforms
- loss of business opportunities, especially for small innovative enterprises, which cannot afford to provide their solution across multiple platforms and protocols.

An IoT governance framework should ensure data integrity and data security for information shared by all IoT devices in the enterprise network. It should also maintain the trusted source of information across the different layers of the IoT architecture.

Cross-platform  
Cross-standard  
Cross-domain IoT services and applications

Ref: Abderrazak Hachani Slides

- [VB17] Ovidiu Vermesan and Joel Bacquet, “**Cognitive Hyperconnected Digital Transformation: Internet of Things Intelligence Evolution**”, River Publishers, 2017.
- [BHC+18] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, “The Industrial Internet of Things (IIoT): An Analysis Framework”, Computers in Industry 101, October 2018.
- [LEA20] Perry Lea, “**IoT and Edge Computing for Architects: Implementing edge and IoT systems from sensors to clouds with communication systems, analytics, and security**”, 2<sup>nd</sup> Edition, Packt Publishing, 2020.



# ITCS447

## Lecture 2

### Basic Electronics for IoT

Asst. Prof. Dr. Thitinan Tantidham



มหาวิทยาลัยมหิดล  
Mahidol University

ห้ามมิให้นักศึกษาทำข้า ดัดแปลง หรือใช้ประโยชน์จากการอันมีลิขสิทธิ์ปราการอยู่ในระบบการศึกษาอิเล็กทรอนิกส์ (E-Learning) ของมหาวิทยาลัย ไม่ว่าจะทั้งหมดหรือบางส่วน โดยไม่ได้รับอนุญาตจากมหาวิทยาลัย นอกจากนี้จากการศึกษาส่วนบุคคล ทั้งนี้ การทำข้า ดัดแปลง หรือเผยแพร่ต่อสาธารณะนั้นงานอันมีลิขสิทธิ์ จะมีโทษปรับตั้งแต่ 20,000 บาท ถึง 200,000 บาท และหากเป็นการกระทำเพื่อการค้า จะมีโทษจำคุกตั้งแต่ 6 เดือน ถึง 10 ปี หรือปรับตั้งแต่ 100,000 บาท ถึง 800,000 บาท หรือทั้งจำทั้งปรับ

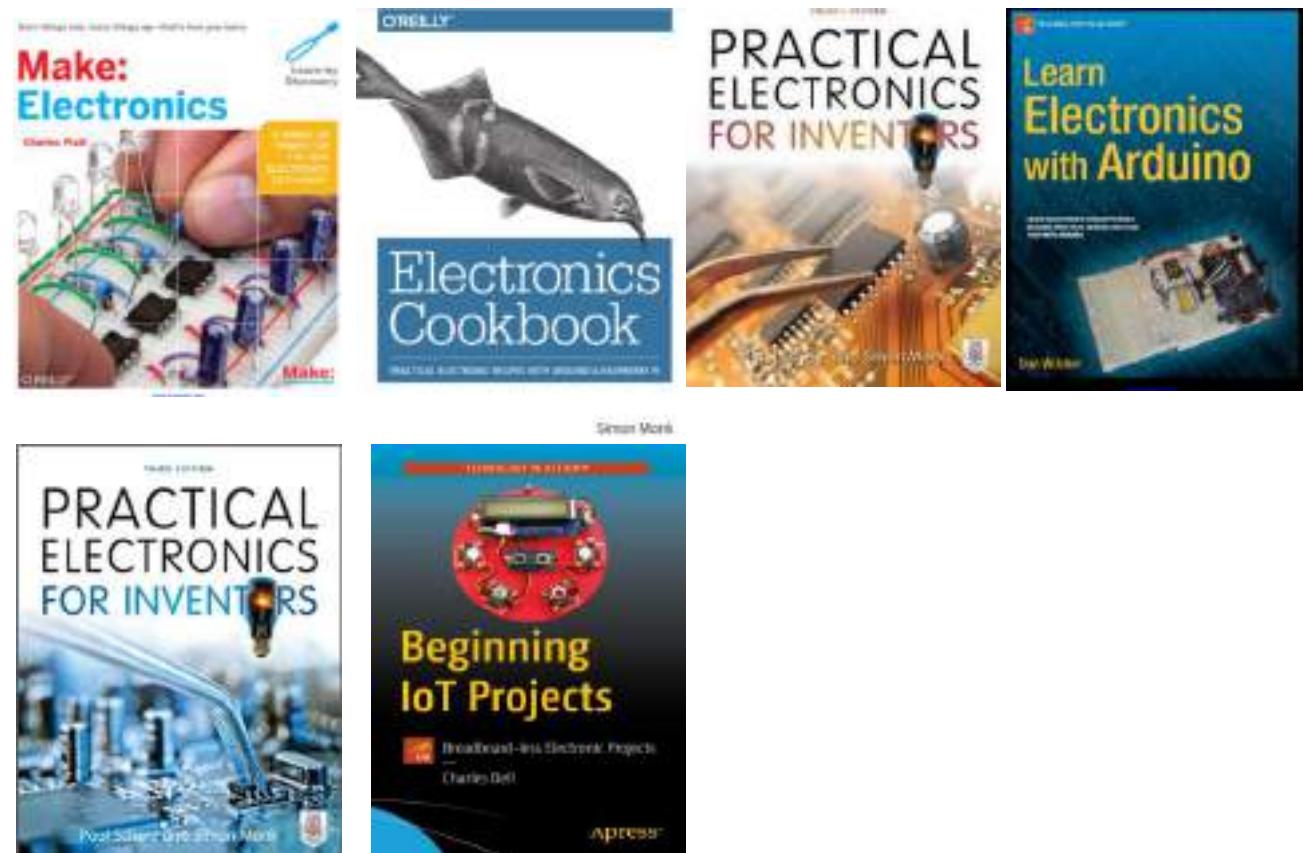
Unless allowed by Mahidol University, the User shall not copy, modify, or exploit in part or in whole of the copyrighted materials on the Platform, other than for your own individual study. The copy, modify or communication to public of the copyrighted materials shall be inflicted with a fine from 20,000 Baht up to 200,000 Baht. If the offence is committed with the commercial purpose, the offender shall be inflicted with imprisonment for a term from 6 months up to 10 years or a fine from 100,000 Baht up to 800,000 Baht or both imprisonment and fine.



# Part I: Outline



- Electrical Circuits
- Electronics Components
- Resistors
- Ohm's Law
- Kirchhoff's Law
- Series-Parallel Circuits
- Dividers
- LED

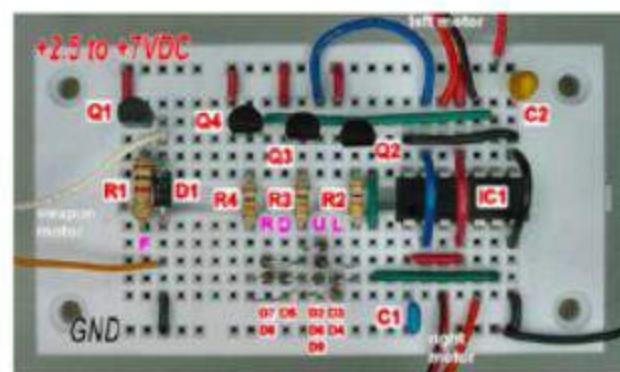


<https://iot.electronicsforu.com/>

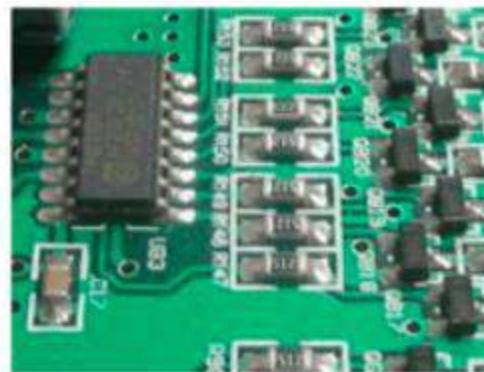
# Electrical Circuits



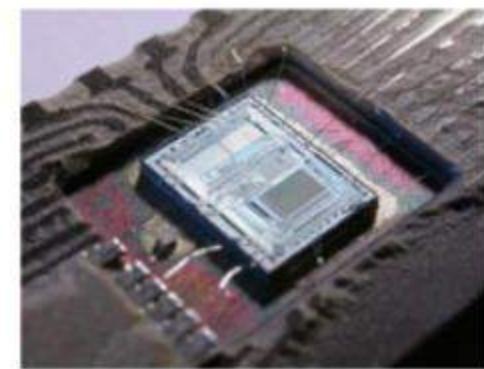
- A circuit consists of electrical or electronic components interconnected with metal wires.
- Every electrical or electronic device is a circuit.



Breadboard  
Prototype



Printed  
PCB (Printed Circuit Board)



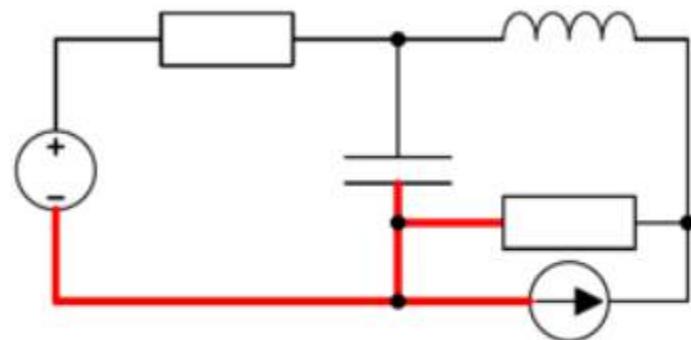
Integrated

- The function of the circuit is determined by which components are used and how they are interconnected: the physical positioning of the components usually has hardly an effect.

# Circuit Diagrams



- A circuit diagram shows the way in which the components are connected.
  - Each component has a special symbol
  - The interconnecting wires are shown as lines.
- A node in a circuit is all the points that are connected together via the interconnecting wires. One of the four nodes in the diagram is coloured red.



- The function of the circuit is determined by which components are used and how they are interconnected: the physical positioning of the components usually has hardly an effect.

# Electrical Charge



- **Charge** is an **electrical property** of the atomic particles of which matter consists, measured in **coulombs (C)**.
- The charge **e** on one electron is **negative** and equal in magnitude to  **$1.602 \times 10^{-19}$  C** which is called as **electronic charge**.
- The **charges** that occur in nature are **integral multiples** of the electronic charge.
- Consequently, charge never accumulates in a **conductor**. Everywhere in a conducting path stays electrically neutral at all times.



## Potential Difference

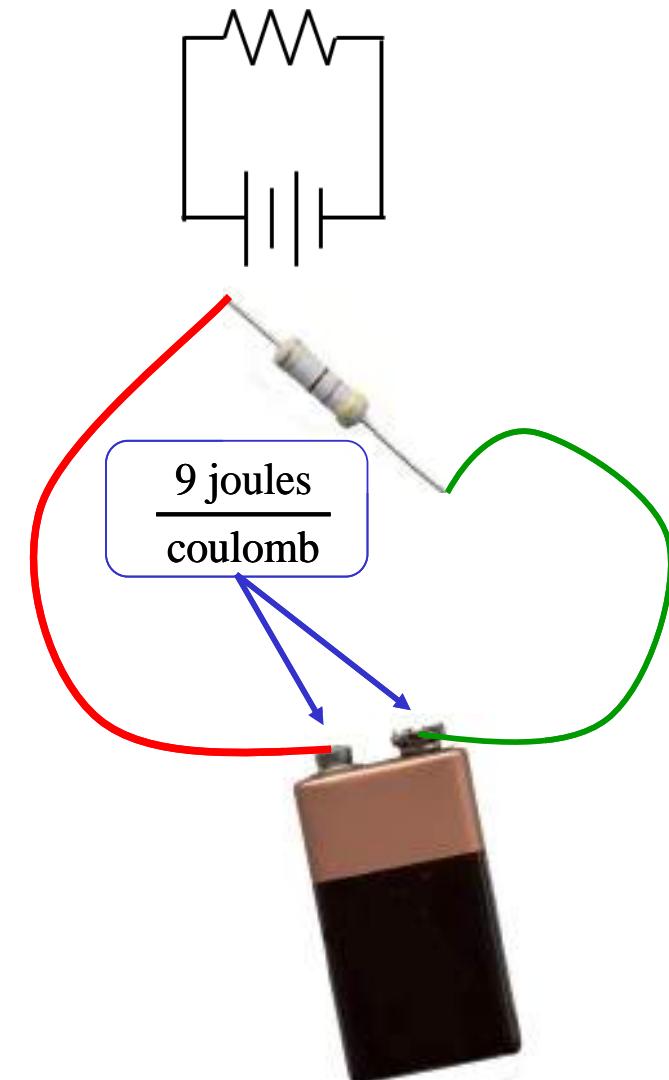
- The volt is a measure of the amount of work or energy needed to move an electric charge.
- The metric unit of work or energy is the **joule (J)**. One joule = 0.7376 ft·lbs.
- The potential difference (or voltage) between two points equals **1 V(olt)** when 1 J(oule) of energy is expended in moving 1 C(oulomb) of charge between those two points.

$$1 \text{ V} = 1 \text{ J} / 1 \text{ C}$$

The unit of voltage is the volt. An AA battery has about 1.5V across its terminals. An Arduino operates at 5V, while an ESP32 or a Raspberry Pi operate at 3.3V, although it requires a 5V supply that it reduces to 3.3V.

<https://www.youtube.com/watch?v=1xPjES-sHwg>

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.



- With respect to the positive and negative terminals of the voltage source, current has direction.
- When free electrons are considered as the moving charges we call the direction of current **electron flow**. Electron flow is from the negative terminal of the voltage source through the external circuit back to the positive terminal.
- Conventional current** is considered as the motion of positive charges. Conventional current flows in the opposite direction from electron flow (**positive to negative**).

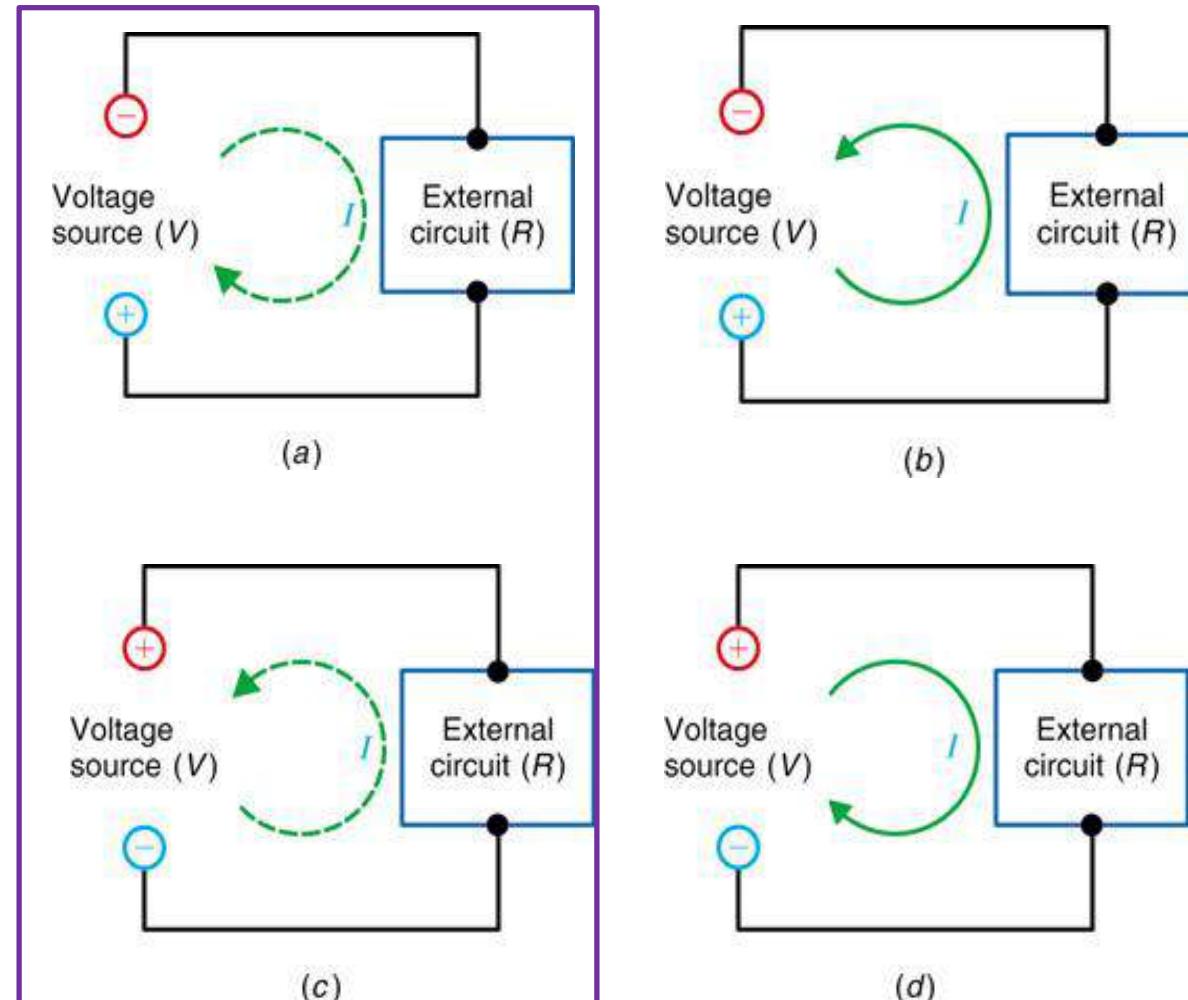
# Current (2/5)



[https://learning.oreilly.com/videos/basic-electronics-for/9781788628679/9781788628679-video2\\_2/](https://learning.oreilly.com/videos/basic-electronics-for/9781788628679/9781788628679-video2_2/)

Direction of  $I$  in a closed circuit, shown for electron flow and conventional current. The circuit works the same way no matter which direction you consider.

- (a) Electron flow indicated with dashed arrow in diagram.
- (b) Conventional current indicated with solid arrow.
- (c) Electron flow as in (a) but with reversed polarity of voltage source.
- (d) Conventional  $I$  as in (b) but reversed polarity for  $V$ .



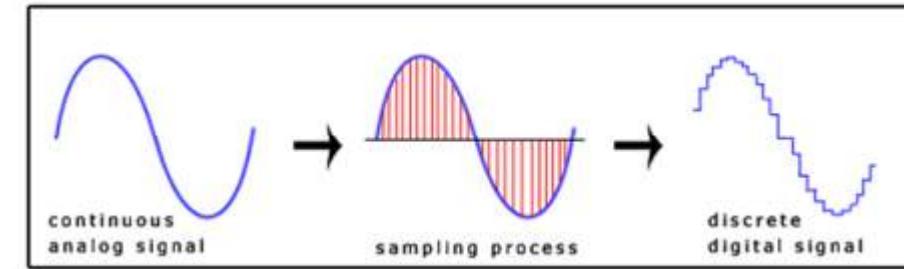
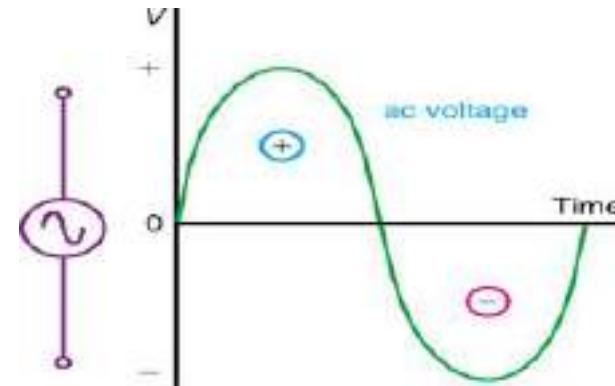
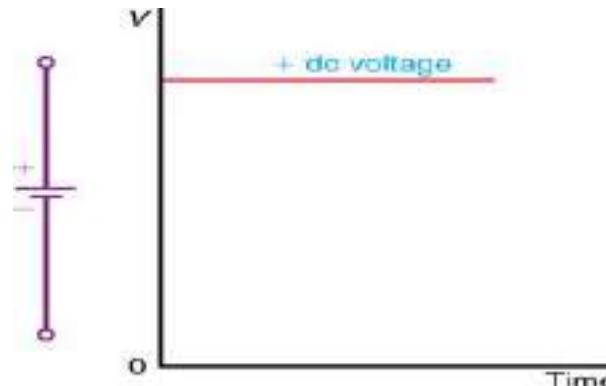


## Direct Current and Alternating Current (1)

- **Direct Current (dc)** flows in only one direction.
- **Alternating Current (ac)** periodically reverses direction.
- The unit for 1 cycle per second is the hertz (Hz). This unit describes the frequency of reversal of voltage polarity and current direction.



## Direct Current and Alternating Current (2)



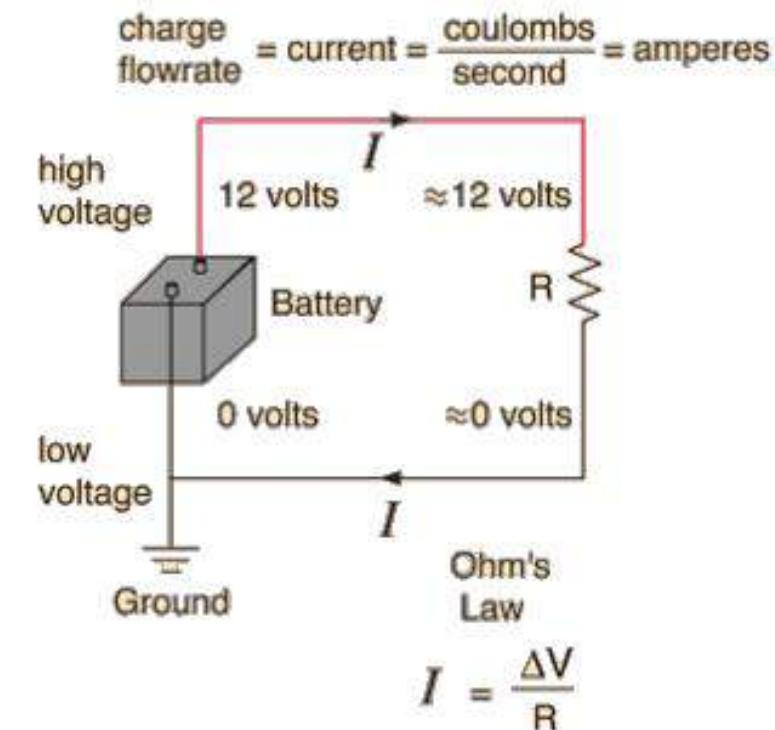
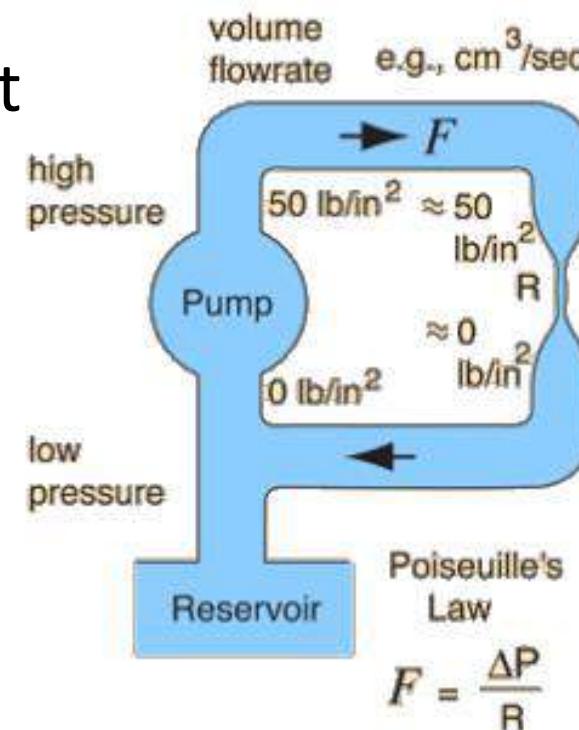
Steady dc voltage of fixed polarity, such as the output of a battery. Note the schematic symbol at left.

Sine-wave ac voltage with alternating polarity, such as from an ac generator. Note the schematic symbol at left. The ac line voltage in your home has this waveform.

# Current (5/5)

## Like a Water Pipe

- Water pressure == voltage
- Water flow == Amps or Current
- Pipe size == Resistance
- Energy created == Watts

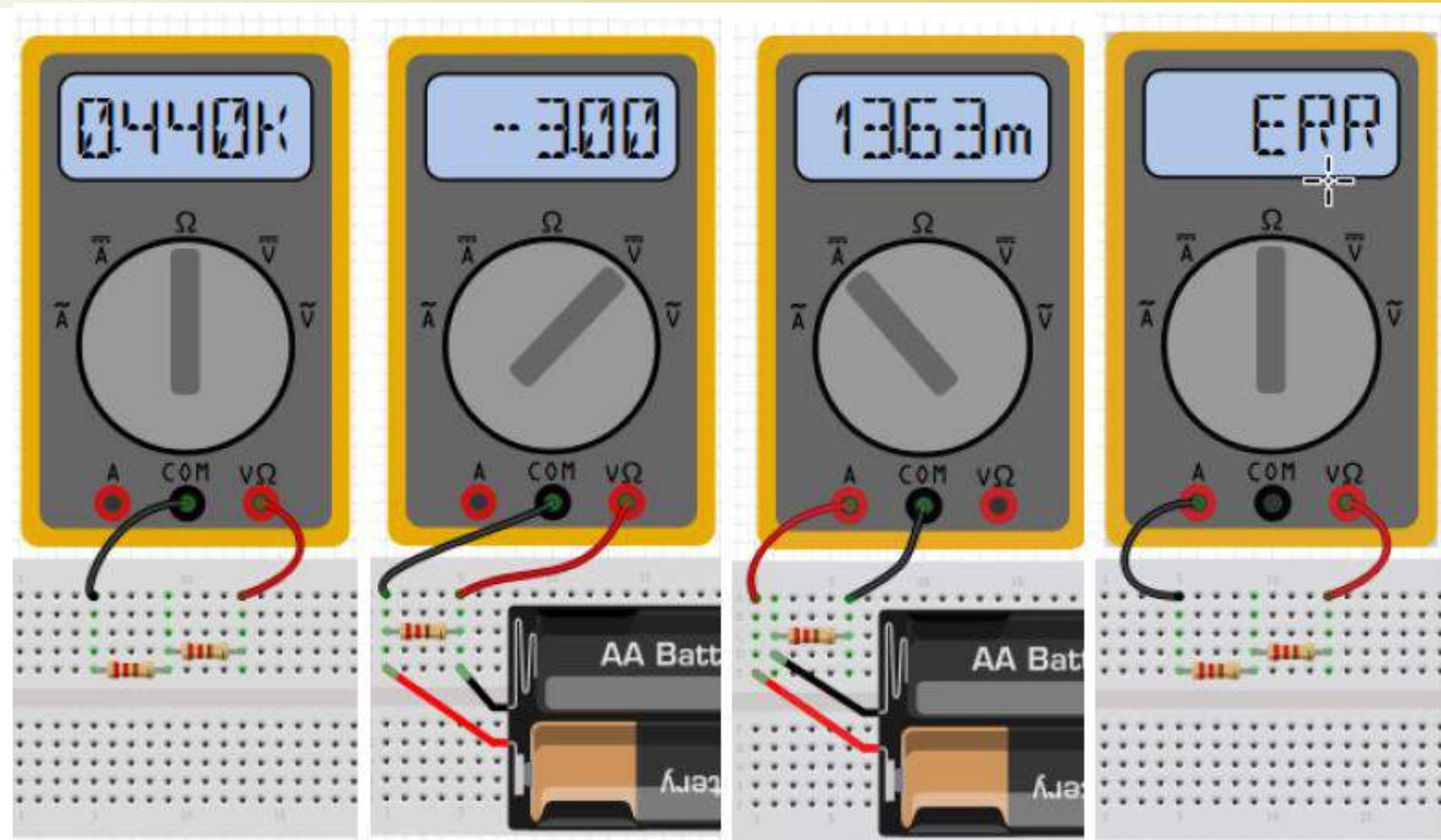




*What will you learn today?*



# Multimeter



(a) Ohmmeter

(b) Voltmeter

(c) Ammeter

(d) Bad wiring



# ELECTRONICS COMPONENTS

[https://makeradvisor.com/tools/?utm\\_source=rnt&utm\\_medium=post&utm\\_campaign=post](https://makeradvisor.com/tools/?utm_source=rnt&utm_medium=post&utm_campaign=post)

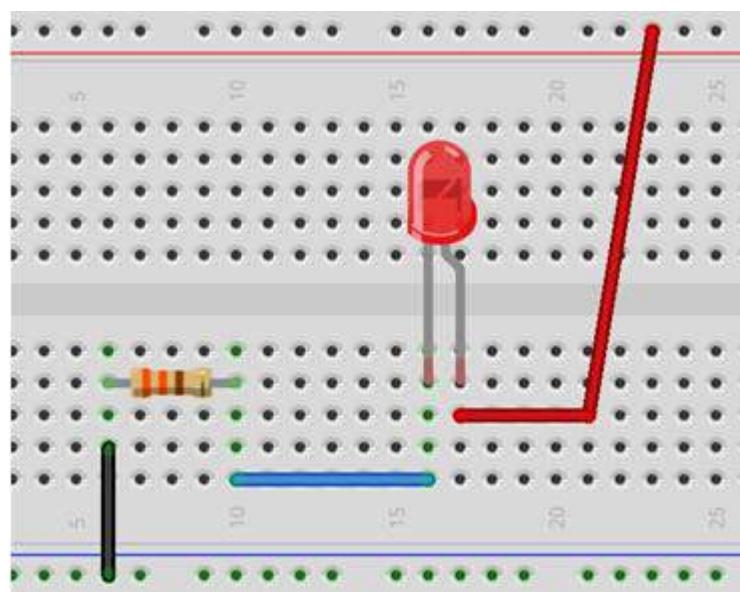
# Components



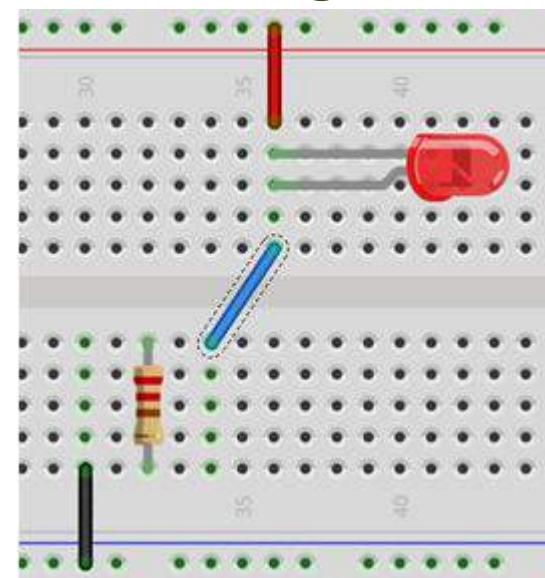
## Breadboard and Wires



Correct



Wrong



<https://www.youtube.com/watch?v=fYInIAmPnGo>

## Multimeter



Analog

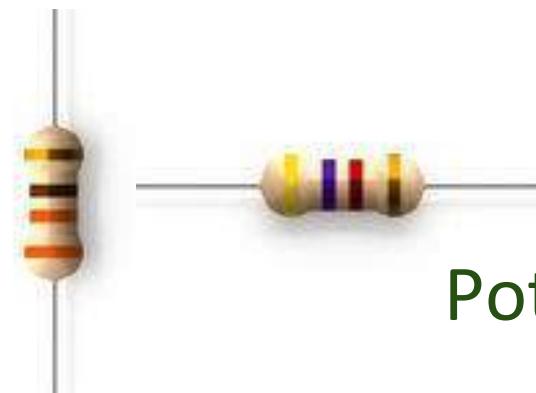


Digital





Resistors



Potentiometer

Capacitors



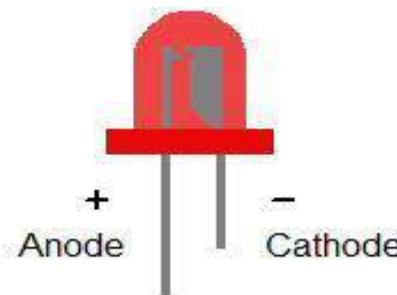
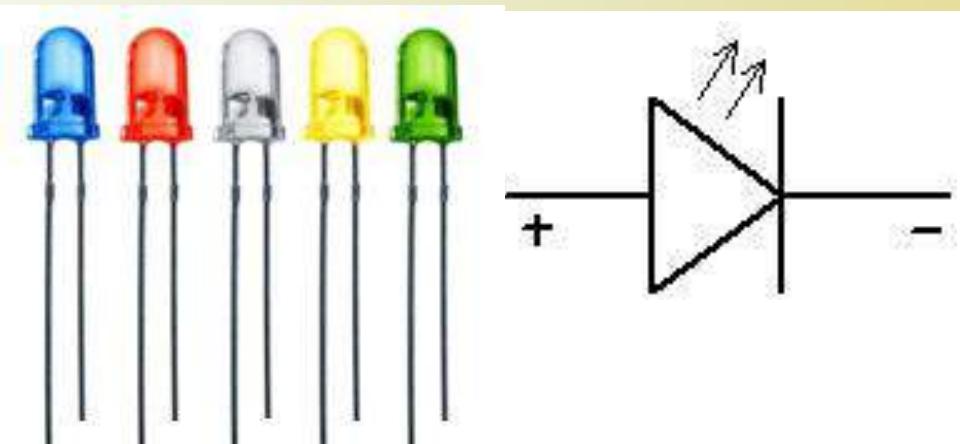
Transistors



Diode

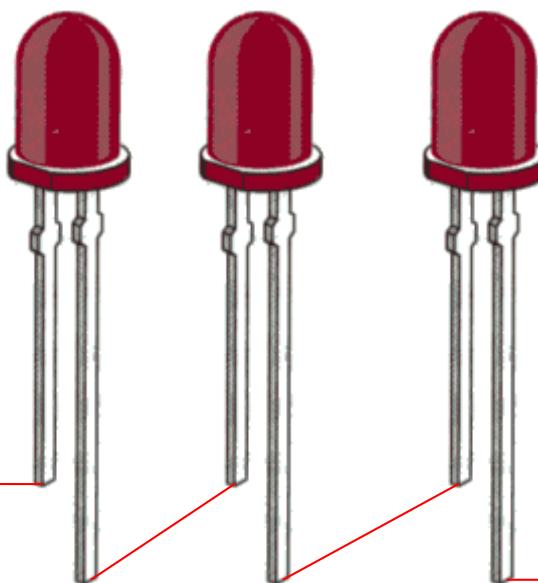


## LED (Light Emitting Diodes)

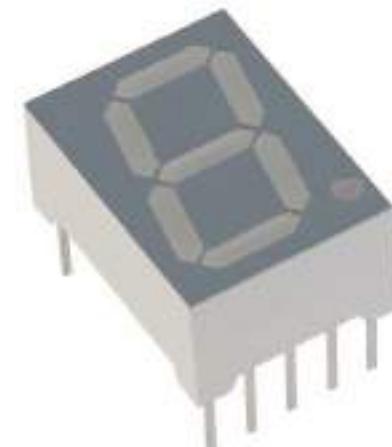


Short leg is negative

Ground Pin



7-segment display

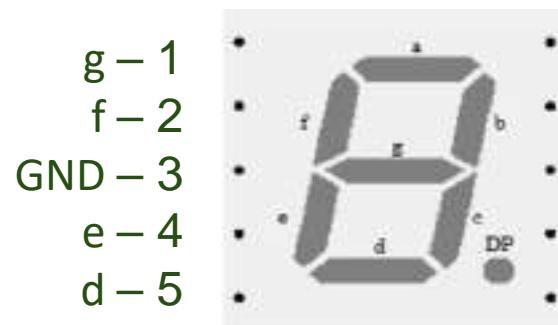
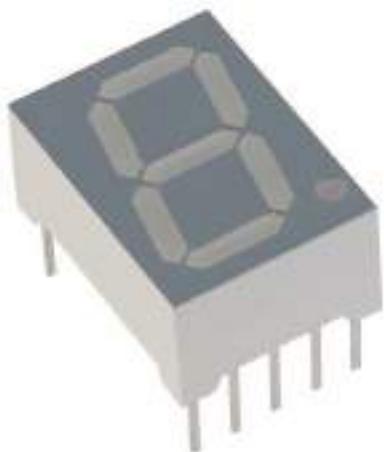


Long leg is positive

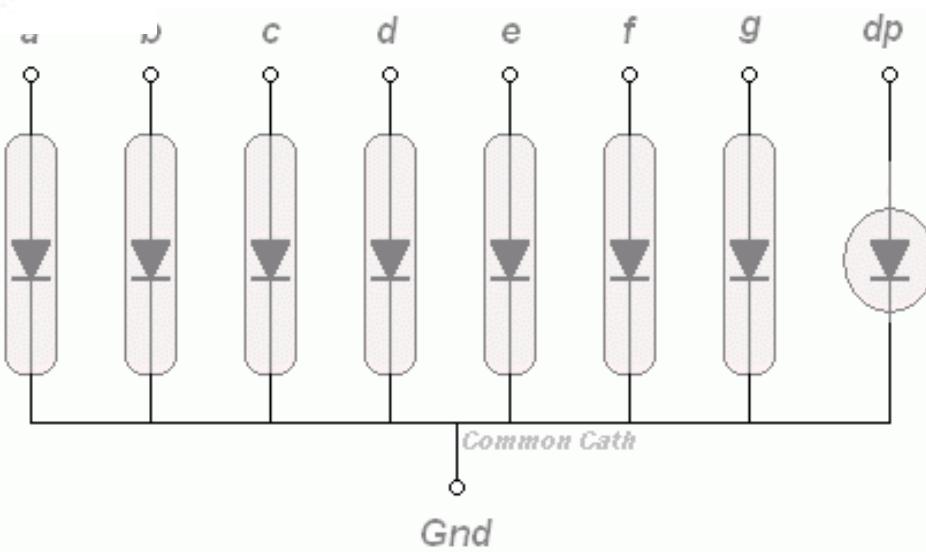
Digital Pin

## LED (Light Emitting Diodes)

7-segment display



10 – a  
9 – b  
8 – GND  
7 – c  
6 – DP



# Components

## Push Buttons and Switches



# Components

## Relays





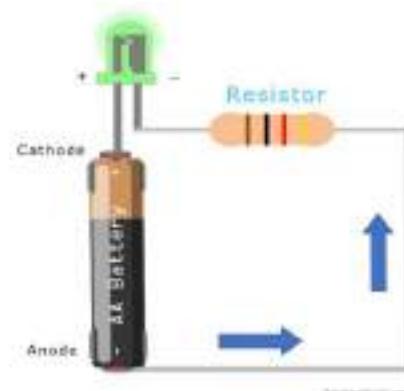
# RESISTORS

## Introduction

- The two main characteristics of a resistor are its resistance,  $R$ , in ohms and its power rating,  $P$ , in Watts.
  - The **resistance,  $R$** , provides the required reduction in current or the desired drop in voltage.
  - Wattage rating:**
    - The **wattage rating** indicates **the amount of power the resistor can safely dissipate as heat**.
    - The **wattage rating** is always more than the actual amount of power dissipated by the resistor, as **a safety factor**.



A resistor does not have a polarity. So you can place it in either side.



## Types of Resistors

- Wire-wound resistors
- Carbon-composition resistors
- Film-type resistors
  - Carbon film
  - Metal film
- Surface-mount resistors (chip resistors)
- Fusible resistors
- Thermistors



## Types of Resistors: Wire Wound Resistor

- Special resistance wire is wrapped around an insulating core, typically porcelain, cement, or pressed paper.
- These resistors are typically used for high-current applications with low resistance and appreciable power.



Large wire-wound resistors with 50-W power ratings.

- (a) Fixed  $R$ , length of 5 in.
- (b) Variable  $R$ , diameter of 3 in.

## Types of Resistors: Carbon Composition Resistors

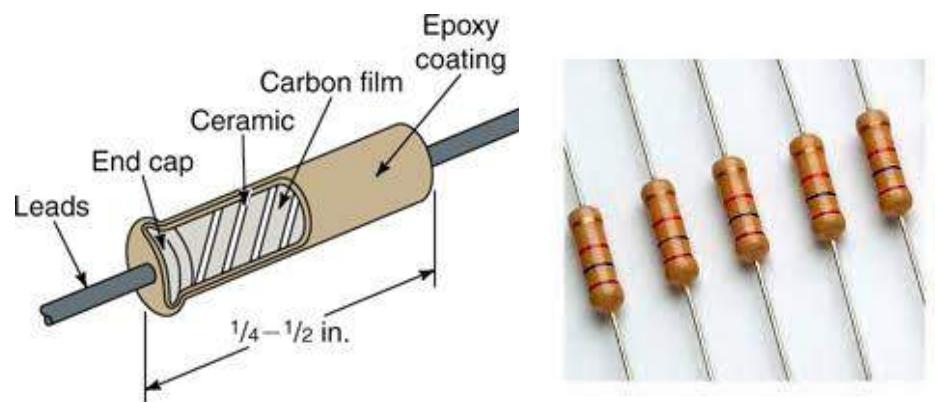
- Made of carbon or graphite mixed with a powdered insulating material.
- Metal caps with tinned copper wire (called axial leads) are joined to the ends of the carbon resistance element. They are used for soldering the connections into a circuit.
- Becoming obsolete because of the development of carbon-film resistors.



Carbon resistors with the same physical size but different resistance values. The physical size indicates a power rating of  $\frac{1}{2}$  W.

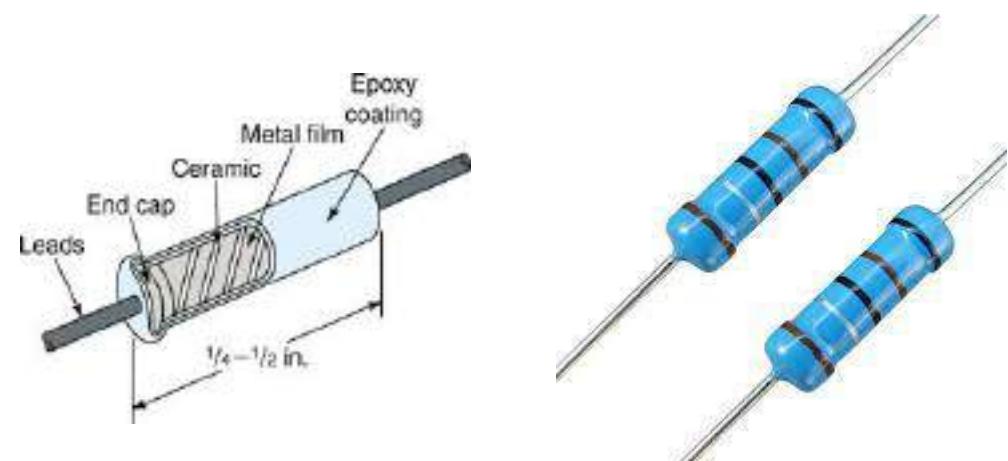
## Type of Resistors: Film Type Resistors

Compared to carbon composition resistors, carbon-film resistors have **tighter tolerances**, are less sensitive to temperature changes and aging, and generate less noise.



Construction of a carbon film resistor.

Metal film resistors have **very tight tolerances**, are **less sensitive to temperature changes** and aging, and generate less noise.



Construction of a metal film resistor.

## Types of Resistors: Surface-Mount Resistors (also called chip resistors)

These resistors are:

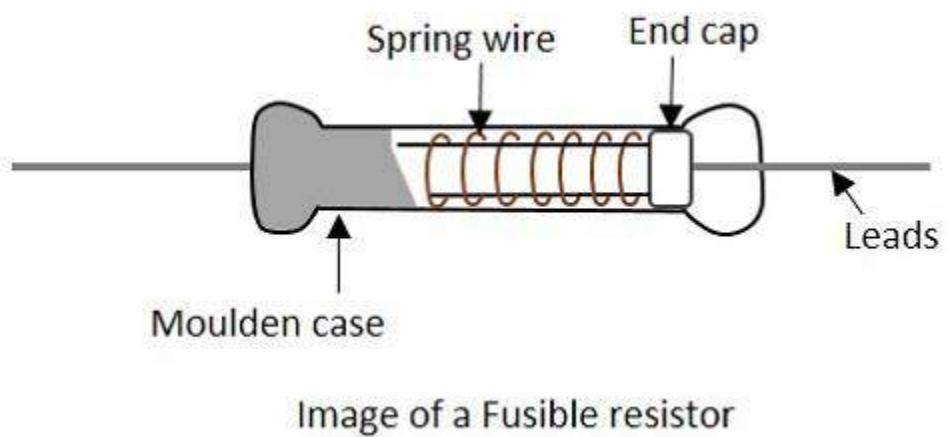
- Temperature-stable and rugged
- Their end electrodes are soldered directly to a circuit board.
- Much smaller than conventional resistors with axial leads.
- Power dissipation rating is usually  $1/8$  to  $\frac{1}{4}$  W



Typical chip resistors.

## Type of Resistors: Fusible Resistors

Fusible resistors are wire-wound resistors made to burn open easily when the power rating is exceeded. They serve a dual function as both a fuse and a resistor.



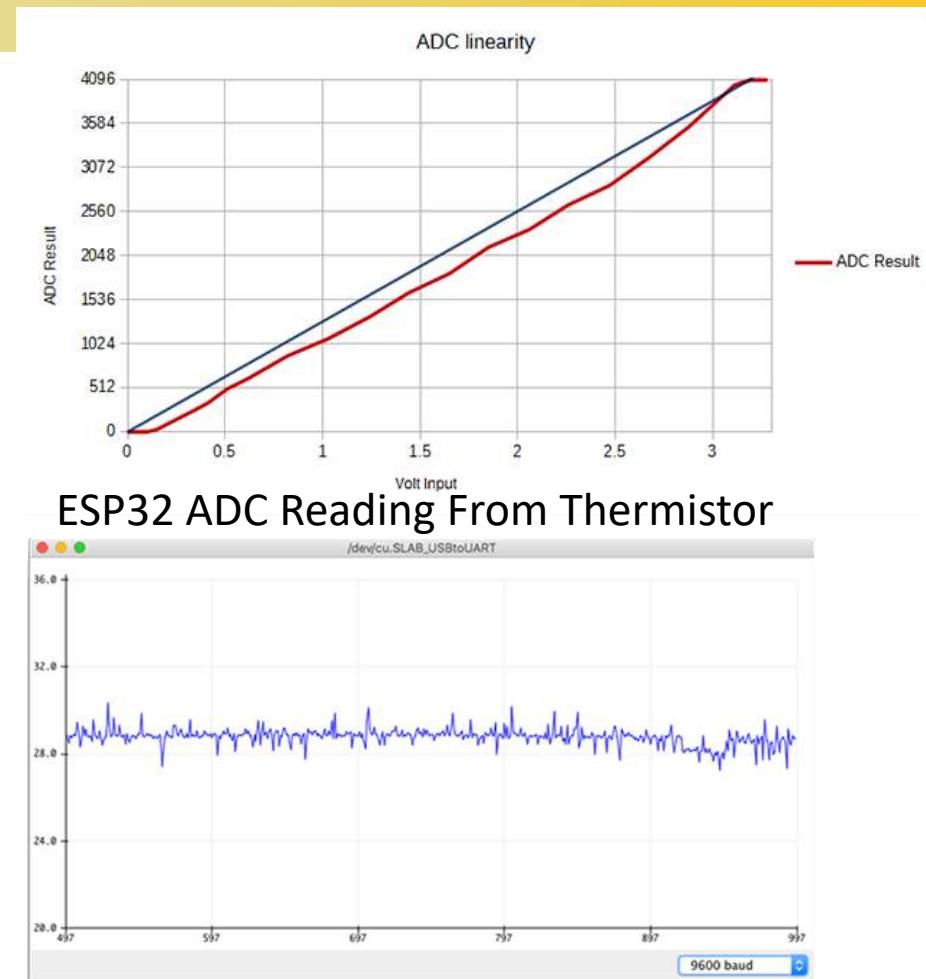


## Type of Resistors: Thermistors

- Thermistors are temperature-sensitive resistors whose resistance value changes with changes in operating temperature.
- Used in electronic circuits where **temperature measurement**, control, and compensation are desired.



Typical thermistor shapes and sizes.



<https://www.e-tinkers.com/2019/10/using-a-thermistor-with-arduino-and-unexpected-esp32-adc-non-linearity/>

# Resistor Color Coding



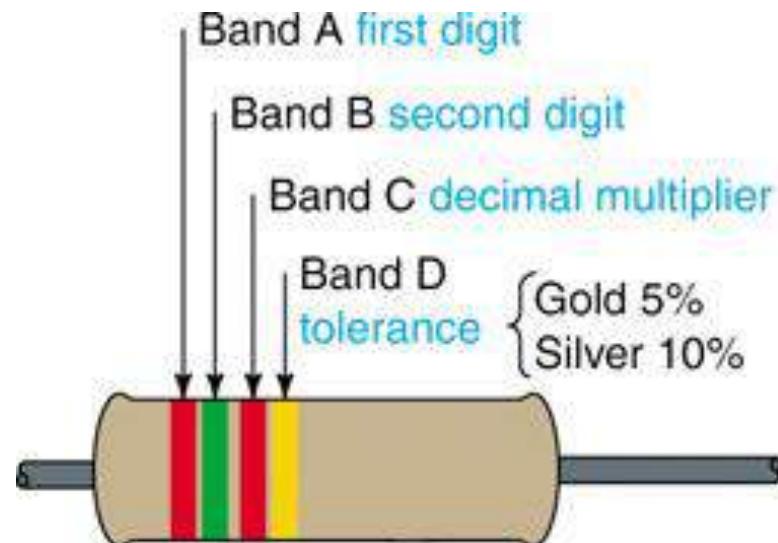
- Carbon resistors are small, so their  $R$  value in ohms is marked using a color-coding system.
- Colors represent numerical values.
- Coding is standardized by the [Electronic Industries Alliance \(EIA\)](#).

# Resistor Color Coding



## Resistor Color Code

Color Code	
0	Black
1	Brown
2	Red
3	Orange
4	Yellow
5	Green
6	Blue
7	Violet
8	Gray
9	White

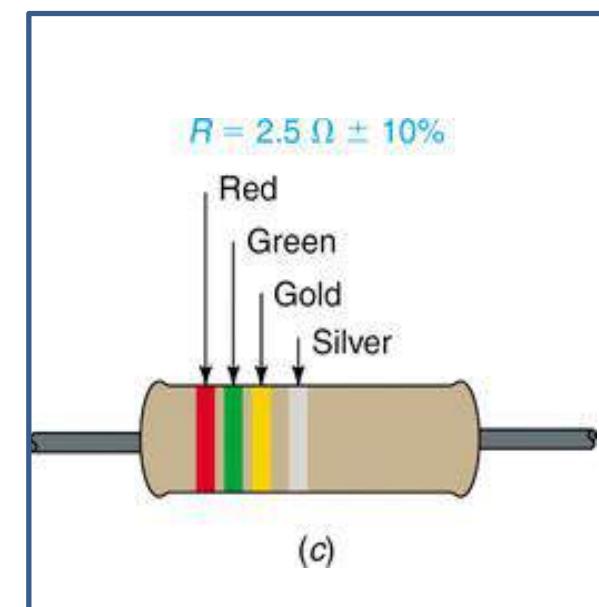
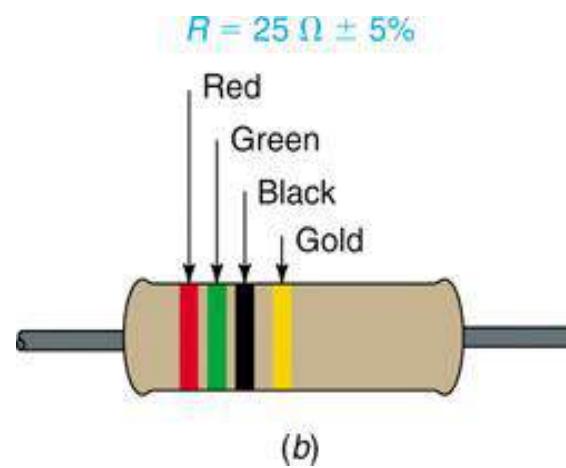
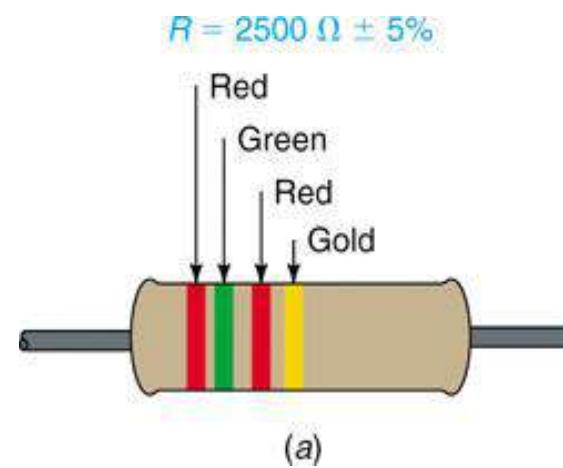


How to read color stripes on carbon resistors for  $R$  in ohms.

# Resistor Color Coding



- Resistors under  $10 \Omega$ :
  - The multiplier band is either gold or silver at the 3<sup>rd</sup> band.
    - For gold, multiply by 0.1.
    - For silver, multiply by 0.01.



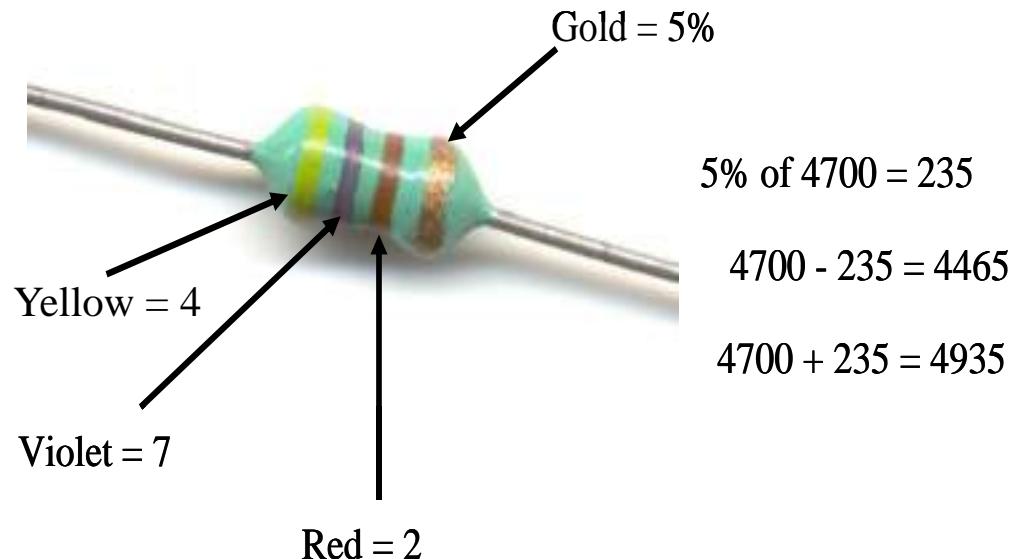
Examples of color-coded  $R$  values, with percent tolerance.

# Resistor Color Coding



## Tolerance

- Applying the Color Code
  - The amount by which the actual  $R$  can differ from the color-coded value is its **tolerance**. Tolerance is usually stated in percentages.



The **actual value** can range from 4465 to 4935  $\Omega$ .

**4700 $\Omega$**  is the **nominal** value.

## Tolerance Examples

- What is the nominal value and permissible ohmic range for each resistor shown?



$1 \text{ k}\Omega$  (950 to 1050  $\Omega$ )



$390 \Omega$  (370.5 to 409.5  $\Omega$ )



$22 \text{ k}\Omega$  (20.9 to 23.1  $\text{k}\Omega$ )

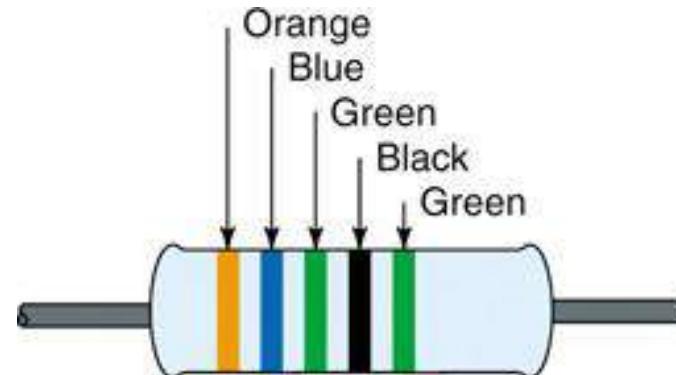


$1 \text{ M}\Omega$  (950  $\text{k}\Omega$  to 1.05  $\text{M}\Omega$ )



## Five-Band Color Code

- Precision resistors often use a five-band code to obtain more accurate  $R$  values.
- The first three stripes indicate the first 3 digits in the  $R$  value.
- The fourth stripe is the multiplier.
- The tolerance is given by the fifth stripe.
  - Brown = 1%
  - Red = 2%
  - Green = 0.5%
  - Blue = 0.25%
  - Violet = 0.1%.



Five-band code.

# Exercise:



- Using the **five-band code**, indicate the colors of the bands for each of the following resistors:

- $110 \Omega \pm 1\%$
- $34 \text{ k}\Omega \pm 0.5\%$
- $82.5 \text{ k}\Omega \pm 2\%$

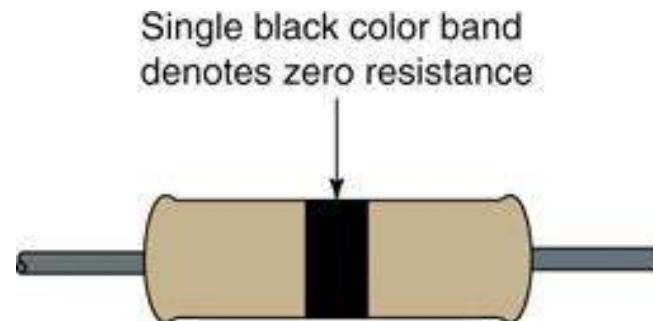
**Tolerance**  
Brown = 1%  
Red = 2%  
Green = 0.5%  
Blue = 0.25%  
Violet = 0.1%.

Color Code	
0	Black
1	Brown
2	Red
3	Orange
4	Yellow
5	Green
6	Blue
7	Violet
8	Gray
9	White



## Zero-Ohm Resistor

- Has zero ohms of resistance.
- Used for connecting two points on a printed-circuit board.
- Body has a single black band around it.
- Wattage ratings are typically 1/8- or 1/4-watt.



A zero-ohm resistor is indicated by a single black color band around the body of the resistor.

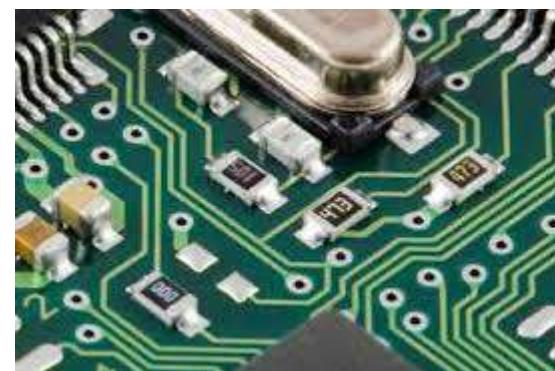
# Chip Resistor Coding System



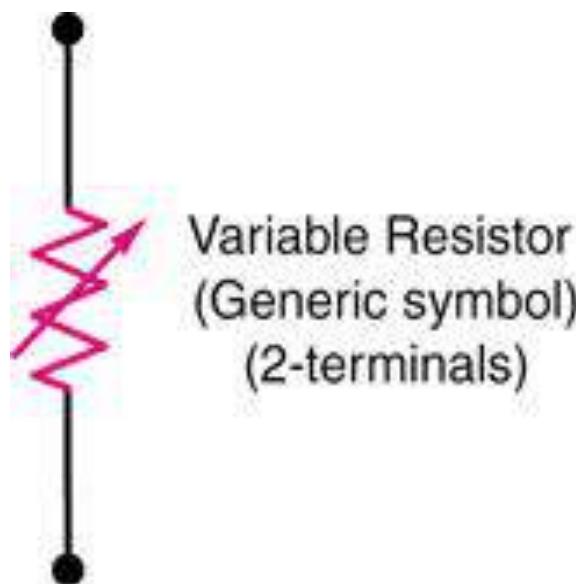
- Body color is usually white or off-white
- End terminals are C-shaped
- Three (four) digits on the body or on the film
  - First 2 (3) digits indicate the first two (three) numbers
  - Third (fourth) digit indicates the multiplier
- Are available in tolerances of  $\pm 1\%$   $\pm 5\%$  but tolerances are not indicated on the chip
- The letter R is used to signify a decimal point for values between 1 to 10 ohms (1R5 means 1.5 ohms)



<b>223</b>	$223 = 22 \times 10^3 = 22,000 \text{ Ohm} = 22 \text{ KOhm}$	Three-Digit Resistor
<b>8202</b>	$8202 = 820 \times 10^2 \text{ Ohm} = 82,000 \text{ Ohm} = 82 \text{ KOhm}$	Four-Digit Resistor
<b>4R7</b>	$4R7 = 4.7 \text{ Ohm}$	Resistor With Radix Point
<b>0R22</b>	$0R22 = 0.22 \text{ Ohm}$	Resistor With Radix Point
<b>0</b>	$0 = 0 \text{ Ohm}$	Zero-Ohm Resistor
<b>000</b>	$000 = 0 \text{ Ohm}$	Fractional-Digit Resistor



- A variable resistor is a resistor whose resistance value can be changed.





## Decade resistance box

- Provides any R within a wide range of values
  - First dial is the units or  $R \times 1$  dial.
  - Second dial is the tens or  $R \times 10$  dial
  - The hundreds or  $R \times 100$  dial has an R of 0 to  $900\Omega$
  - etc.
- Dials are connected internally so that their values add to one another.
- Rheostats and potentiometers are variable resistances used to vary the amount of current or voltage in a circuit.

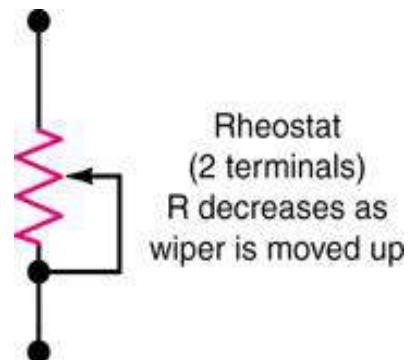
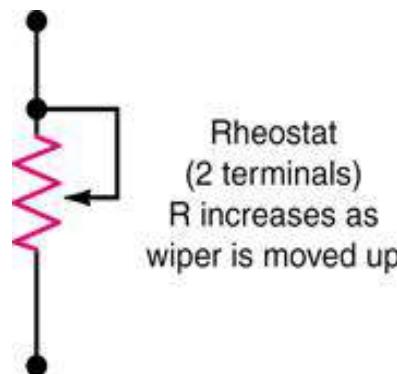


# Rheostats and Potentiometers

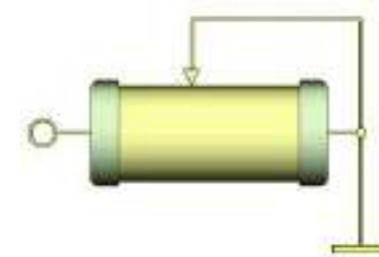
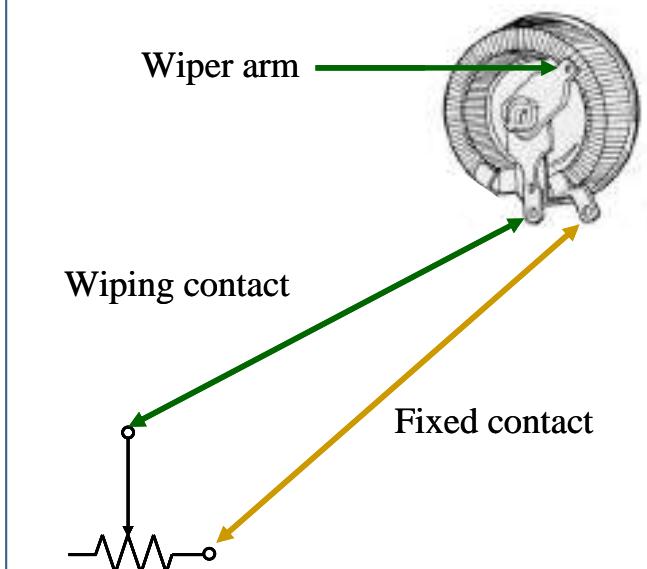


- **Rheostats:**

- ✓ Two terminals.
- ✓ Connected in series with the load and the voltage source.
- ✓ Varies the current.

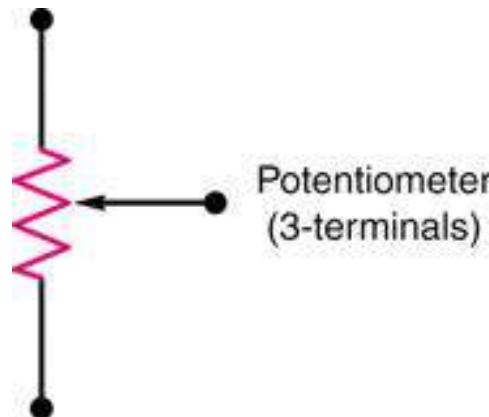


Rheostats are two-terminal devices.



## Potentiometers:

- Three terminals.
- Ends connected across the voltage source.
- Third variable arm taps off part of the voltage.

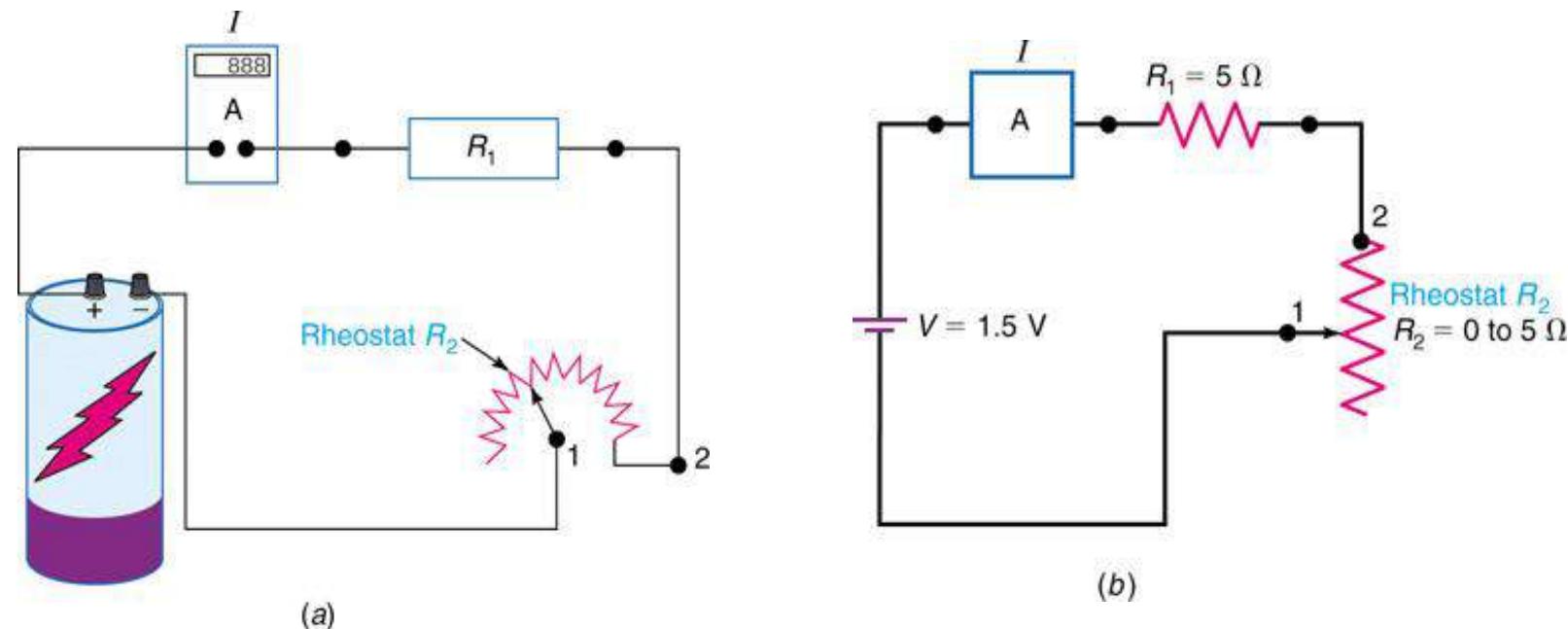


# Rheostats and Potentiometers



## Rheostat

- Using for Controlling Current Flow
- The rheostat must have a wattage rating high enough for the maximum  $I$  when  $R$  is minimum.



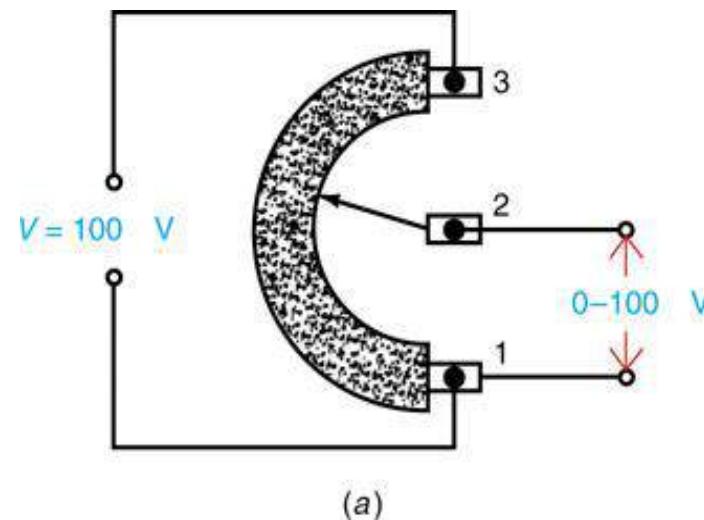
Rheostat connected in series circuit to vary the current  $I$ . Symbol for the current meter is A, for amperes. (a) Wiring diagram with digital meter for  $I$ . (b) Schematic diagram.

# Rheostats and Potentiometers

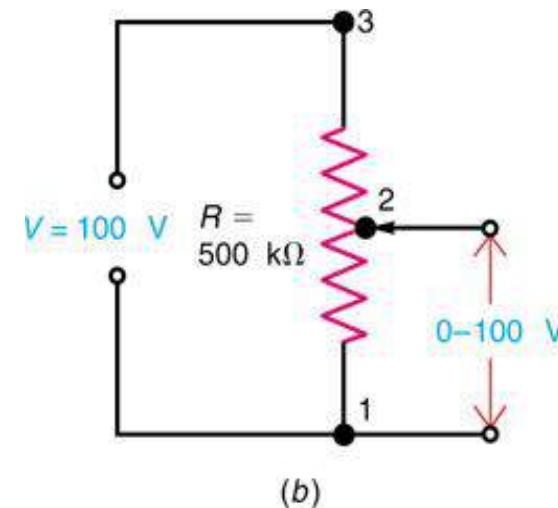


## Potentiometers

- Potentiometers are three-terminal devices.
- The applied  $V$  is input to the two end terminals of the potentiometer.
- The variable  $V$  is output between the variable arm and an end terminal.



(a)



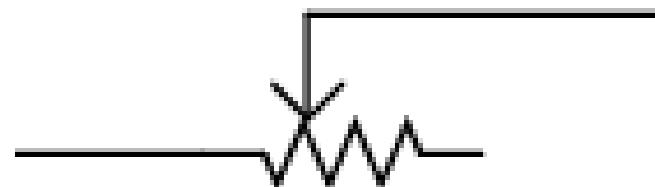
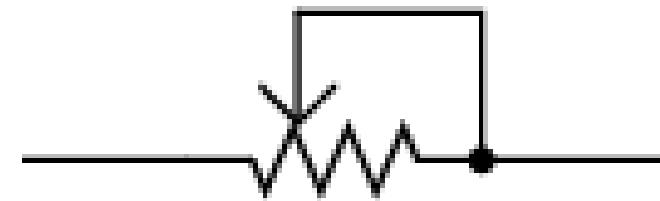
(b)

Potentiometer connected across voltage source to function as a voltage divider.  
(a) Wiring diagram. (b) Schematic diagram.



# Potentiometer Used as a Rheostat

- A potentiometer may be used as a rheostat by simply using the wiper terminal and one of the other terminals, the third terminal is left unconnected and unused
- Another method is to wire the unused terminal to the center terminal



# Power Rating of Resistors



- In addition to having the required ohms value, a resistor should have a wattage rating high enough to dissipate the power produced by the current without becoming too hot.
- Power rating depends on the resistor's construction.
- A larger physical size indicates a higher power rating.
- Higher-wattage resistors can operate at higher temperatures.
- Wire-wound resistors are physically larger and have higher power ratings than carbon resistors.
- Maximum allowable current for any resistance setting is calculated as:  $I_{max}$
- Maximum voltage which produces the rated power dissipation can be calculated as:  $V_{max}$

$$I_{max} = \sqrt{\frac{P}{R}}$$

$$V_{max} = \sqrt{P \cdot R}$$

Exercise:

**P** and **R** are the rated value of rheostat

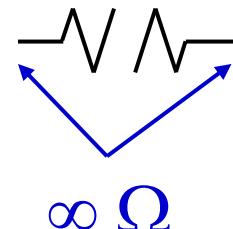
Q. What is  $I_{max}$  of a 5-KΩ 2-W rheostat?

# Resistor Troubles



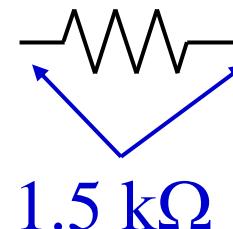
- Resistors can become open or they can drift **out of tolerance**.
- Some controls (especially volume and tone controls) may become noisy or scratchy-sounding, indicating a dirty or worn-out resistance element.
- Due to the very nature of their construction, resistors can short out internally. They may, however, become short-circuited by another component in the circuit.

An example of an out-of-tolerance resistor:



An open resistor measures **infinite resistance**.

1 k $\Omega$ , 5% nominal



- Resistance measurements are made with an **ohmmeter**.
- The ohmmeter has its **own voltage source**, so **voltage must be off in the circuit being tested**. Otherwise the ohmmeter may become damaged.

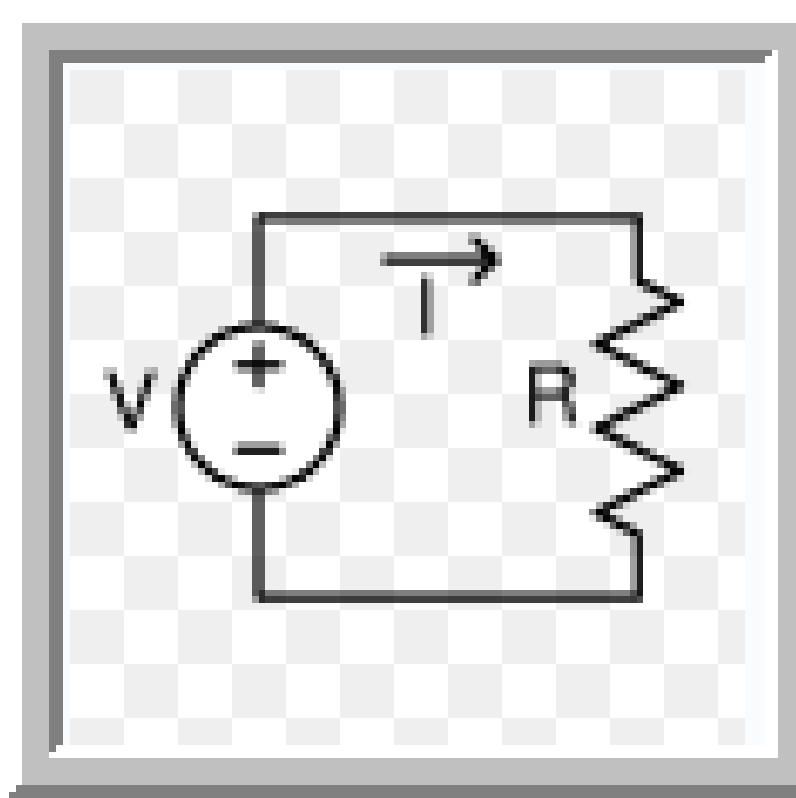
- All experienced technicians have seen a **burnt resistor**.
- This is usually caused by a short somewhere else in the circuit which causes a high current to flow in the resistor.
- When a resistor's power rating is exceeded, it can burn open or drift way out of tolerance.





# OHM'S LAW

Ohm's law states that, in an electrical circuit, the current passing through most materials is directly proportional to the potential difference applied across them.

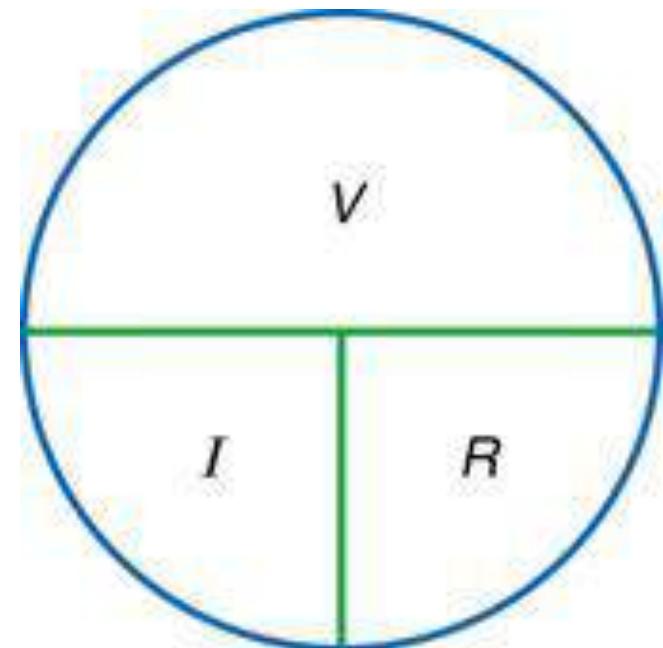


**There are three forms of Ohm's Law:**

- $I = V/R$
- $V = IR$
- $R = V/I$

**where:**

- $I$  = Current
- $V$  = Voltage
- $R$  = Resistance

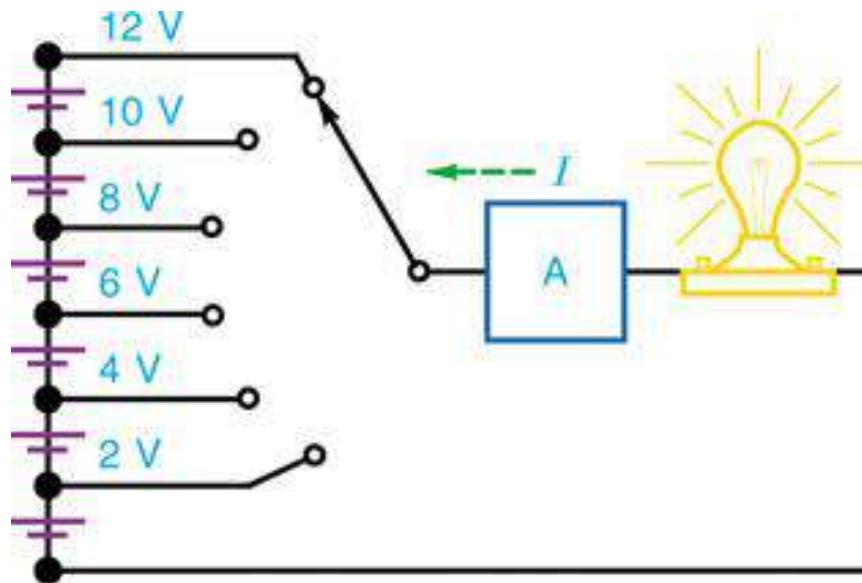


A circle diagram to help in memorizing the Ohm's Law formulas  $V = IR$ ,  $I = V/R$ , and  $R = V/I$ .  
The  $V$  is always at the top.

# The Current $I = V/R$



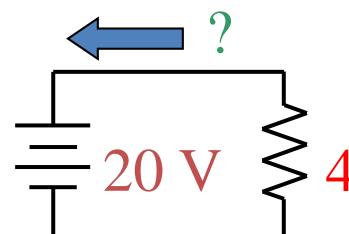
- $I = V/R$
- In practical units, this law may be stated as:
  - amperes = volts / ohms

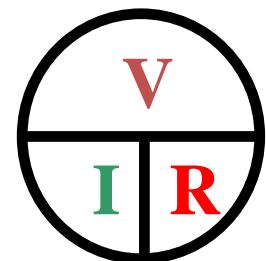


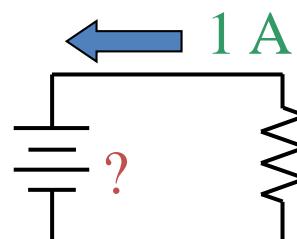
Increasing the applied voltage  $V$  produces more current  $I$  to light the bulb with more intensity.

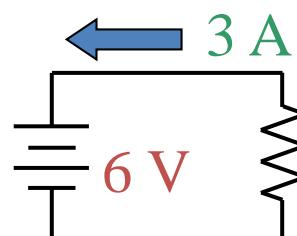
- The **three** forms of Ohm's law can be used to define the practical units of current, voltage, and resistance:
  - $1 \text{ ampere} = 1 \text{ volt} / 1 \text{ ohm}$
  - $1 \text{ volt} = 1 \text{ ampere} \times 1 \text{ ohm}$
  - $1 \text{ ohm} = 1 \text{ volt} / 1 \text{ ampere}$

## Applying Ohm's Law


$$20 \text{ V} \quad 4 \Omega \quad I = \frac{20 \text{ V}}{4 \Omega} = 5 \text{ A}$$



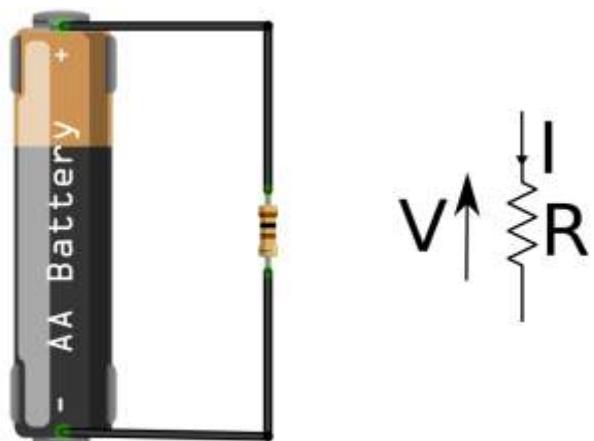

$$? \quad 12 \Omega \quad V = 1 \text{ A} \times 12 \Omega = 12 \text{ V}$$


$$6 \text{ V} \quad ? \quad R = \frac{6 \text{ V}}{3 \text{ A}} = 2 \Omega$$

# Exercise

- Solve for the resistance,  $R$ , when  $V$  and  $I$  are known

- a.  $V = 14 \text{ V}$ ,  $I = 2 \text{ A}$ ,  $R = ?$
- b.  $I = 5 \text{ A}$ ,  $R = 5$ ,  $V = ?$
- c.  $V = 6 \text{ V}$ ,  $R = 4$ ,  $I = ?$



## Voltage

- **Units of Voltage**

- The basic unit of voltage is the **volt (V)**.

- Multiple units of voltage are:

- **kilovolt (kV)**

- 1 thousand volts or  $10^3$  V

- **megavolt (MV)**

- 1 million volts or  $10^6$  V

- Submultiple units of voltage are:

- **millivolt (mV)**

- 1-thousandth of a volt or  $10^{-3}$  V

- **microvolt ( $\mu$ V)**

- 1-millionth of a volt or  $10^{-6}$  V

## Current

- **Units of Current**

- The basic unit of current is the **ampere (A)**.
- Submultiple units of current are:

- **milliampere (mA)**

- 1-thousandth of an ampere or  $10^{-3}$  A

- **microampere ( $\mu$ A)**

- 1-millionth of an ampere or  $10^{-6}$  A

## Resistance

- **Units of Resistance**

- The basic unit of resistance is the **Ohm ( $\Omega$ )**.
- Multiple units of resistance are:

- **kilohm ( $k\Omega$ )**

- 1 thousand ohms or  $10^3 \Omega$

- **Megohm ( $M\Omega$ )**

- 1 million ohms or  $10^6 \Omega$

# Exercise

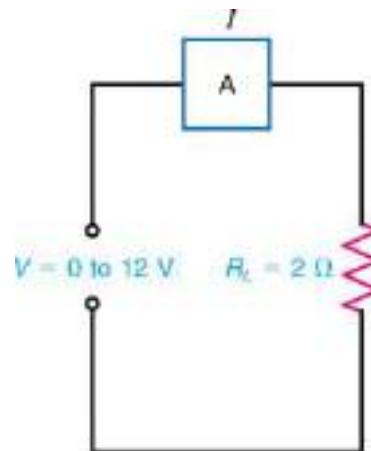


- How much is the current,  $I$ , in a  $470\text{-k}\Omega$  resistor if its voltage is  $23.5\text{ V}$ ?
- How much voltage will be dropped across a  $40\text{ k}\Omega$  resistance whose current is  $250\text{ }\mu\text{A}$ ?

# The Linear Proportion between $V$ and $I$ (1/4)



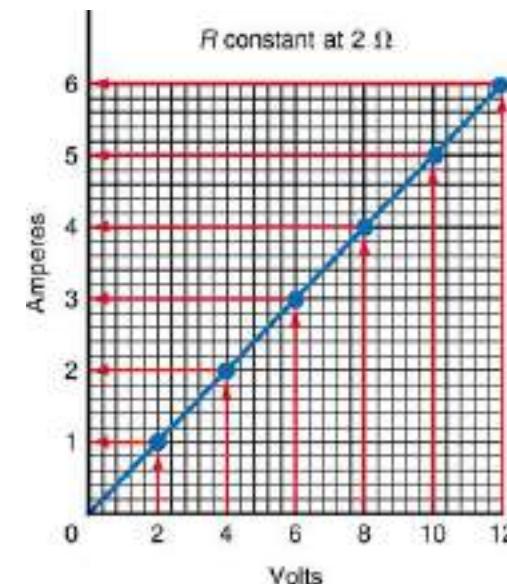
- The Ohm's Law formula  $I = V/R$  states that  $V$  and  $I$  are directly proportional for any one value of  $R$ .



(a)

Volts $V$	Ohms $\Omega$	Amperes $A$
0	2	0
2	2	1
4	2	2
6	2	3
8	2	4
10	2	5
12	2	6

(b)



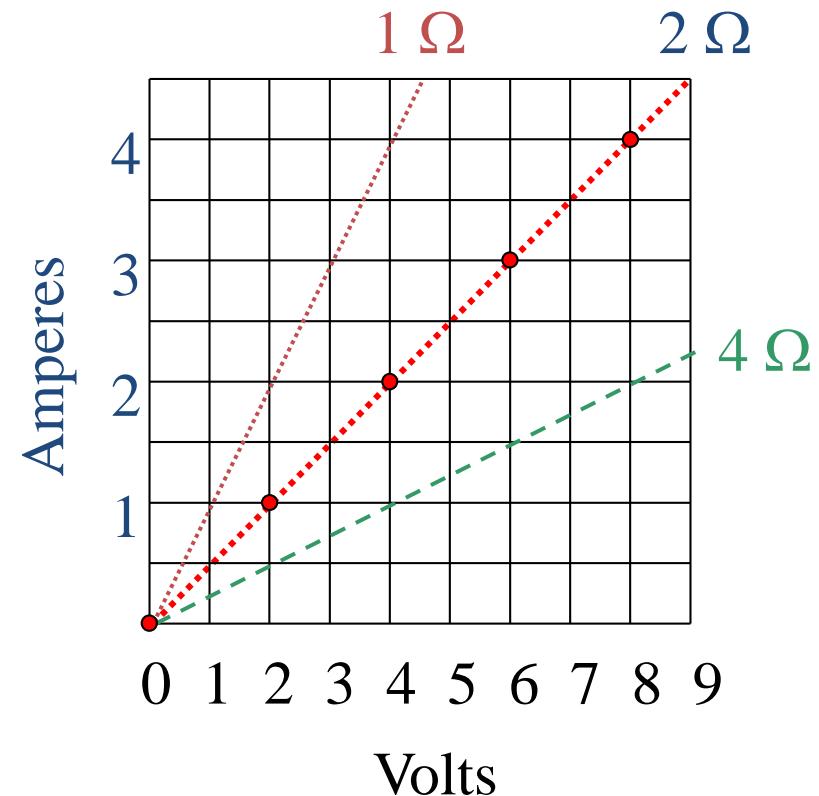
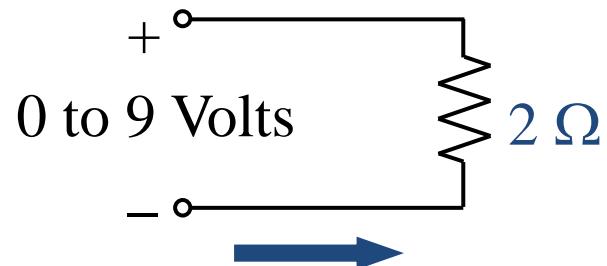
(c)

Experiment to show that  $I$  increases in direct proportion to  $V$  with the same  $R$ . (a) Circuit with variable  $V$  but constant  $R$ . (b) Table of increasing  $I$  for higher  $V$ . (c) Graph of  $V$  and  $I$  values. This is a linear volt-ampere characteristic. It shows a direct proportion between  $V$  and  $I$ .

- When  $V$  is constant:
  - $I$  decreases as  $R$  increases.
  - $I$  increases as  $R$  decreases.
- Examples:
  - If  $R$  doubles,  $I$  is reduced by half.
  - If  $R$  is reduced to  $\frac{1}{4}$ ,  $I$  increases by 4.
  - This is known as an *inverse relationship*.

- Linear Resistance
  - A linear resistance has a constant value of ohms. Its  $R$  does not change with the applied voltage, so  $V$  and  $I$  are directly proportional.
  - Carbon-film and metal-film resistors are examples of linear resistors.

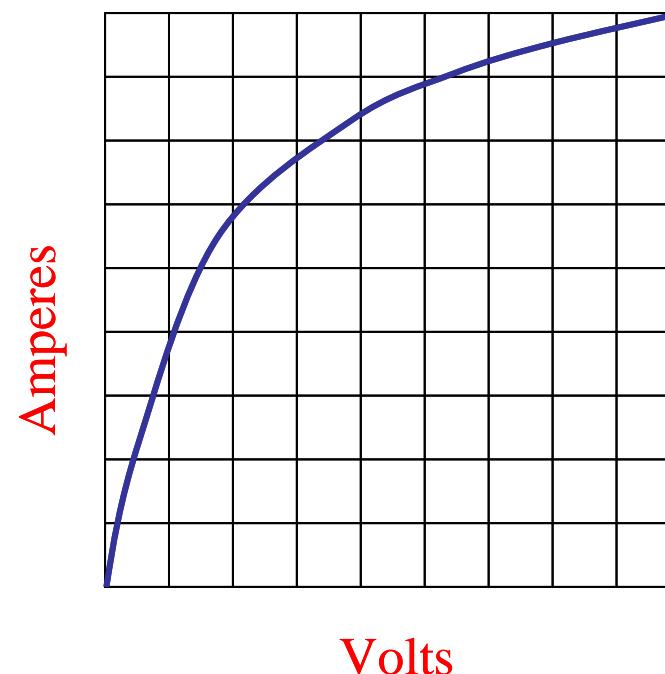
# The Linear Proportion between $V$ and $I$ (4/4)



The smaller the resistor, the steeper the slope.

## Nonlinear Resistance

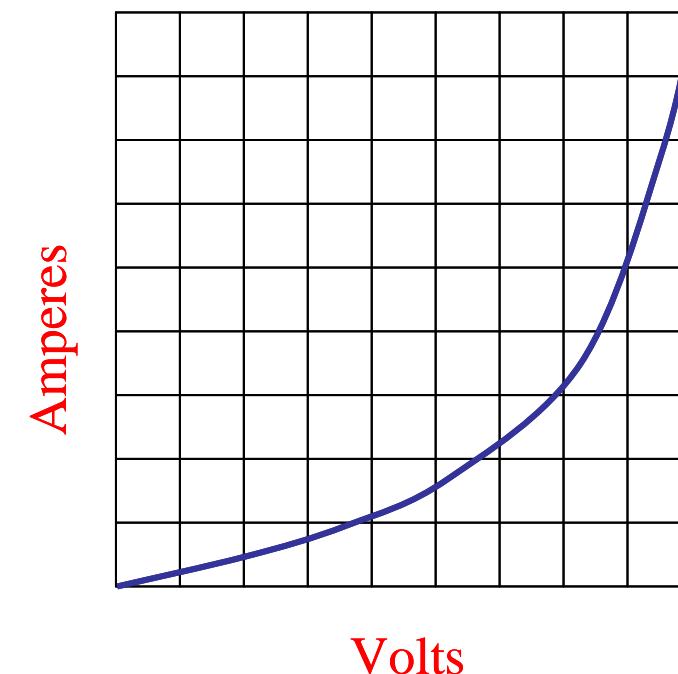
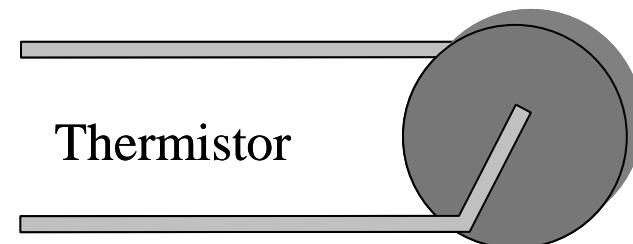
- In a nonlinear resistance, increasing the applied  $V$  produces more current, **but  $I$  does not increase in the same proportion as the increase in  $V$ .**
- Example of a Nonlinear Volt–Ampere Relationship:
  - As the tungsten filament in a light bulb gets hot, its resistance increases.



# The Nonlinear Proportion between $V$ and $I$ (2/2)



- ✓ Another nonlinear resistance is a thermistor.
- ✓ A thermistor is a resistor whose resistance value changes with its operating temperature.
- ✓ As an NTC (negative temperature coefficient) thermistor gets hot, its resistance decreases.





- **The basic unit of power is the watt (W).**

- Multiple units of power are:
  - **kilowatt (kW):**  
1000 watts or  $10^3$  W
  - **megawatt (MW):**  
1 million watts or  $10^6$  W
- Submultiple units of power are:
  - **milliwatt (mW):**  
1-thousandth of a watt or  $10^{-3}$  W
  - **microwatt ( $\mu$ W):**  
1-millionth of a watt or  $10^{-6}$  W

- Work and energy are basically the same, with identical units.
- Power is different. It is the time rate of doing work.
  - Power = work / time.
  - Work = power × time.



- Practical Units of Power and Work:

- The rate at which work is done (power) equals the product of voltage and current. This is derived as follows:
- First, recall that:

$$1 \text{ volt} = \frac{1 \text{ joule}}{1 \text{ coulomb}} \quad \text{and} \quad 1 \text{ ampere} = \frac{1 \text{ coulomb}}{1 \text{ second}}$$



Power = Volts × Amps, or

$$P = V \times I$$

$$\text{Power (1 watt)} = \frac{1 \text{ joule}}{1 \text{ coulomb}} \times \frac{1 \text{ coulomb}}{1 \text{ second}} = \frac{1 \text{ joule}}{1 \text{ second}}$$

- **Kilowatt Hours**
  - The kilowatt hour (kWh) is a unit commonly used for large amounts of electrical work or energy.
  - For example, **electric bills** are calculated in kilowatt hours. The kilowatt hour is the billing unit.
  - The **amount of work (energy)** can be found by multiplying power (in kilowatts)  $\times$  time in hours.



To calculate electric cost, start with the power:

- An air conditioner operates at 240 volts and 20 amperes.
- The power is  $P = V \times I = 240 \times 20 = 4800$  watts.
  - Convert to kilowatts: 4800 watts = 4.8 kilowatts
  - Multiply by hours: (Assume it runs half the day)  
$$\text{energy} = 4.8 \text{ kW} \times 12 \text{ hours} = 57.6 \text{ kWh}$$
  - Multiply by rate: (Assume a rate of \$0.08/ kWh)  
$$\text{cost} = 57.6 \times \$0.08 = \$4.61 \text{ per day}$$

# Exercise



- How much is the output voltage of a power supply if it supplies 75 W of power while delivering a current of 5 A?
- How much does it cost to light a 300-W light bulb for 30 days if the cost of the electricity is 7 ¢/kWh.

# Power Dissipation in Resistance



- When current flows in a resistance, heat is produced from the friction between the moving free electrons and the atoms obstructing their path.
- Heat is evidence that power is used in producing current.
- The amount of power dissipated in a resistance may be calculated using any one of three formulas, depending on which factors are known:
  - ✓  $P = I^2 \times R$
  - ✓  $P = V^2 / R$
  - ✓  $P = V \times I$

# Exercise



- Solve for the power,  $P$ , dissipated by the resistance,  $R$ 
  - a.  $I = 1 \text{ A}$ ,  $R = 100\Omega$ ,  $P = ?$
  - b.  $I = 20 \text{ mA}$ ,  $R = 1 \text{ k}\Omega$ ,  $P = ?$
  - c.  $V = 5 \text{ V}$ ,  $R = 150\Omega$ ,  $P = ?$
  - d.  $V = 22.36 \text{ V}$ ,  $R = 1 \text{ k}\Omega$ ,  $P = ?$
- How much power is dissipated by an  $8\text{-}\Omega$  load if the current in the load is  $200 \text{ mA}$ ?

# Power Formulas (1)



There are three basic power formulas, but each can be in three forms for nine combinations.

$$P = VI$$

$$I = \frac{P}{V}$$

$$V = \frac{P}{I}$$

$$P = I^2R$$

$$R = \frac{P}{I^2}$$

$$I = \sqrt{\frac{P}{R}}$$

$$P = \frac{V^2}{R}$$

$$R = \frac{V^2}{P}$$

$$V = \sqrt{PR}$$

Where:

P = Power

V = Voltage

I = Current

R=Resistance

- Combining Ohm's Law and the Power Formula
  - All nine power formulas are based on Ohm's Law.

$$V = IR$$

$$I = \frac{V}{R}$$

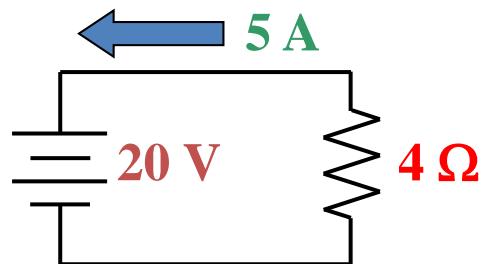
$$P = VI$$

- Substitute  $IR$  for  $V$  to obtain:

$$P = VI = (IR)I = I^2R$$

# Example

- Applying Power Formulas:



$$P = VI = 20 \times 5 = 100 \text{ W}$$

$$P = I^2R = 25 \times 4 = 100 \text{ W}$$

$$P = \frac{V^2}{R} = \frac{400}{4} = 100 \text{ W}$$

# Exercise



- What is the resistance of a device that dissipates 1.2 kW of power when its current is 10 A?
- How much current does a 960 W coffeemaker draw from the 120 V power line?
- What is the resistance of a 20 W, 12 V halogen lamp?

- Follow these steps when choosing a resistor for a circuit:
  - Determine the required resistance value as  $R = V / I$ .
  - Calculate the power dissipated by the resistor using any of the power formulas.
  - Select a wattage rating for the resistor that will provide an adequate cushion between the actual power dissipation and the resistor's power rating.
  - Ideally, **the power dissipation in a resistor should never be more than 50% of its power rating.**

- Maximum Working Voltage Rating
  - A resistor's maximum working voltage rating is the maximum voltage a resistor can withstand without internal arcing.
  - The higher the wattage rating of the resistor, the higher the maximum working voltage rating.

- Maximum Working Voltage Rating
  - With very large resistance values, the maximum working voltage rating may be exceeded before the power rating is exceeded.
  - For any resistor, the maximum voltage which produces the rated power dissipation is:
$$V_{\max} = \sqrt{P_{\text{rating}} \times R}$$
  - *Exceeding  $V_{\max}$  causes the resistor's power dissipation to exceed its power rating*

# Example

- Determine the required resistance and appropriate wattage rating of a carbon-film resistor to meet the following requirements:
- The resistor has a 54-V IR drop when its current is 20 mA. The resistors available have the following wattage ratings:

$$P_{\text{rating}} = V^2/R = I^2R = VI = 54 \times 0.020 = 1.08 W$$

# Electric Shock



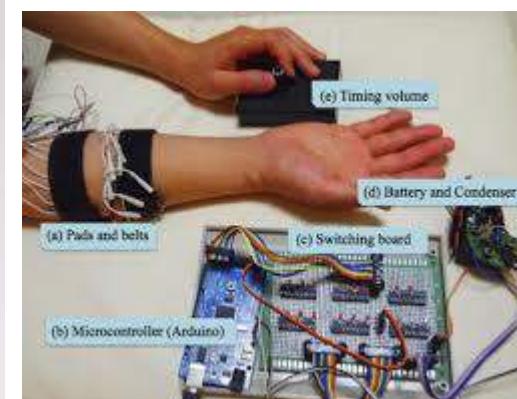
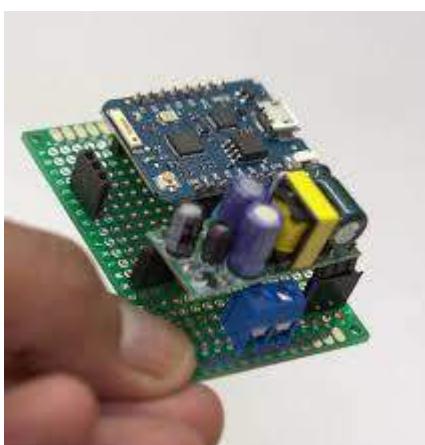
- When possible, work only on circuits that have the power shut off.
- If the power must be on, use only one hand when making voltage measurements.
- Keep yourself insulated from earth ground.
- Hand-to-hand shocks can be very dangerous because current is likely to flow through the heart!

## Electrical Harm

Estimated Effects of AC Currents (U.S. Standard 60 Hz)	
1 milliamp (mA)	Barely perceptible
16 mA	Maximum current an average man can grasp and "let go"
20 – 30 mA	Paralysis of respiratory muscles
100 mA	Ventricular fibrillation threshold
2 Amps	Cardiac standstill and internal organ damage
15/20/30 Amps	Common U.S. household breakers



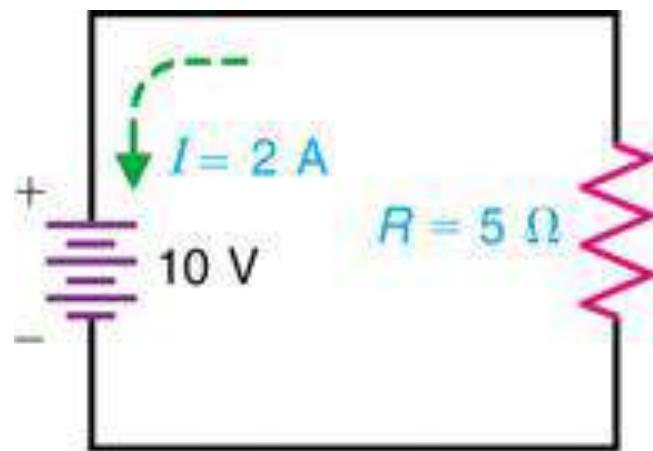
**PATH:**  
Harm is related to the path by which current passes through the body.



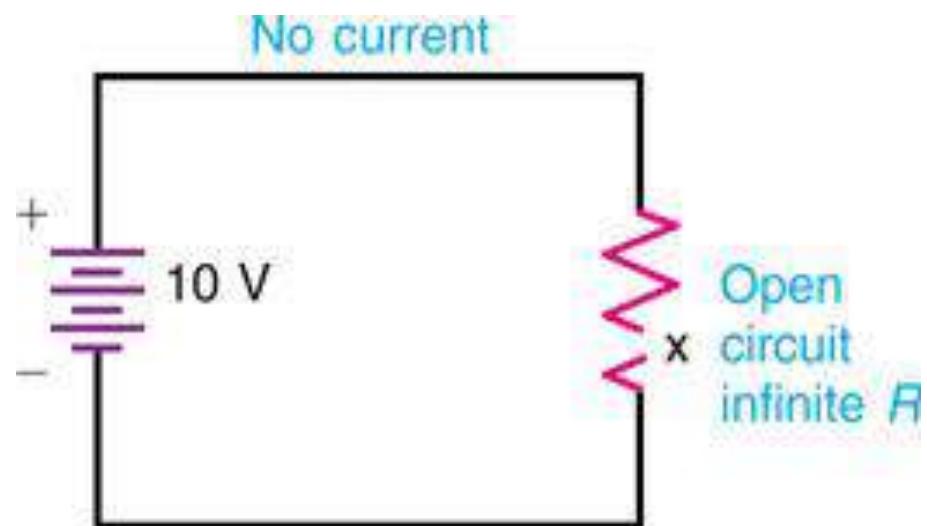


## Open Circuit

An open circuit has zero current flow.



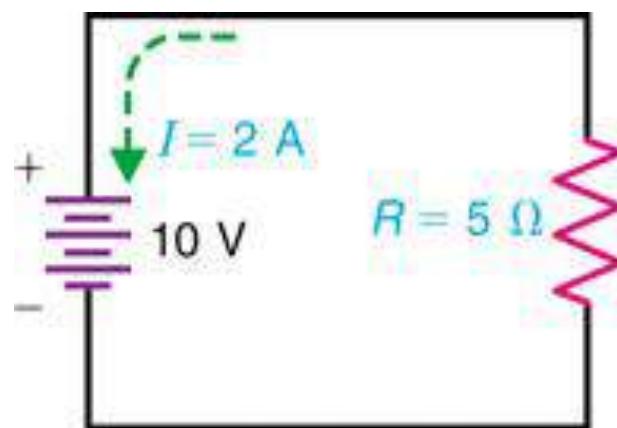
(a)



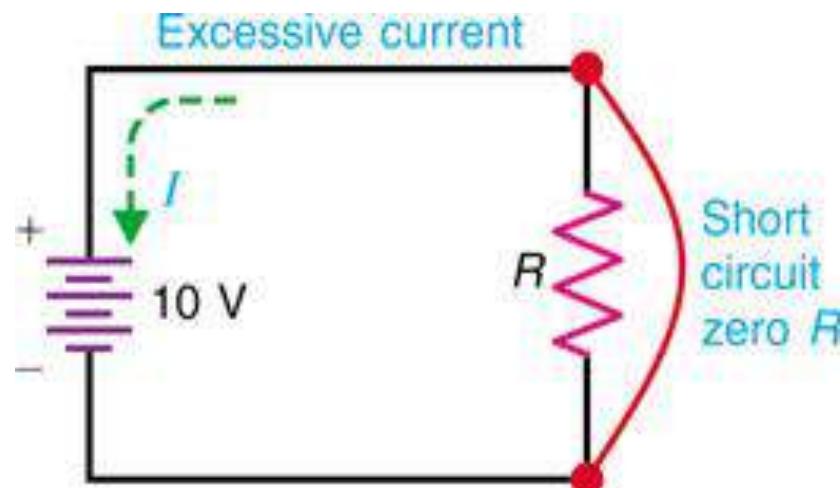
(b)

## Short Circuit

A short circuit has excessive current flow.  
As  $R$  approaches 0,  $I$  approaches  $\infty$ .



(a)



(b)

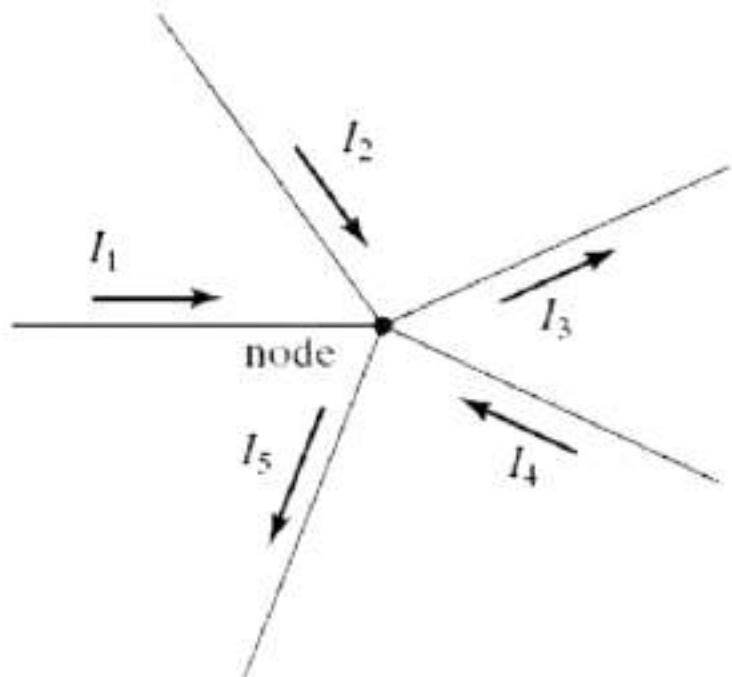


# KIRCHHOFF'S LAW

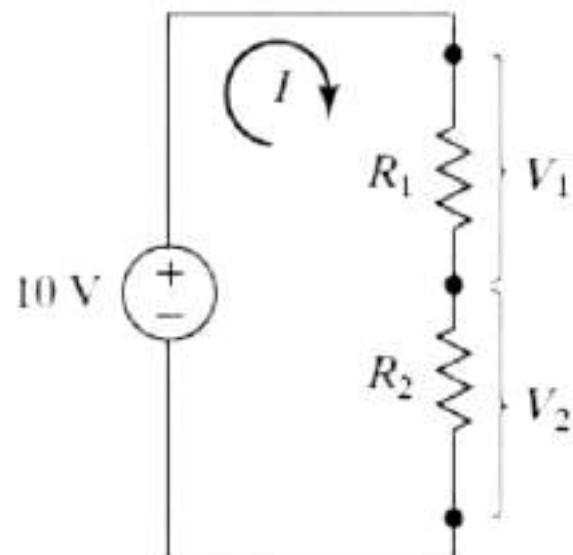
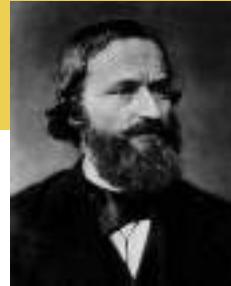
# Kirchhoff's Law



$$\begin{aligned}\sum I_{\text{in}} &= 0 \\ &= I_1 + I_2 - I_3 + I_4 - I_5\end{aligned}$$



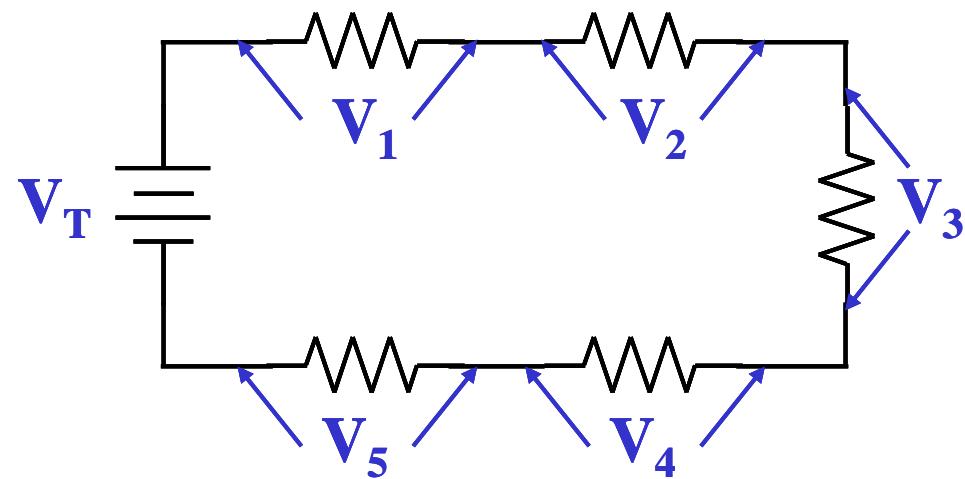
$$\begin{aligned}\sum V &= 0 \\ &= +10 - V_1 - V_2\end{aligned}$$



# Kirchhoff's Voltage Law (KVL)



The total voltage is equal to the sum of the drops.



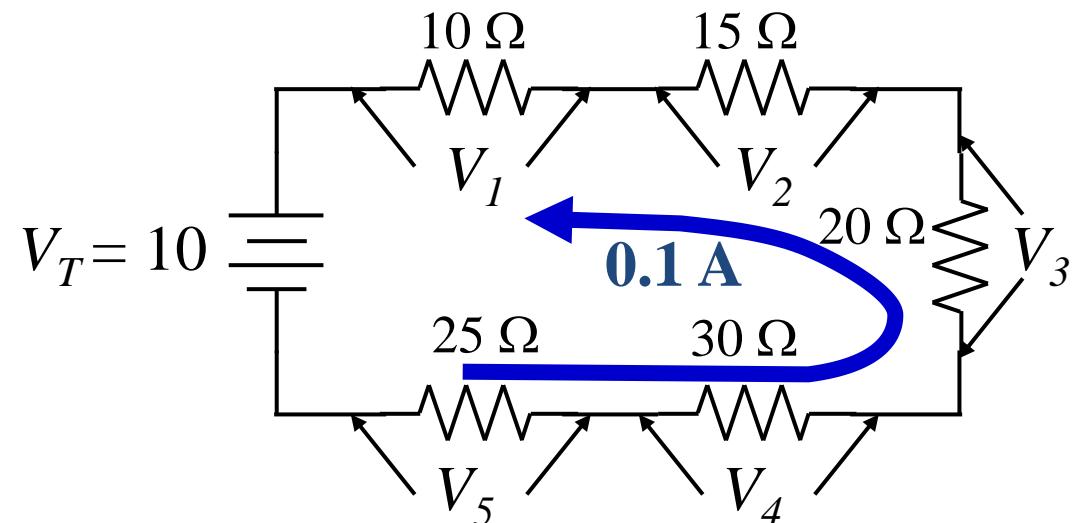
$$V_T = V_1 + V_2 + V_3 + V_4 + V_5$$

This is known as  
**Kirchhoff's voltage law (KVL).**

# Kirchhoff's Voltage Law (KVL)



The  $IR$  drops must add to equal the applied voltage (KVL).



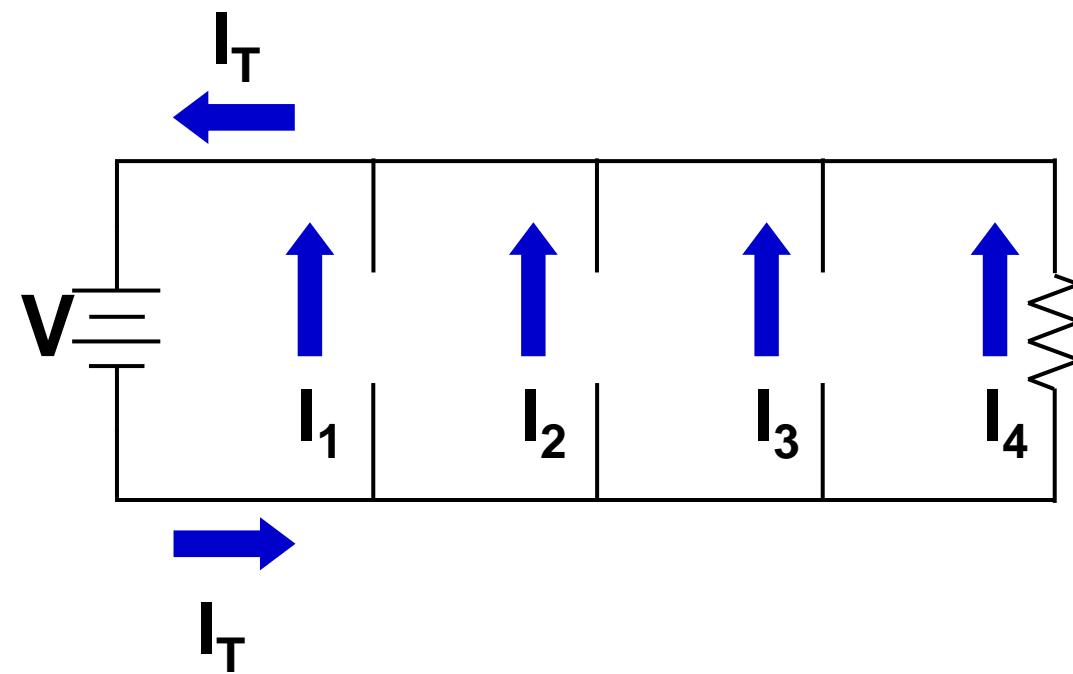
$$V_T = V_1 + V_2 + V_3 + V_4 + V_5$$

$$V_T = IR_1 + IR_2 + IR_3 + IR_4 + IR_5$$

$$V_T = 0.1 \times 10 + 0.1 \times 15 + 0.1 \times 20 + 0.1 \times 30 + 0.1 \times 25$$

$$V_T = 1\text{ V} + 1.5\text{ V} + 2\text{ V} + 3\text{ V} + 2.5\text{ V} = 10\text{ V}$$

# Kirchhoff's Current Law (KCL)

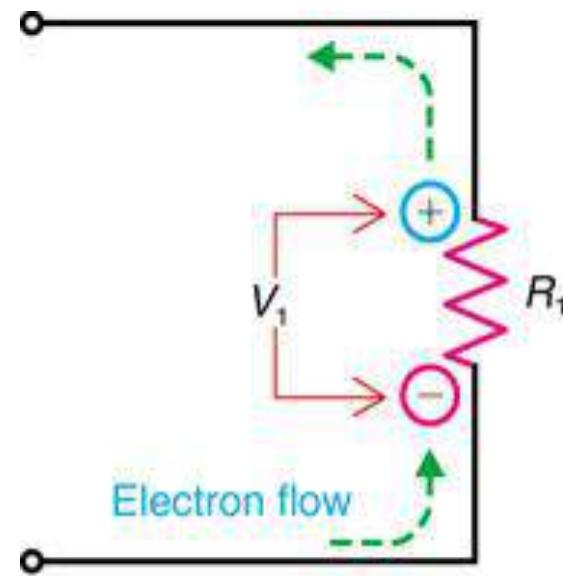


$$I_T = I_1 + I_2 + I_3 + I_4$$

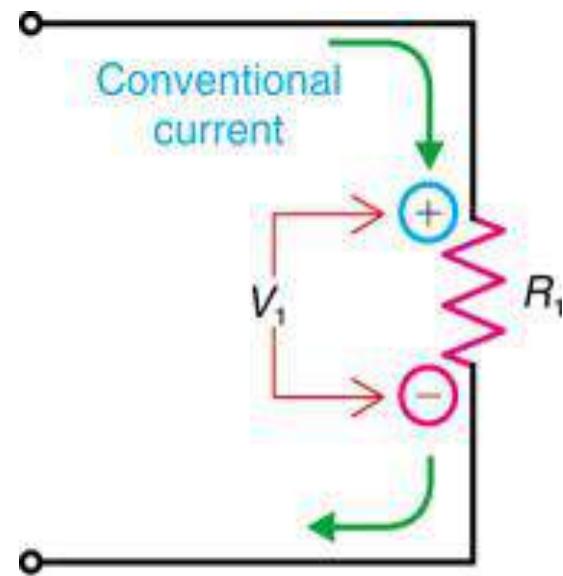


# SERIES-PARALLEL CIRCUITS

# Polarity of $IR$ Voltage Drops



(a)



(b)

Polarity of  $IR$  voltage drops.

- (a) Electrons flow into the negative side of  $V_1$  across  $R_1$ .
- (b) Same polarity of  $V_1$  with positive charges into the positive side.

## Why / Is the Same in All Parts of a Series Circuit

- Characteristics of a Series Circuit (1)
  - The current is the same everywhere in a series circuit.
  - The total resistance is equal to the sum of the individual resistance values.
  - The total voltage is equal to the sum of the  $IR$  voltage drops across the individual resistances.
  - The total power is equal to the sum of the power dissipated by each resistance.



## Why $I$ Is the Same in All Parts of a Series Circuit

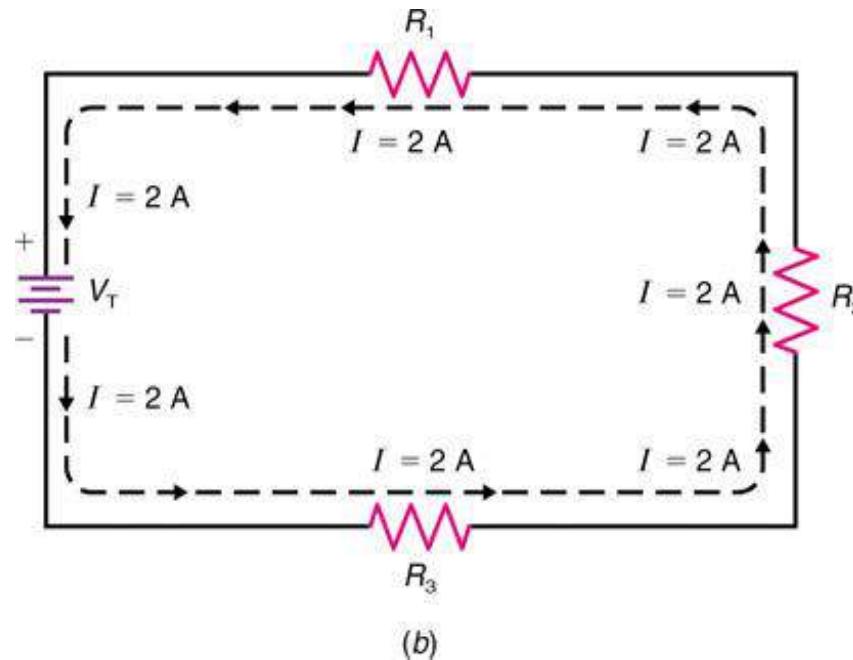
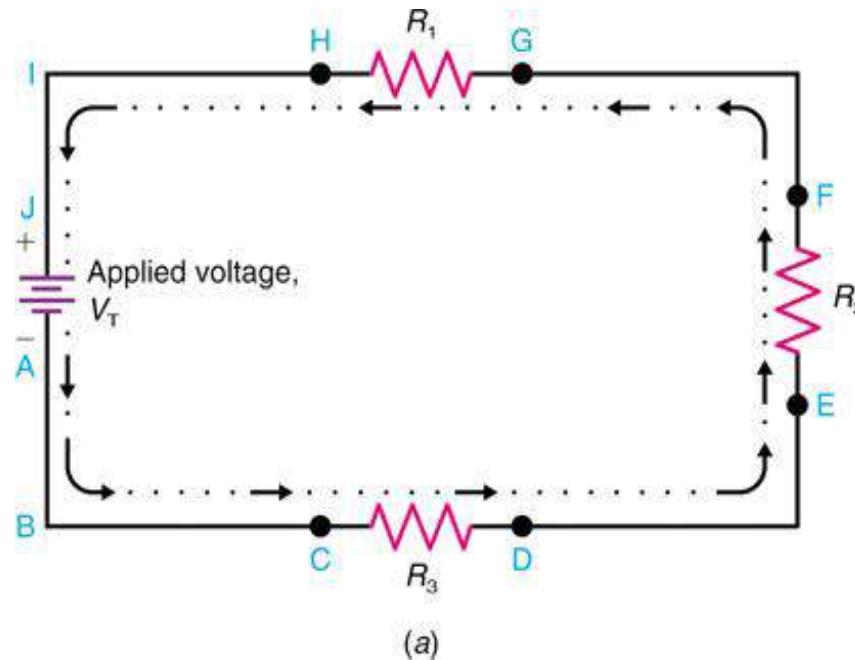
- Characteristics of a Series Circuit (2)
  - Current is the movement of electric charge between two points, produced by the applied voltage.
  - The free electrons moving away from one point are continuously replaced by free electrons flowing from an adjacent point in the series circuit.
  - All electrons have the same speed as those leaving the voltage source.
  - Therefore,  $I$  is the same in all parts of a series circuit.



## Why $I$ Is the Same in All Parts of a Series Circuit

There is only one current through  $R_1$ ,  $R_2$ , and  $R_3$  in series.

- (a) Electron drift is the same in all parts of a series circuit.
- (b) Current  $I$  is the same at all points in a series circuit.



## Why $I$ Is the Same in All Parts of a Series Circuit

- Series Current Formulas
  - Total current is the same as the individual currents in the series string:

$$I_T = I_1 = I_2 = I_3 = \dots = \text{etc.}$$

- Total current is equal to total voltage divided by total resistance:

$$I_T = \frac{V_T}{R_T}$$



## Total $R$ Equals the Sum of All Series Resistances

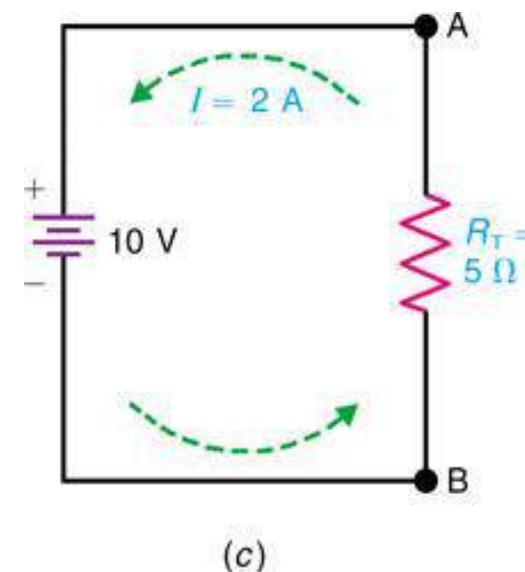
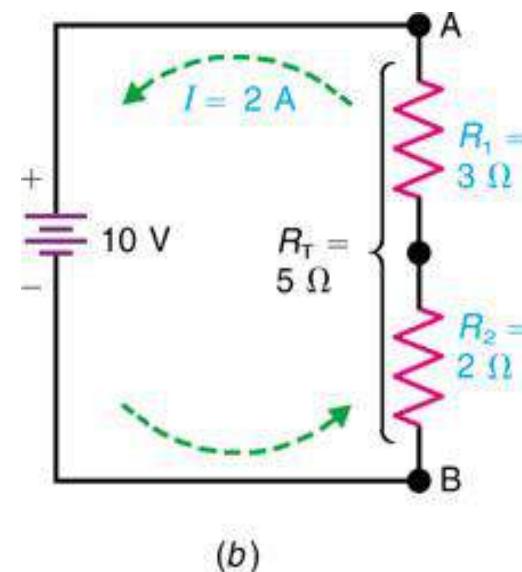
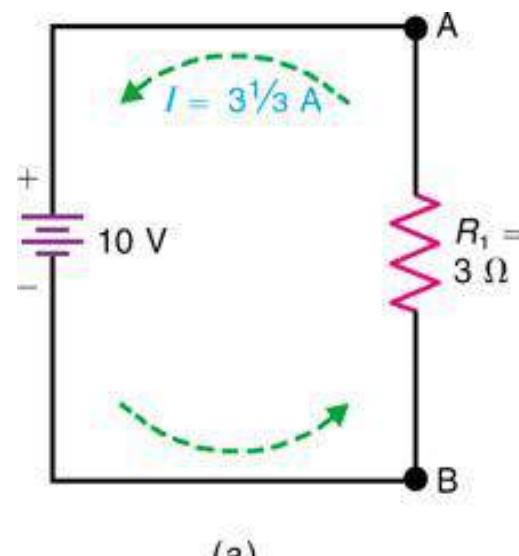
- When a series circuit is connected across a voltage source, the free electrons must drift through all the series resistances.
- There is only one path for free electrons to follow.
- If there are two or more resistances in the same current path, the total resistance across the voltage source is the sum of all the resistances.



## Total $R$ Equals the Sum of All Series Resistances

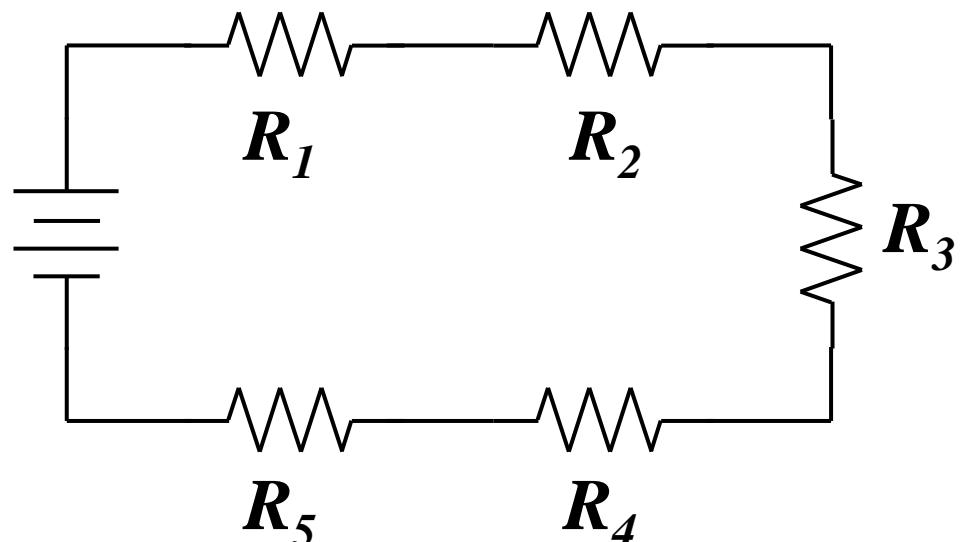
Series resistances are added for the total  $R_T$ .

- (a)  $R_1$  alone is  $3 \Omega$ .
- (b)  $R_1$  and  $R_2$  in series together total  $5 \Omega$ .
- (c) The  $R_T$  of  $5 \Omega$  is the same as one resistance of  $5 \Omega$  between points A and B.



## Total $R$ Equals the Sum of All Series Resistances

- Series Resistance Formulas
  - The total resistance is the sum of the individual resistances.



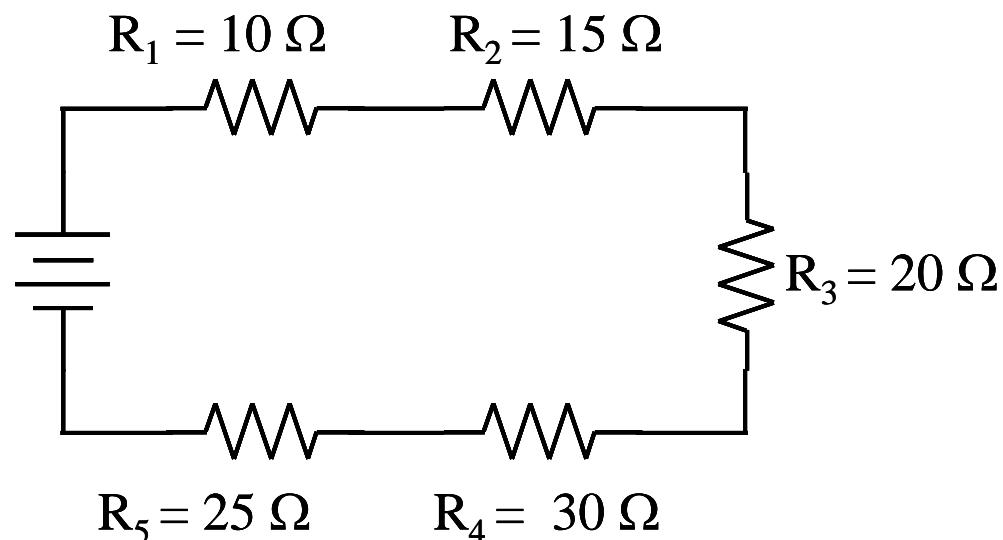
$$R_T = R_1 + R_2 + R_3 + R_4 + R_5$$

## Total $R$ Equals the Sum of All Series Resistances

- Series Resistance Formulas
  - Total resistance is equal to total voltage divided by the circuit current:

$$R_T = \frac{V_T}{I_T}$$

## Determining the Total Resistance



$$R_T = R_1 + R_2 + R_3 + R_4 + R_5$$

$$R_T = 10 \Omega + 15 \Omega + 20 \Omega + 30 \Omega + 25 \Omega = 100 \Omega$$

# Series *IR* Voltage Drops

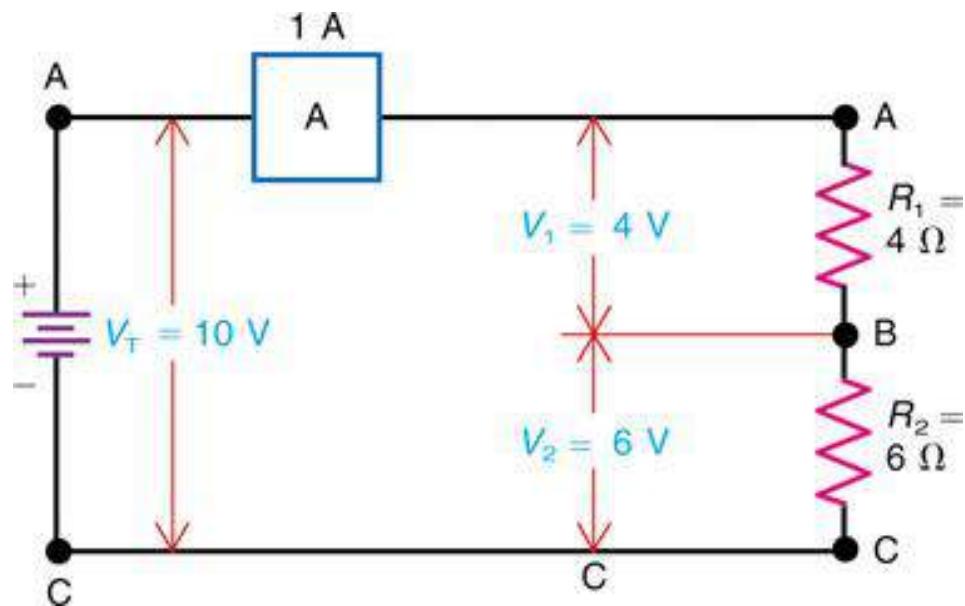


- By **Ohm's Law**, the voltage across a resistance equals  $I \times R$ .
- In a series circuit, the  $IR$  voltage across each resistance is called an  **$IR$  drop** or **voltage drop**, because it reduces the potential difference available for the remaining resistances in the circuit.

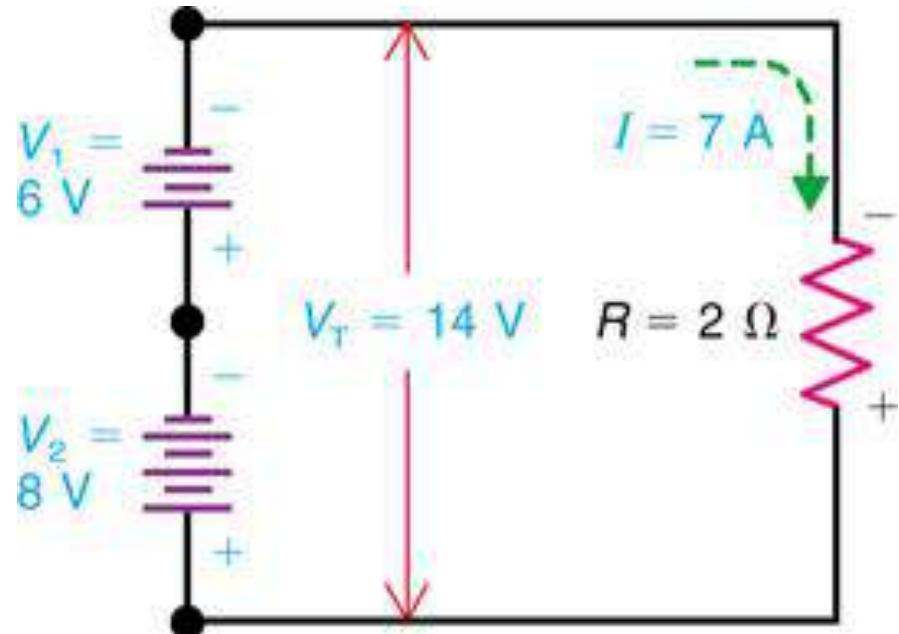
# Series *IR* Voltage Drops



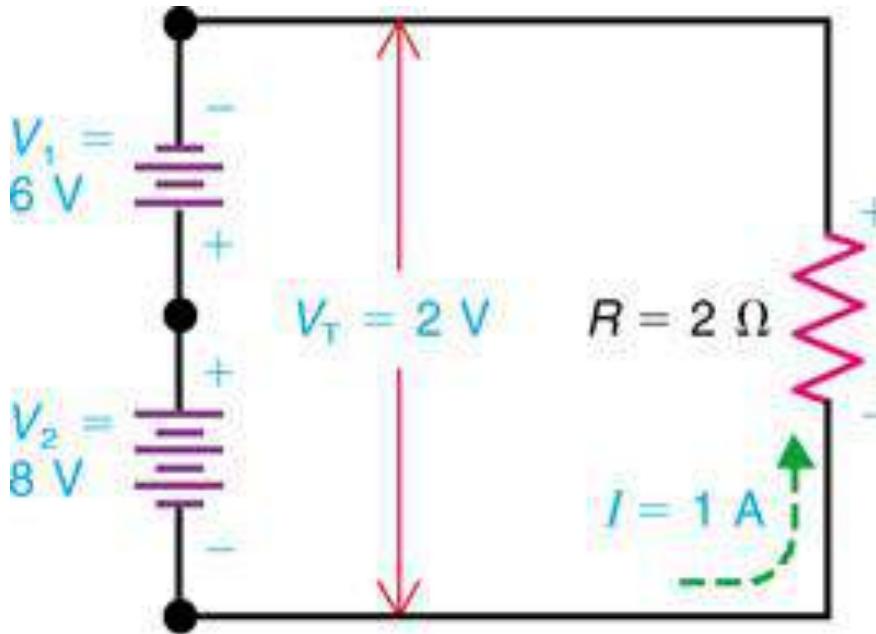
An example of *IR* voltage drops  $V_1$  and  $V_2$  in a series circuit.



# Series-Aiding and Series-Opposing Voltages



(a)



(b)

Example of voltage sources  $V_1$  and  $V_2$  in series.

- (a) Note the connections for series-aiding polarities. Here  $8\text{ V} + 6\text{ V} = 14\text{ V}$  for the total  $V_T$ .
- (b) Connections for series-opposing polarities. Now  $8\text{ V} - 6\text{ V} = 2\text{ V}$  for  $V_T$ .

# Ground Connections in Electrical and Electronic Systems



- In most electrical and electronic systems, one side of the voltage source is connected to ground.
- The reason for doing this is to reduce the possibility of electric shock.

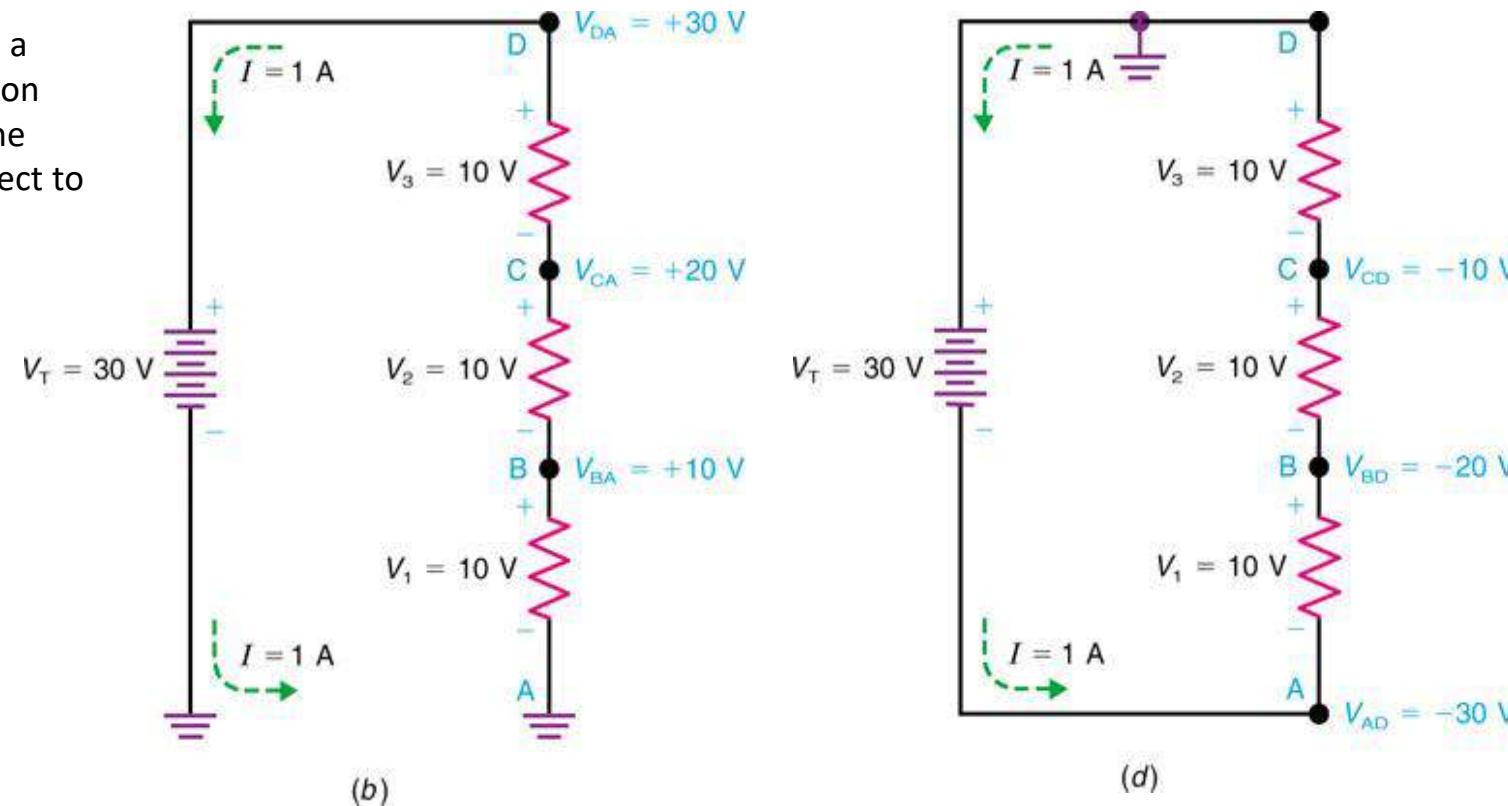


- Ground is assumed to have a potential of 0 V regardless of the schematic symbol shown.
- These symbols are sometimes used inconsistently with their definitions. However, these symbols always represent a common return path for current in a given circuit.

# Ground Connections in Electrical and Electronic Systems



When a circuit has a ground as a common return, measure the voltages with respect to this ground.



An example of how to calculate dc voltages measured with respect to ground. (b) Negative side of  $V_T$  grounded to make all voltages positive with respect to ground. (d) Positive side of  $V_T$  grounded, all voltages are negative to ground.



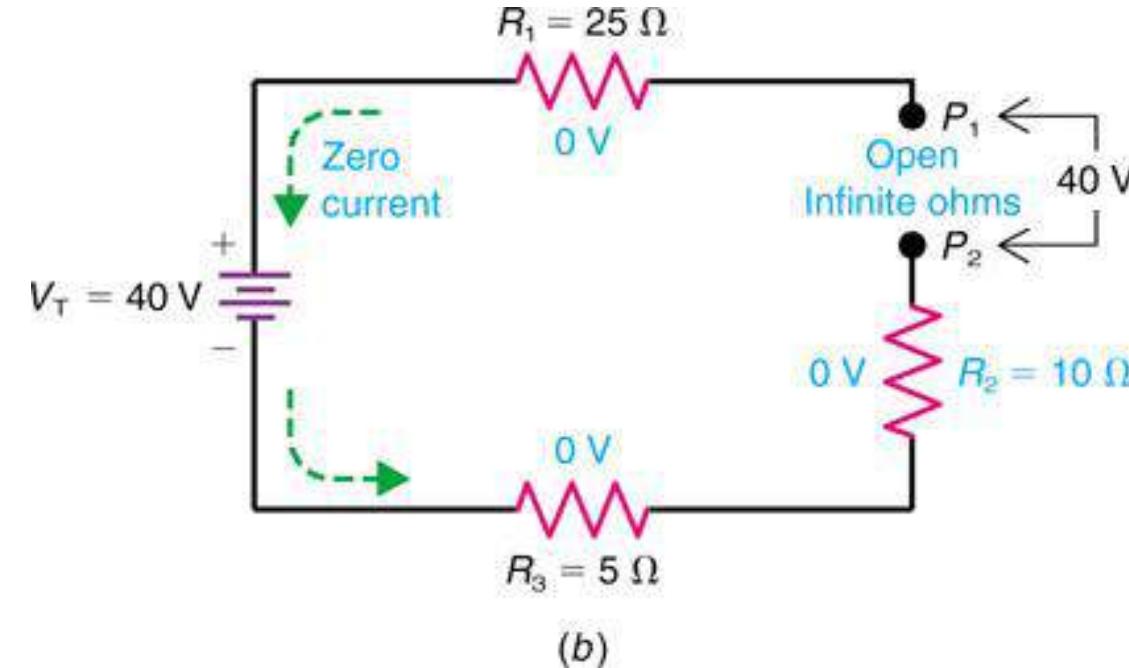
## The Effect of an Open in a Series Circuit

- The Effect of an Open in a Series Circuit
  - An open circuit is a circuit with a break in the current path. When a series circuit is open, the current is zero in all parts of the circuit.
  - The total resistance of an open circuit is infinite ohms.
  - When a series circuit is open, the applied voltage appears across the open points.

# Troubleshooting: Opens and Shorts in Series Circuits



## Example of an Open Circuit



Effect of an open in a series circuit. (b) Open path between points  $P1$  and  $P2$  results in zero current in all parts of the circuit.

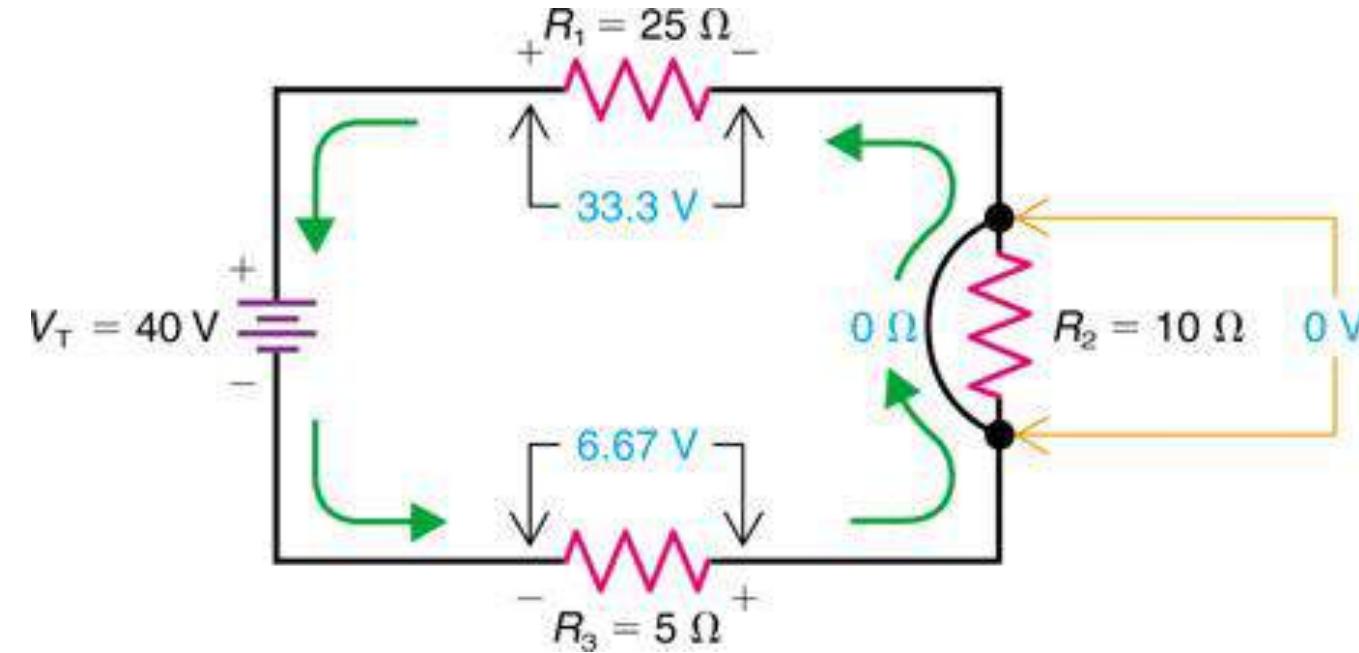
- Applied voltage  $V_T$  is still present, even with zero current.
- The voltage source still has its same potential difference across its positive and negative terminals.
- Example: The 120-V potential difference is always available from the terminals of a wall outlet.
  - If an appliance is connected, current will flow.
  - If you touch the metal terminals when nothing else is connected, you will receive a shock.



## The Effect of a Short in a Series Circuit

- The Effect of a Short in a Series Circuit
  - When part of a series circuit is shorted, the current flow increases.
  - When part of a series circuit is shorted, the voltage drops across the non-shorted elements increase.
  - The voltage drop across the shorted component drops to 0 V.

## Example of a Short Circuit



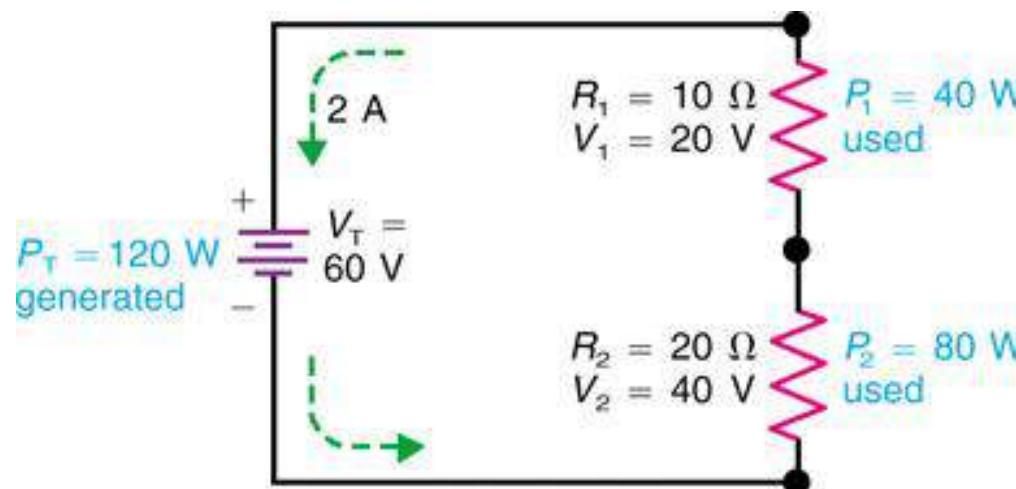
Series circuit with  $R_2$  shorted.

- When trying to analyze a series circuit, keep the following principles in mind:
  1. If  $I$  is known for one component, use this value in all components. The current is the same in all parts of a series circuit.
  2. If  $I$  is unknown, it may be calculated in one of two ways:
    - Divide  $V_T$  by  $R_T$
    - Divide an individual  $IR$  drop by its  $R$ .
    - Remember not to mix a total value for an entire circuit with an individual value for part of the circuit.
  3. If all individual voltage drops are known, add them to determine the applied  $V_T$ .
    - A known voltage drop may be subtracted from  $V_T$  to find a remaining voltage drop.

# Total Power in a Series Circuit

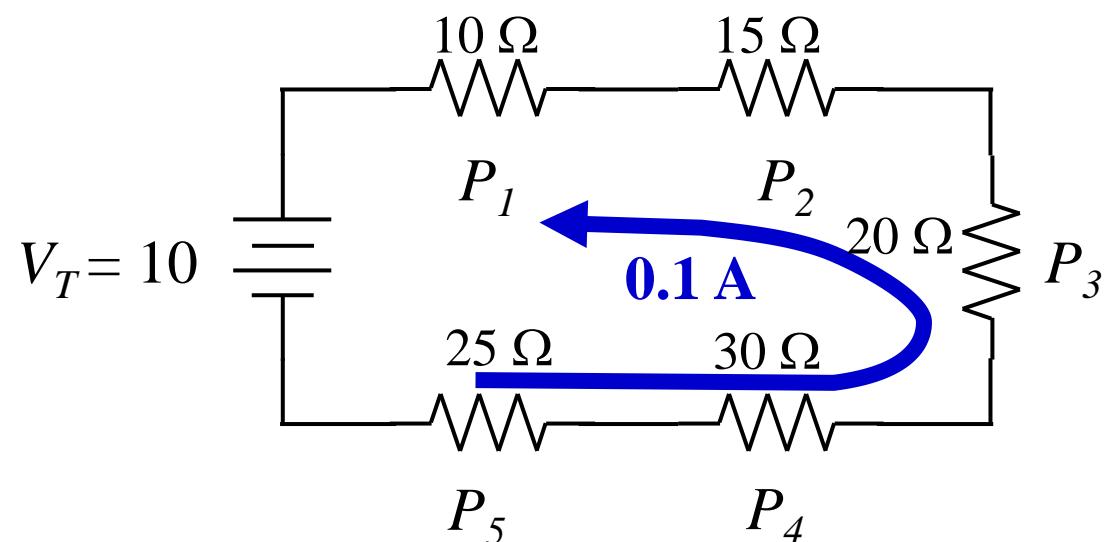


- The power needed to produce current in each series resistor is used up in the form of heat.
- The total power used in the circuit is equal to the sum of the individual powers dissipated in each part of the circuit.
- Total power can also be calculated as  $V_T \times I$



The sum of the individual powers  $P_1$  and  $P_2$  used in each resistance equals the total power  $P_T$  produced by the source.

## Finding Total Power



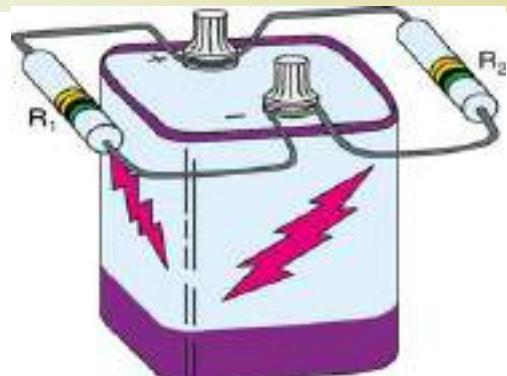
$$P_T = P_1 + P_2 + P_3 + P_4 + P_5$$

$$P_T = I^2R_1 + I^2R_2 + I^2R_3 + I^2R_4 + I^2R_5$$

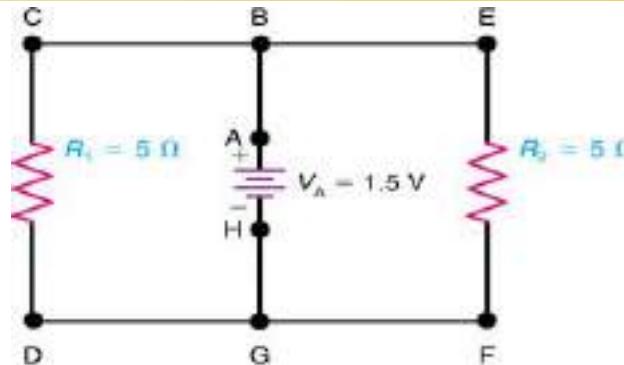
$$P_T = 0.1 \text{ W} + 0.15 \text{ W} + 0.2 \text{ W} + 0.3 \text{ W} + 0.25 \text{ W} = 1 \text{ W}$$

$$\text{Check: } P_T = V_T \times I = 10 \text{ V} \times 0.1 \text{ A} = 1 \text{ W}$$

# Parallel Circuit



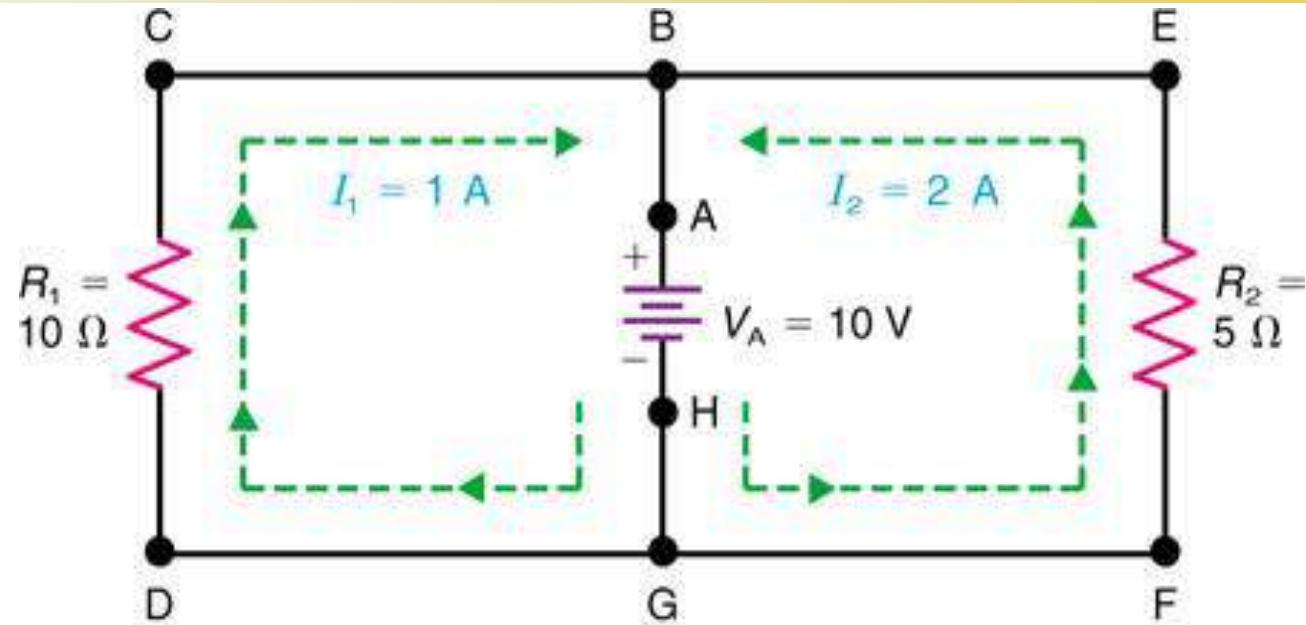
(a)



(b)

- The current in a parallel circuit equals the voltage applied across the circuit divided by the resistance between the two points where the voltage is applied.
- Each path for current in a parallel circuit is called a **branch**. Each branch current equals  $V/R$  where  $V$  is the same across all branches.

# Parallel Circuit



The current in each parallel branch equals the applied voltage  $V_A$  divided by each branch resistance  $R$ .



## Resistance in Parallel

- Total Current and Reciprocal Resistance Formulas
  - The equivalent resistance of a parallel circuit equals the reciprocal of the sum of the reciprocals:

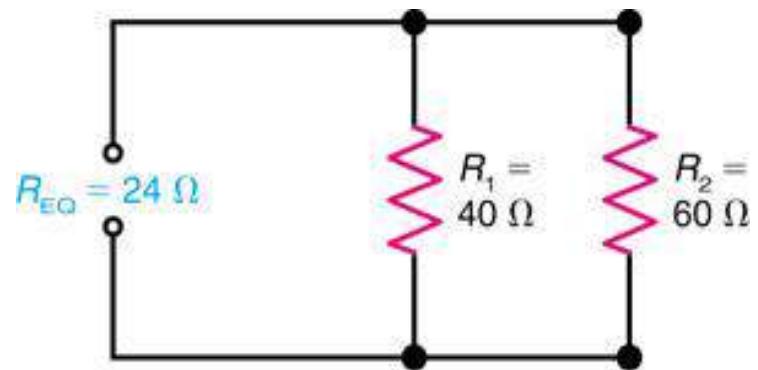
$$R_{EQ} = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \dots + \text{etc.}}$$

- Equivalent resistance also equals the applied voltage divided by the total current:

$$R_{EQ} = \frac{V_A}{I_T}$$

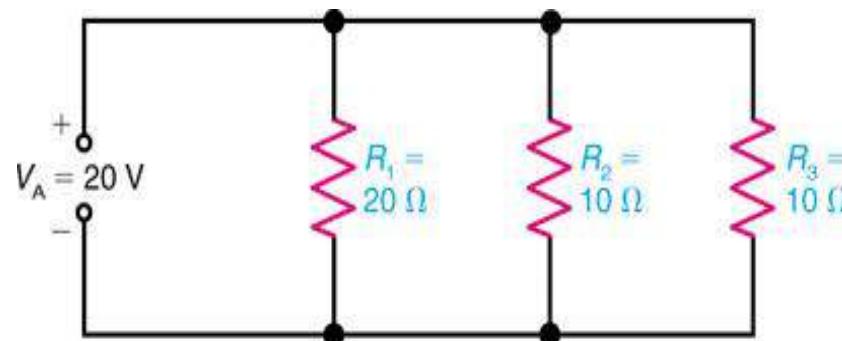
## Resistances in Parallel Example

$$R_{EQ} = \frac{R_1 \times R_2}{R_1 + R_2}$$



$$R_{EQ} = \frac{R_1 \times R_2}{R_1 + R_2} = \frac{2400}{100}$$

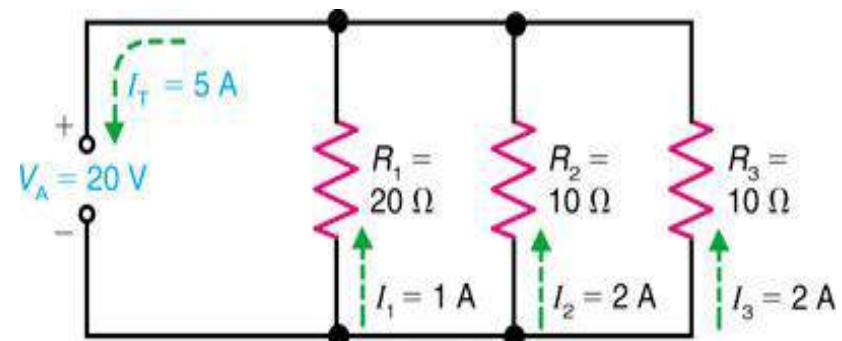
## Resistances in Parallel Example



$$R_{EQ} = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}}$$

$R_{EQ} = 4\Omega$

(a)



$$R_{EQ} = \frac{V_A}{I_T} = \frac{20\text{ V}}{5\text{ A}}$$

$R_{EQ} = 4\Omega$

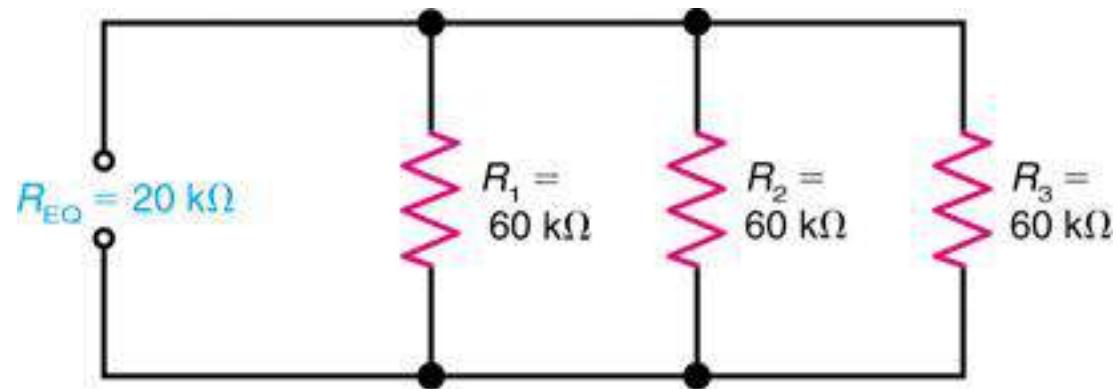
(b)

## (Equal Value) Resistances in Parallel Example

$$R_{EQ} = \frac{R}{N}$$

$$R_{EQ} = \frac{60 \text{ k}\Omega}{3 \text{ resistors}}$$

$$R_{EQ} = 20 \text{ k}\Omega$$

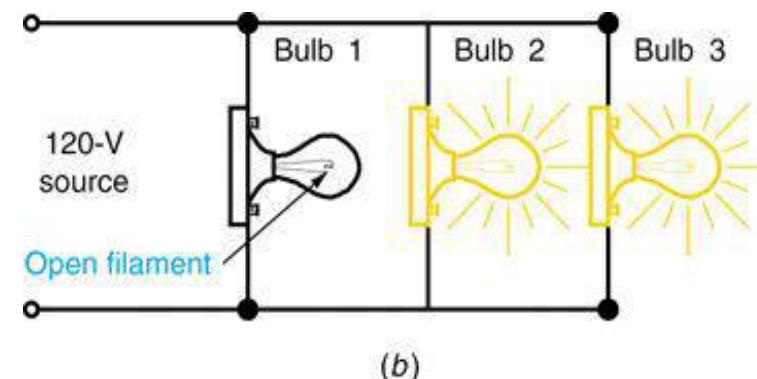
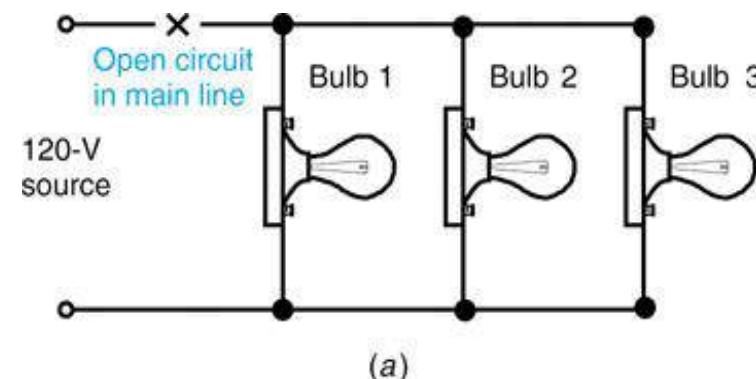


$$R_{EQ} = \frac{\text{value of one resistance}}{\text{number of resistances}} = \frac{60 \text{ k}\Omega}{3}$$



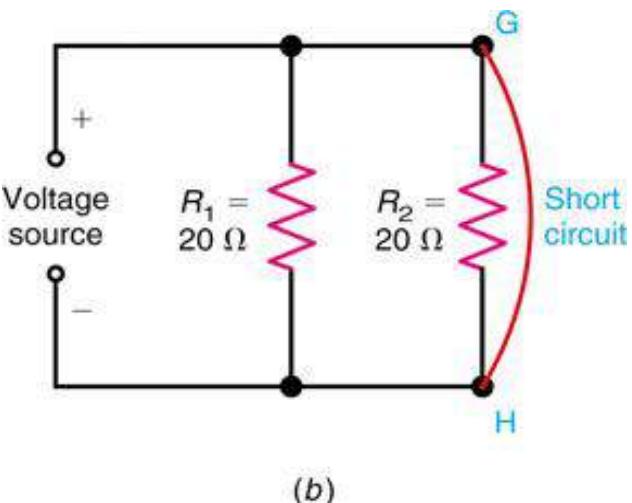
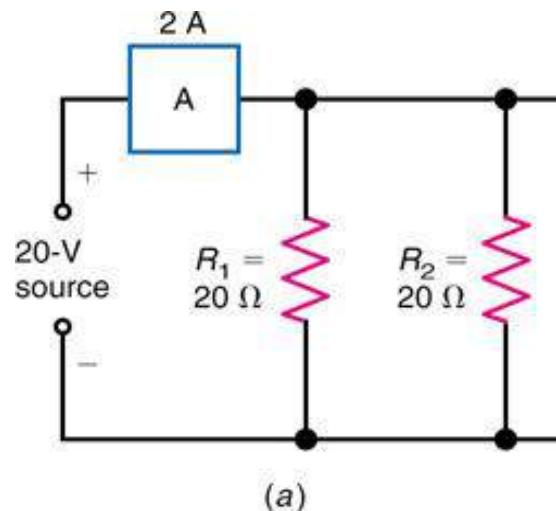
## Troubleshooting

- An open circuit in one branch results in no current through that branch.
- However, an open circuit in one branch has no effect on the other branches. This is because the other branches are still connected to the voltage source.
- An open in the main line prevents current from reaching any branch, so all branches are affected.



## Troubleshooting

- How about the Short Circuit?



Effect of a short circuit across parallel branches.

- Normal circuit. (
- Short circuit across points H and G shorts out all the branches.



# DIVIDER

# Series Voltage Dividers (1/4)



- $V_T$  is divided into  $IR$  voltage drops that are proportional to the series resistance values.

- Each resistance provides an  $IR$  voltage drop equal to its proportional part of the applied voltage:

$$V_R = (R/R_T) \times V_T$$

- This formula can be used for any number of series resistances because of the direct proportion between each voltage drop  $V$  and its resistance  $R$ .
- The largest series  $R$  has the largest  $IR$  voltage drop.

# Series Voltage Dividers (2/4)



The Largest Series  $R$  Has the Most  $V$ .

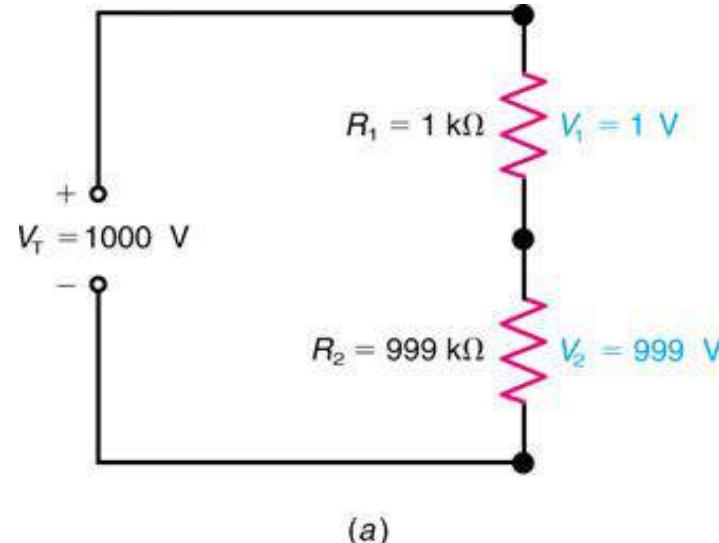
$$V_1 = \frac{R_1}{R_T} \times V_T$$
$$= \frac{1 \text{ k}\Omega}{1000 \text{ k}\Omega} \times 1000 \text{ V} = 1 \text{ V}$$

$$V_2 = \frac{R_2}{R_T} \times V_T$$
$$= \frac{999 \text{ k}\Omega}{1000 \text{ k}\Omega} \times 1000 \text{ V} = 999 \text{ V}$$

KVL check:  $1 \text{ V} + 999 \text{ V} = 1000 \text{ V}$

KVL: Kirchhoff's Voltage Law

KCL: Kirchhoff's Current Law

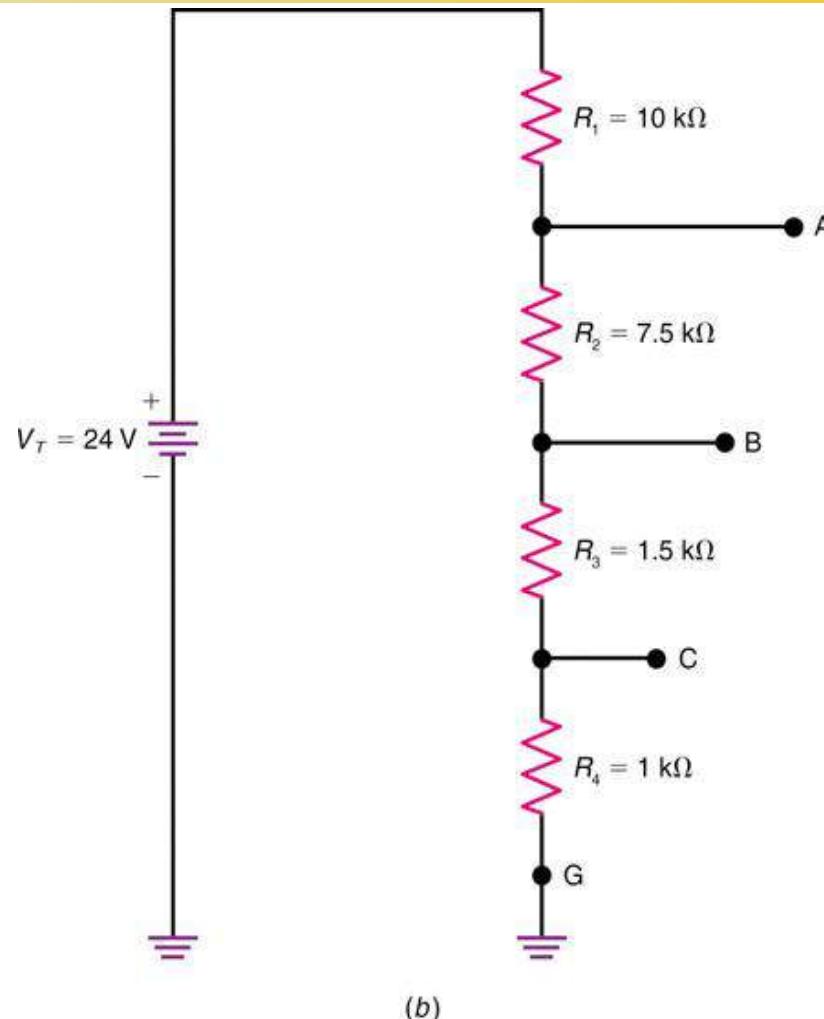


Example of a very small  $R_1$  in series with a large  $R_2$ ;  $V_2$  is almost equal to the whole  $V_T$ .

# Series Voltage Dividers (3/4)



- Voltage Taps in a Series Voltage Divider
  - Different voltages are available at **voltage taps** A, B, and C.
  - The voltage at each tap point is measured with respect to ground.
  - Ground is the reference point.



Series voltage divider with voltage taps.

# Series Voltage Dividers (4/4)

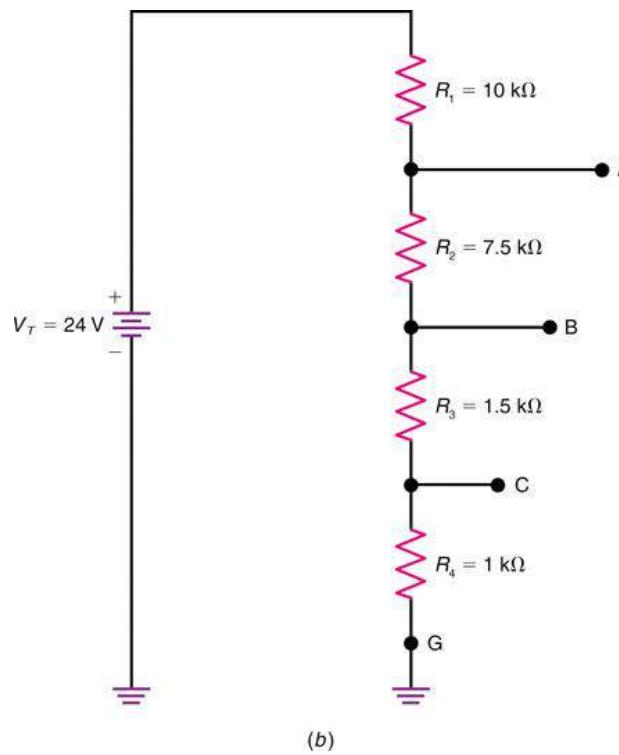


- Note:  $V_{AG}$  is the sum of the voltage across  $R_2$ ,  $R_3$ , and  $R_4$ .
- $V_{AG}$  is one-half of the applied voltage  $V_T$ ,  
because  $R_2+R_3+R_4 = 50\%$  of  $R_T$

## Voltage Taps in a Series Voltage Divider

$$V_{AG} = 12 \text{ V} \quad V_{BG} = \frac{2.5 \text{ k}\Omega}{20 \text{ k}\Omega} \times 24 \text{ V} = 3 \text{ V}$$

$$V_{CG} = \frac{1 \text{ k}\Omega}{20 \text{ k}\Omega} \times 24 \text{ V} = 1.2 \text{ V}$$





# Current Dividers with Two Parallel Resistances (1/2)

$$I_1 = \frac{R_2}{R_1 + R_2} \times I_T$$

- $I_T$  is divided into individual branch currents.
- Each branch current is inversely proportional to the branch resistance value.
- For two resistors,  $R_1$  and  $R_2$ , in parallel:
- Note that this formula can only be used for two branch resistances.
- The largest current flows in the branch that has the smallest  $R$ .

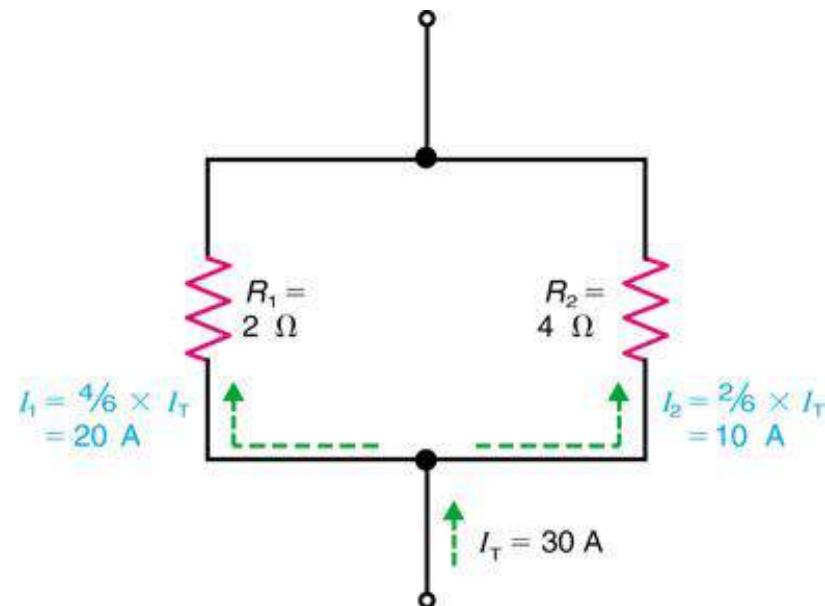
## Current Dividers with Two Parallel Resistances (2/2)



- Current Divider

$$I_1 = 4 \Omega / (2 \Omega + 4 \Omega) \times 30A = 20A$$

$$I_2 = 2 \Omega / (2 \Omega + 4 \Omega) \times 30A = 10A$$



Current divider with two branch resistances. Each branch  $I$  is inversely proportional to its  $R$ . The smaller  $R$  has more  $I$ .

# Current Division by Parallel Conductances (1/3)



- For any number of parallel branches,  $I_T$  is divided into currents that are proportional to the conductance of the branches.
- For a branch having conductance  $G$ :

$$I = \frac{G}{G_T} \times I_T$$

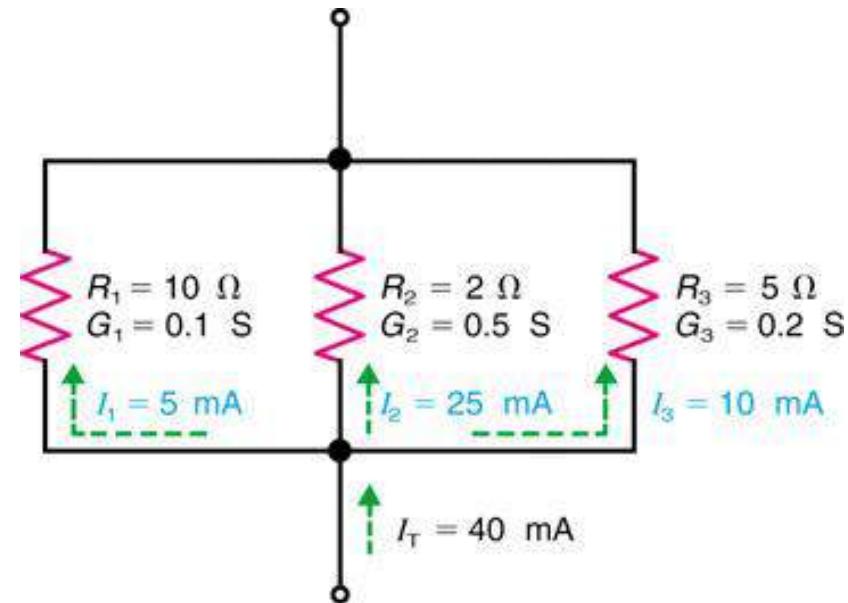
# Current Division by Parallel Conductances (2/3)



$$G_1 = 1/R_1 = 1/10 \Omega = 0.1 \text{ S}$$

$$G_2 = 1/R_2 = 1/2 \Omega = 0.5 \text{ S}$$

$$G_3 = 1/R_3 = 1/5 \Omega = 0.2 \text{ S}$$



Current divider with branch conductances  $G_1$ ,  $G_2$ , and  $G_3$ , each equal to  $1/R$ . Note that S is the siemens unit for conductance. With conductance values, each branch  $I$  is directly proportional to the branch  $G$ .



The Siemens (S) unit is the reciprocal of the ohm ( $\Omega$ )

$$\begin{aligned}G_T &= G_1 + G_2 + G_3 \\&= 0.1 + 0.5 + 0.2\end{aligned}$$

$$G_T = 0.8 \text{ S}$$

$$I_1 = 0.1/0.8 \times 40 \text{ mA} = 5 \text{ mA}$$

$$I_2 = 0.5/0.8 \times 40 \text{ mA} = 25 \text{ mA}$$

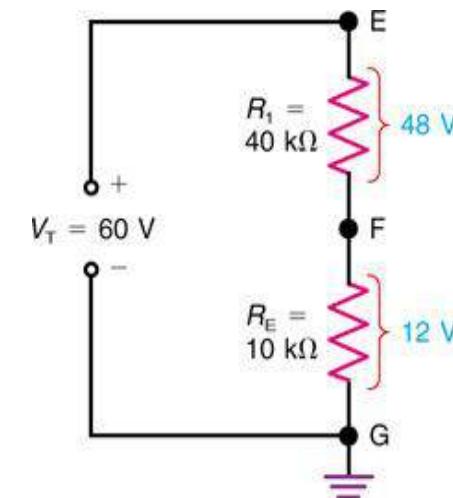
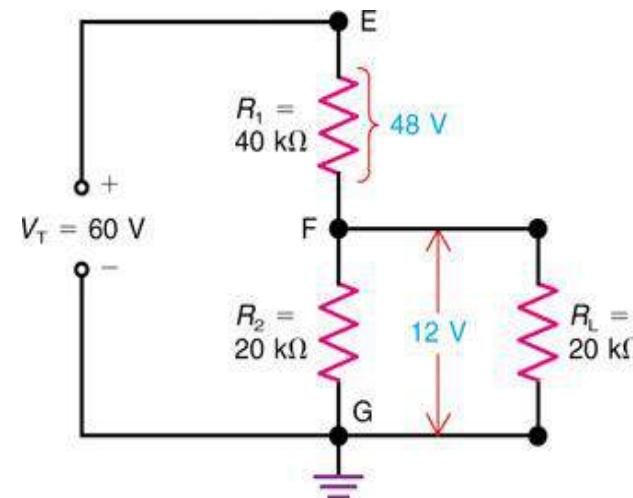
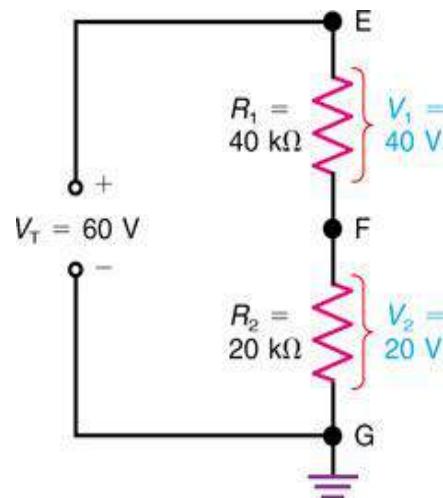
$$I_3 = 0.2/0.8 \times 40 \text{ mA} = 10 \text{ mA}$$

$$\text{KCL check: } 5 \text{ mA} + 25 \text{ mA} + 10 \text{ mA} = 40 \text{ mA} = I_T$$

# Series Voltage Divider with Parallel Load Current (1/3)



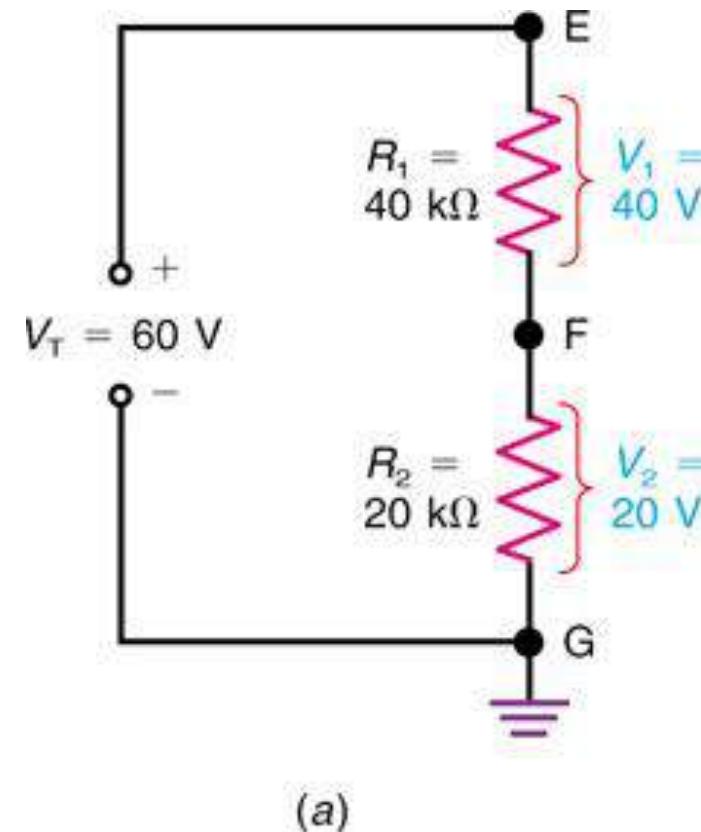
- Voltage dividers are often used to tap off part of the applied voltage for a load that needs less than the total voltage.



Effect of a parallel load in part of a series voltage divider. (a)  $R_1$  and  $R_2$  in series without any branch current. (b) Reduced voltage across  $R_2$  and its parallel load  $R_L$ . (c) Equivalent circuit of the loaded voltage divider.

## Series Voltage Divider with Parallel Load Current (2/3)

- $V_1 = 40/60 \times 60 \text{ V} = 40 \text{ V}$
- $V_2 = 20/60 \times 60 \text{ V} = 20 \text{ V}$
- $V_1 + V_2 = V_T = 60 \text{ V}$   
(Applied Voltage)

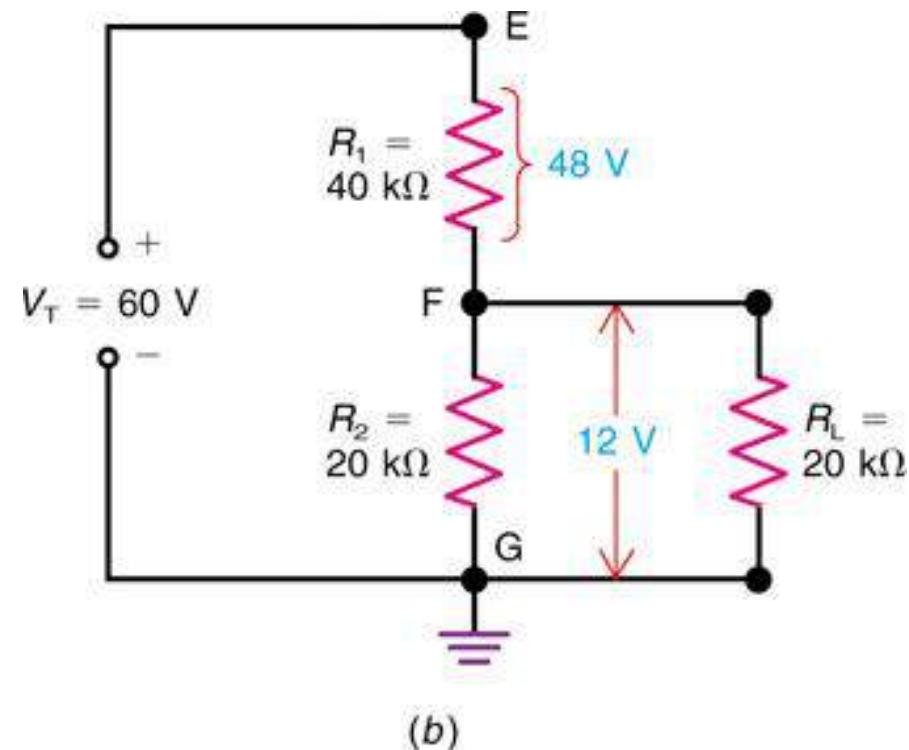


(a)

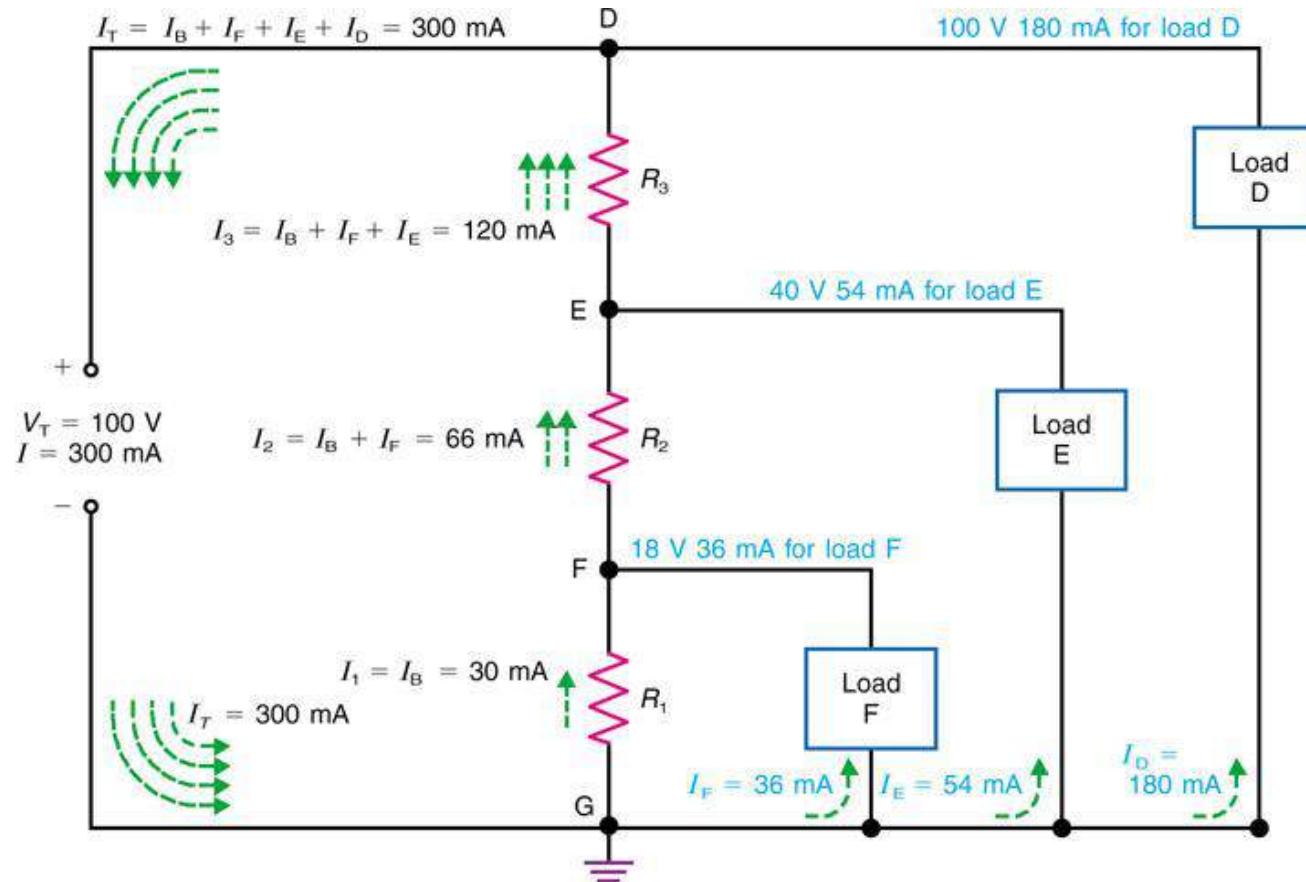
# Series Voltage Divider with Parallel Load Current (3/3)



- The current that passes through all the resistances in the voltage divider is called the **bleeder current**,  $I_B$ .
- Resistance  $R_L$  has just its load current  $I_L$ .
- Resistance  $R_2$  has only the bleeder current  $I_B$ .
- Resistance  $R_1$  has both  $I_L$  and  $I_B$ .



# Design of a Loaded Voltage Divider (1/3)



Voltage divider for different voltages and currents from the source  $V_T$ .



## Design of a Loaded Voltage Divider (2/3)

- $I_1$  through  $R_1$  equals 30 mA
  - $I_2$  through  $R_2$  is  $36 + 30 = 66$  mA
  - $I_3$  through  $R_3$  is  $54 + 36 + 30 = 120$  mA
- 
- $V_1$  is 18 V to ground
  - $V_2$  is  $40 - 18 = 22$  V
  - $V_3$  is 100 V (Point D) – 40 = 60 V

## Design of a Loaded Voltage Divider (3/3)



- $R_1 = V_1/I_1 = 18 \text{ V}/30 \text{ mA} = 0.6 \text{ k}\Omega = 600 \text{ }\Omega$
- $R_2 = V_2/I_2 = 22 \text{ V}/66 \text{ mA} = 0.333 \text{ k}\Omega = 333 \text{ }\Omega$
- $R_3 = V_3/I_3 = 60 \text{ V}/120 \text{ mA} = 0.5 \text{ k}\Omega = 500 \text{ }\Omega$

NOTE: When these values are used for  $R_1$ ,  $R_2$ , and  $R_3$  and connected in a voltage divider across a source of 100 V, each load will have the specified voltage at its rated current.



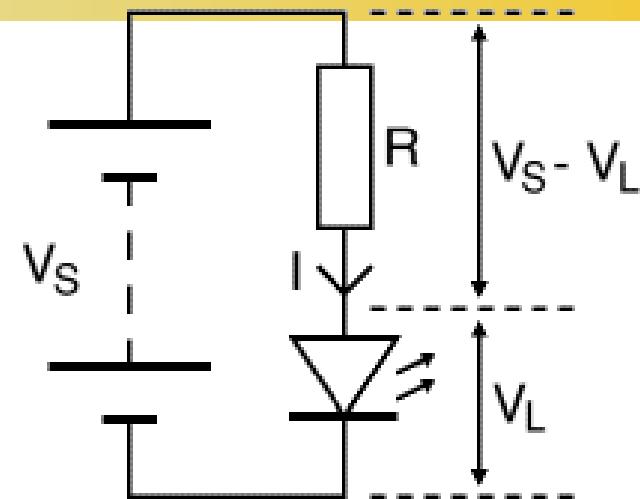
# LED



# Calculating an LED Resistor Value

An LED must have a resistor connected in series to limit the current through the LED. The resistor value, R is given by:

$$R = (V_s - V_L) / I$$



$V_s$  = supply voltage

$V_L$  = LED voltage (usually around 2V, but around 4V for blue and white LEDs)

$I$  = LED current (e.g. 20mA), this must be less than the maximum permitted

If the calculated value is not available, choose the nearest standard resistor value which is **greater**, to limit the current. Even greater resistor value will increase the battery life but this will make the LED less bright.

## For example

If the supply voltage  $V_s = 9V$ , and you have a red LED ( $V_L = 2V$ ), requiring a current  $I = 20mA = 0.020A$ ,  
 $R = (9V - 2V) / 0.02A = 350$ , so choose 390 (the nearest greater standard value).

# Connecting LEDs in Series

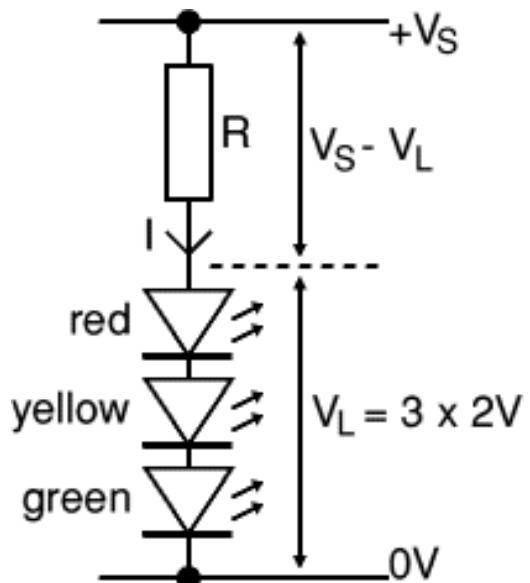


If you wish to have several LEDs on at the same time, connect them in series.

This prolongs battery life by lighting several LEDs with the same current as just one LED.

The power supply must have sufficient voltage to provide about 2V for each LED (4V for blue and white) plus at least another 2V for the resistor.

To work out a value for the resistor you must add up all the LED voltages and use this for  $V_L$ .



# Connecting LEDs in Series

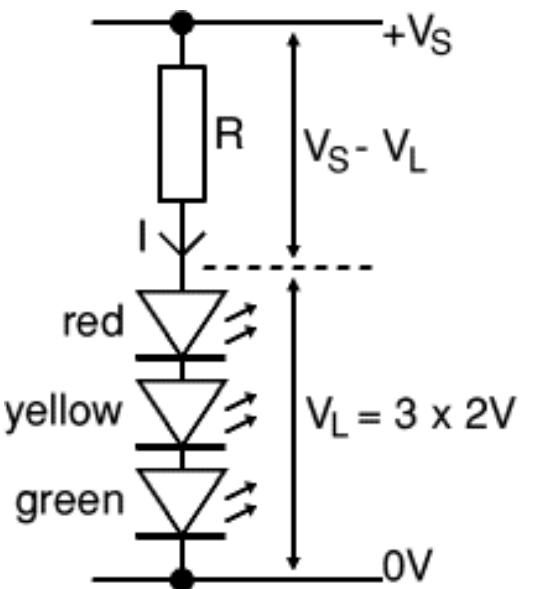


## Example

A red, a yellow and a green LED in series need a supply voltage of at least  $3 \times 2V + 2V = 8V$ ,

so choose a 9V battery.

Adjust the resistor R to have current  $I=15\text{ mA}$ .



# Connecting LEDs in Series



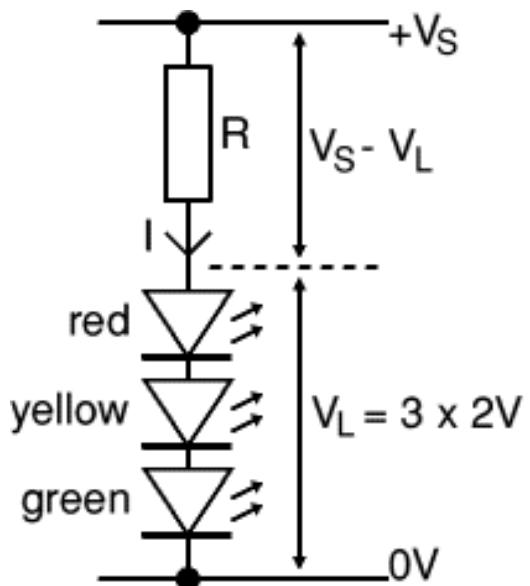
## Example

A red, a yellow and a green LED in series need a supply voltage of at least  $3 \times 2V + 2V = 8V$ , so choose a 9V battery. Adjust the resistor R to have current  $I=15\text{ mA}$ .

$V_L = 2V + 2V + 2V = 6V$  (the three LED voltages added up).

If the supply voltage  $V_S$  is 9V and the current I must be  $15\text{mA} = 0.015\text{A}$ ,

Resistor  $R = (V_S - V_L) / I = (9 - 6) / 0.015 = 3 / 0.015 = 200$ , so choose  $R = 220\Omega$  (the nearest standard value which is greater).



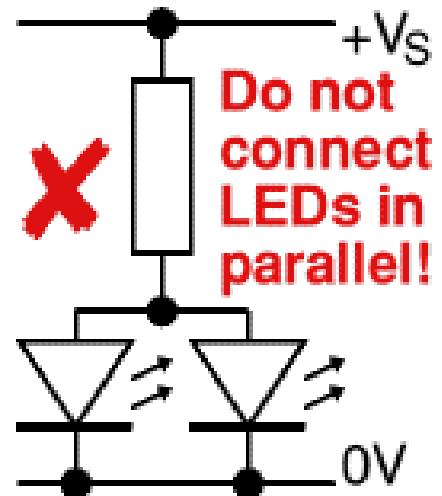
# Avoid Connecting LEDs in Parallel



Connecting several LEDs in parallel with just one resistor shared between them is a bad idea.

If the LEDs require slightly different voltages **only the lowest voltage LED will light** and it may be destroyed by the larger current flowing through it.

Although identical LEDs can be successfully connected in parallel with one resistor this rarely offers any useful benefit because resistors are very cheap and the current used is the same as connecting the LEDs individually.

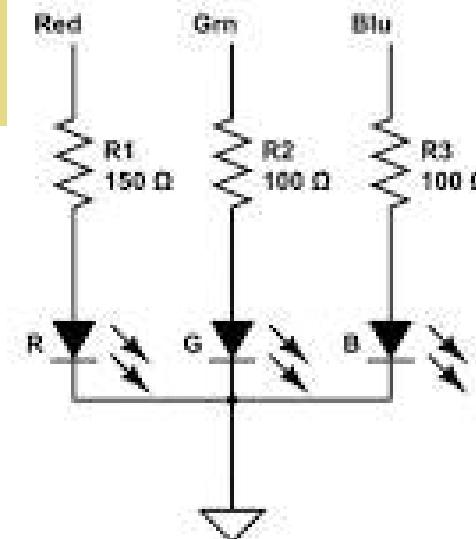
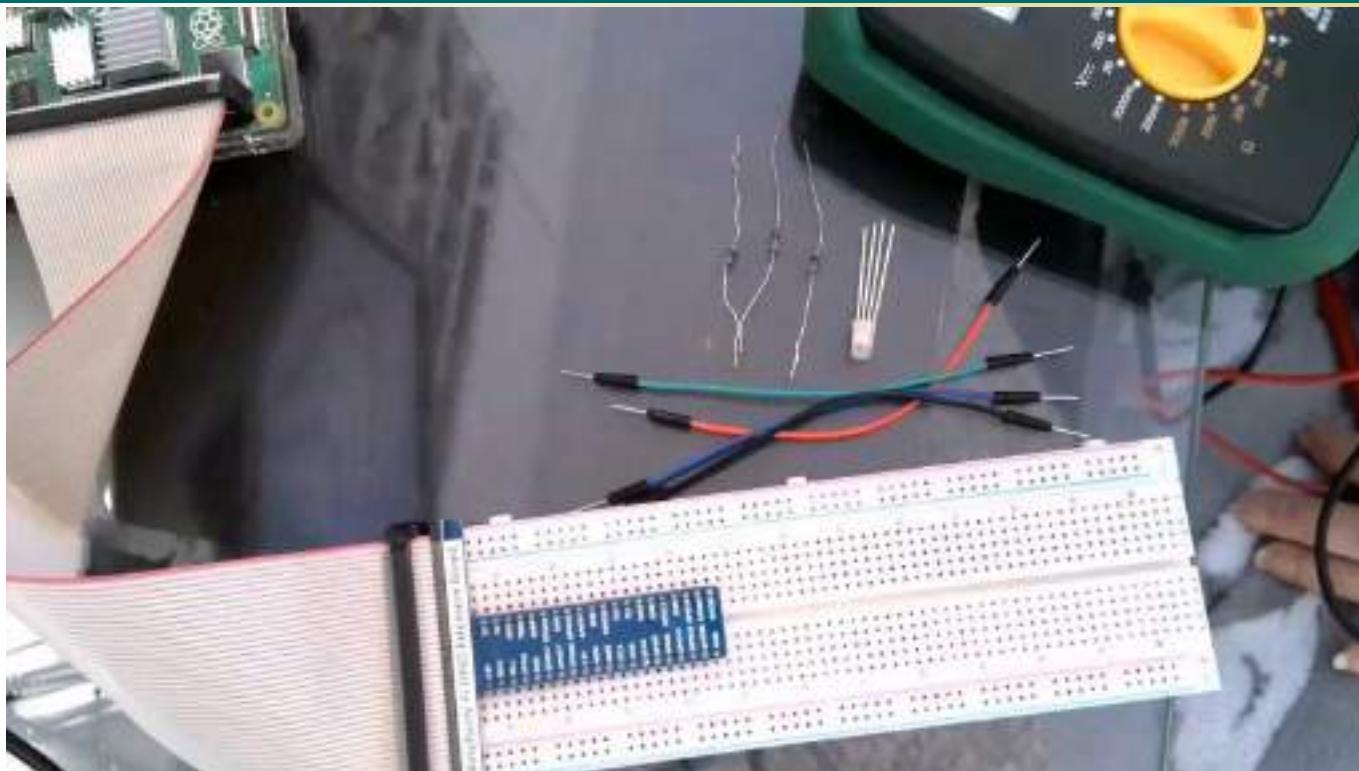


It has a risk even you recompute the new value of R!  
Therefore, if LEDs are in parallel, each one should have its own resistor.

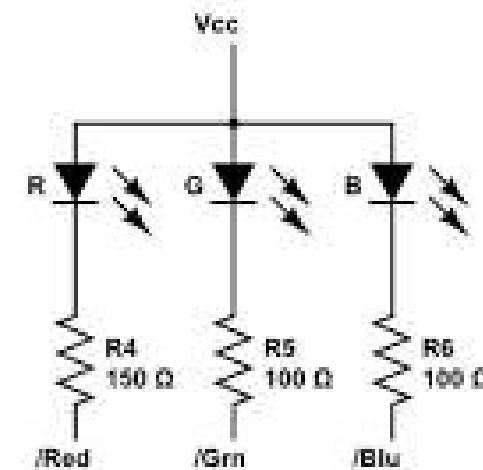
# Avoid Connecting LEDs in Parallel



# RGB LED



Common Cathode Circuit



Common Anode Circuit





- Mitchel E. Schultz, “Grob’s Basic Electronics”, ed. 12<sup>th</sup>, McGraw-Hill, 2011.
- Paul Scherz and Simon Monk, “Practical Electronics for Inventors”, ed. 4<sup>th</sup>, McGraw-Hill 2016.
- Simon Monk, Electronics Cookbook, O’Reilly, 2017.
- <https://hamblen.ece.gatech.edu/489X/>



# **ITCS447**

## **Lecture 3**

**IoT Systems [CISCO]**  
**Cyber Physical Systems**  
**Embedded and Autonomous Systems**  
**RTOS and ROS**

**Asst. Prof. Dr. Thitinan Tantidham**



ห้ามมิให้นักศึกษาทำข้า ดัดแปลง หรือใช้ประโยชน์จากการอันมีลิขสิทธิ์ปราการอยู่ในระบบการศึกษาอิเล็กทรอนิกส์ (E-Learning) ของมหาวิทยาลัย ไม่ว่าจะหั้งหมดหรือบางส่วนโดยไม่ได้รับอนุญาตจากมหาวิทยาลัย นอกเหนือจากการศึกษาส่วนบุคคล ทั้งนี้ การทำข้า ดัดแปลง หรือเผยแพร่ต่อสาธารณะชื่องานอันมีลิขสิทธิ์ จะมีโทษปรับตั้งแต่ 20,000 บาท ถึง 200,000 บาท และหากเป็นการกระทำเพื่อการค้า จะมีโทษจำคุกตั้งแต่ 6 เดือน ถึง 10 ปี หรือปรับตั้งแต่ 100,000 บาท ถึง 800,000 บาท หรือทั้งจำทั้งปรับ

Unless allowed by Mahidol University, the User shall not copy, modify, or exploit in part or in whole of the copyrighted materials on the Platform, other than for your own individual study. The copy, modify or communication to public of the copyrighted materials shall be inflicted with a fine from 20,000 Baht up to 200,000 Baht. If the offence is committed with the commercial purpose, the offender shall be inflicted with imprisonment for a term from 6 months up to 10 years or a fine from 100,000 Baht up to 800,000 Baht or both imprisonment and fine.



# Outline

- IoT Systems [CISCO]
- Cyber-Physical Systems
- Autonomous Systems
- Embedded Systems
- Real Time Operating Systems
- Robot Operating Systems
- References



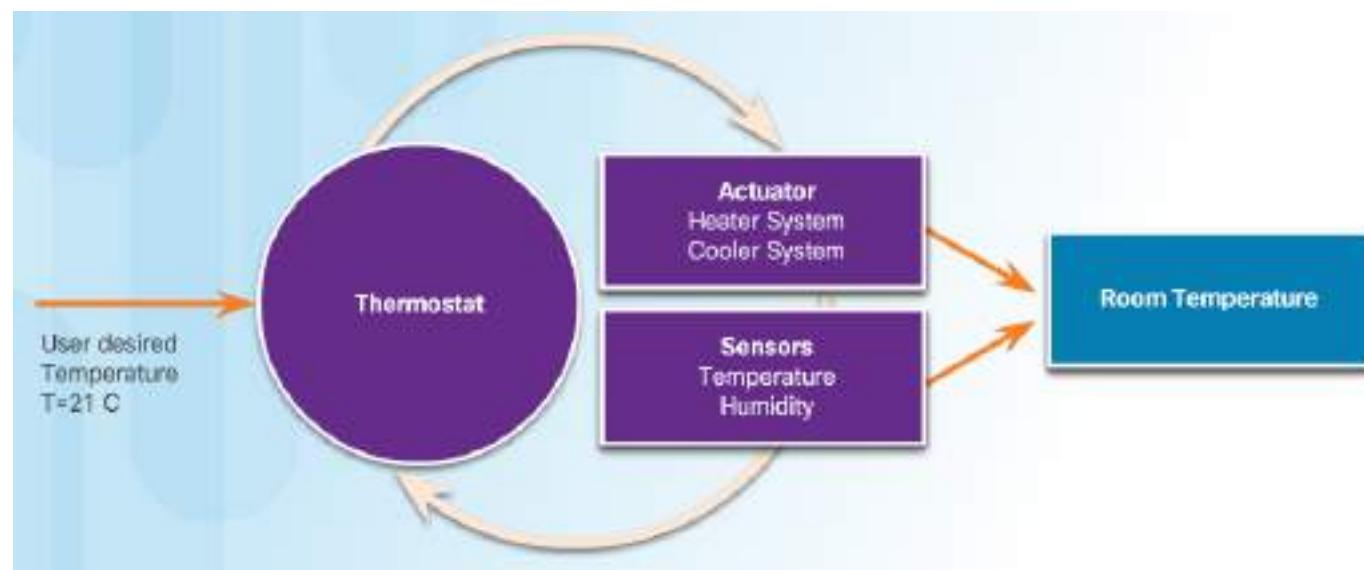


- The Presence of IoT in Today's World
  - The IoT is all around us.
  - The IoT helps individuals to improve quality of life.
  - The IoT also helps industries to become more efficient.
- Cisco IoT Solutions
  - The rapid IoT growth has introduced new challenges.
  - Cisco IoT System reduces the complexities of digitization.
  - **Six Pillars of the Cisco IoT System are:**
    - ✓ Network Connectivity
    - ✓ Fog Computing
    - ✓ Cybersecurity and Physical Security
    - ✓ Data Analytics
    - ✓ Management and Automation
    - ✓ Application Enablement Platform



## Building Blocks of an IoT System (1/2)

- Overview of a Controlled System
  - Feedback loops are used to provide real-time information to its controller based on current behavior.
  - In a closed loop, feedback is continuously being received by the controller from its sensors.
  - The controller continuously analyzes and processes information, and use actuators to modify conditions.
- Sensors
  - A sensor is a device that can be used to measure a physical property by detecting some type of information from the physical world.
  - A sensor may be connected to a controller either directly or remotely.





## Building Blocks of an IoT System (2/2)

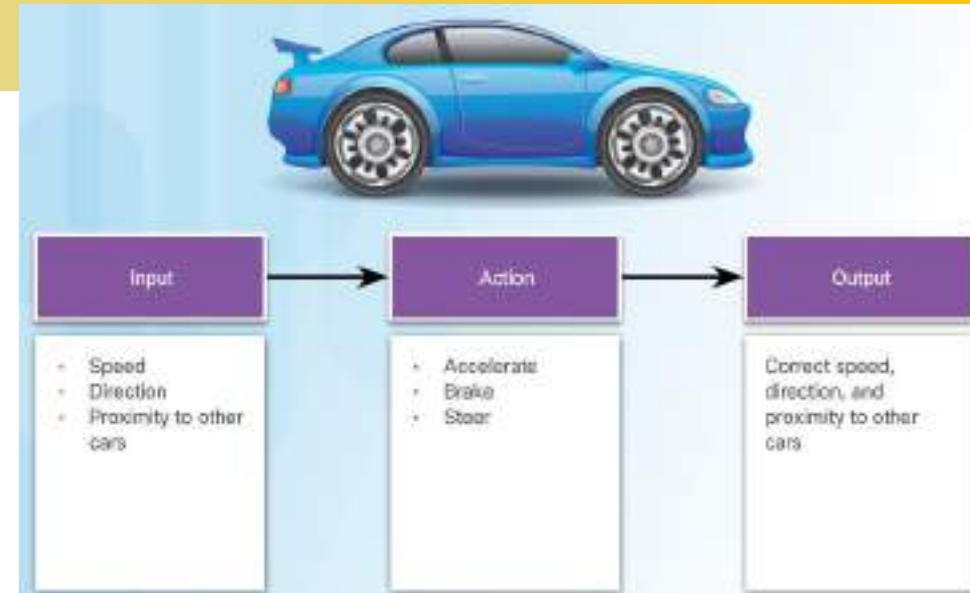
- Actuators
  - An actuator is a basic motor that can be used to control a system.
  - Can be hydraulic, electric or pneumatic.
  - can be responsible for transforming an electrical signal into physical output.
- Controllers
  - Responsible for collecting data from sensors and providing network connectivity.
  - Controllers may have the ability to make immediate decisions.
  - May also send data to remote and more powerful computer for analysis.
- IoT Process Flow
  - A simple IoT system include sensors connecting, through a wireless or wired connection, to actuators or controllers.
  - Some devices can have more than one function.





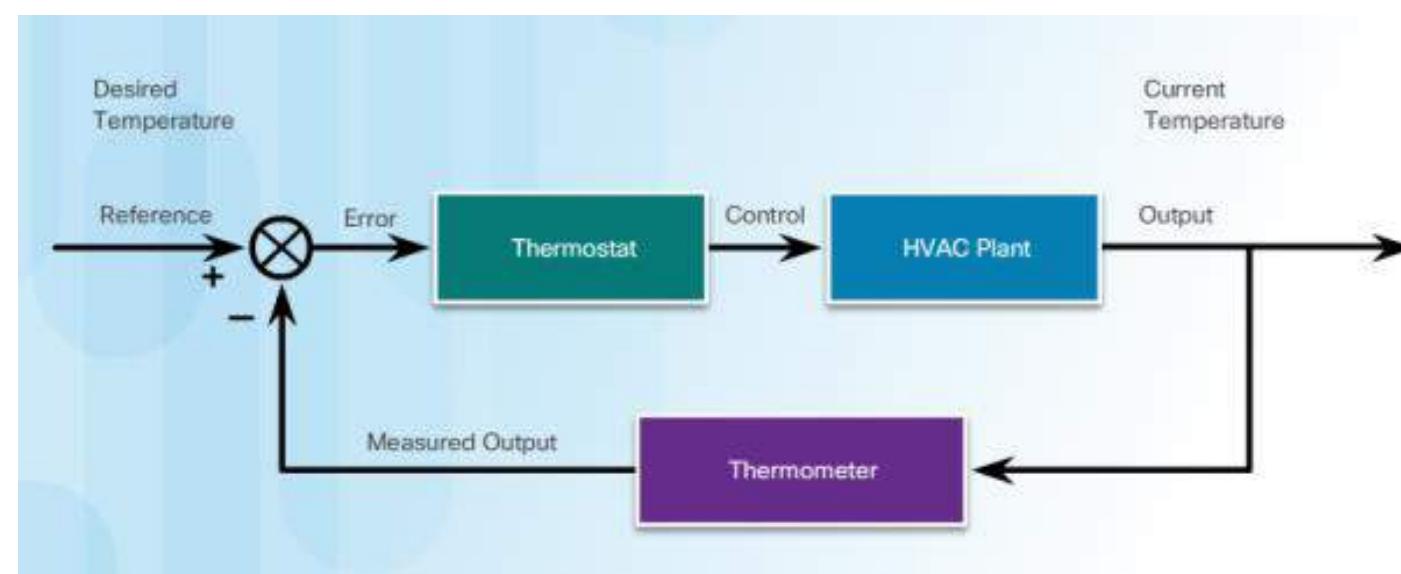
## Processes in Controlled Systems (1/3)

- Processes
  - A process is a series of steps or actions taken to achieve a desired result by the consumer of the process.
- Feedback
  - Feedback is when the output of a process affects the input.
  - Feedback is often referred to as a feedback loop.
  - Feedback loops can be positive or negative.
- Control Systems
  - Includes a controller that uses inputs and outputs to manage and regulate the behavior of the system in an attempt to achieve a desired state.
    - ✓ The controlled portion of the system is often called the plant.
    - ✓ Choosing the adjustments to apply to a plant to achieve a desired output is called **control theory**.
    - ✓ Control theory is applied to many systems, including driving a car.



## Processes in Controlled Systems (2/3)

- Open-Loop Control Systems
  - Open-loop control systems do not use feedback.
  - The plant performs a predetermined action without any verification of the desired results.
  - Open-loop control systems are often used for simple processes.
- Closed-Loop Control Systems
  - A closed-loop control system uses feedback to determine whether the collected output is the desired output.
  - The result is then fed back into a controller to adjust the plant for the next iteration of output, and the process repeats.



## Processes in Controlled Systems (3/3)

- Closed-Loop Controllers

- There are many types of closed-loop controllers:
  - Proportional controllers (P): based on the difference between the measured output and the desired output.
  - Integral controllers (PI): use historical data to measure how long the system has deviated from the desired output.
  - Proportional, Integral and Derivative controllers (PID): include data about how quickly the system is approaching the desired output.
- ✓ PID controller is an efficient way to implement feedback control.
- ✓ The Arduino and Raspberry Pi devices can be used to implement PID controllers.

- Interdependent Systems

- ✓ Most systems have many interdependent pieces contributing to and affecting the output.



## Definitions

- Cyber-Physical Systems (CPS) are integrations of computation, networking, and physical processes.
- CPS integrates the dynamics of the physical processes with those of the software and networking, providing abstractions and modeling, design, and analysis techniques for the integrated whole.
- Note: The technology builds on the older (but still very young) discipline of embedded systems, computers and software embedded in devices whose principle mission is not computation, such as cars, toys, medical devices, and scientific instruments.

Ref:

<https://ptolemy.berkeley.edu/projects/cps/>

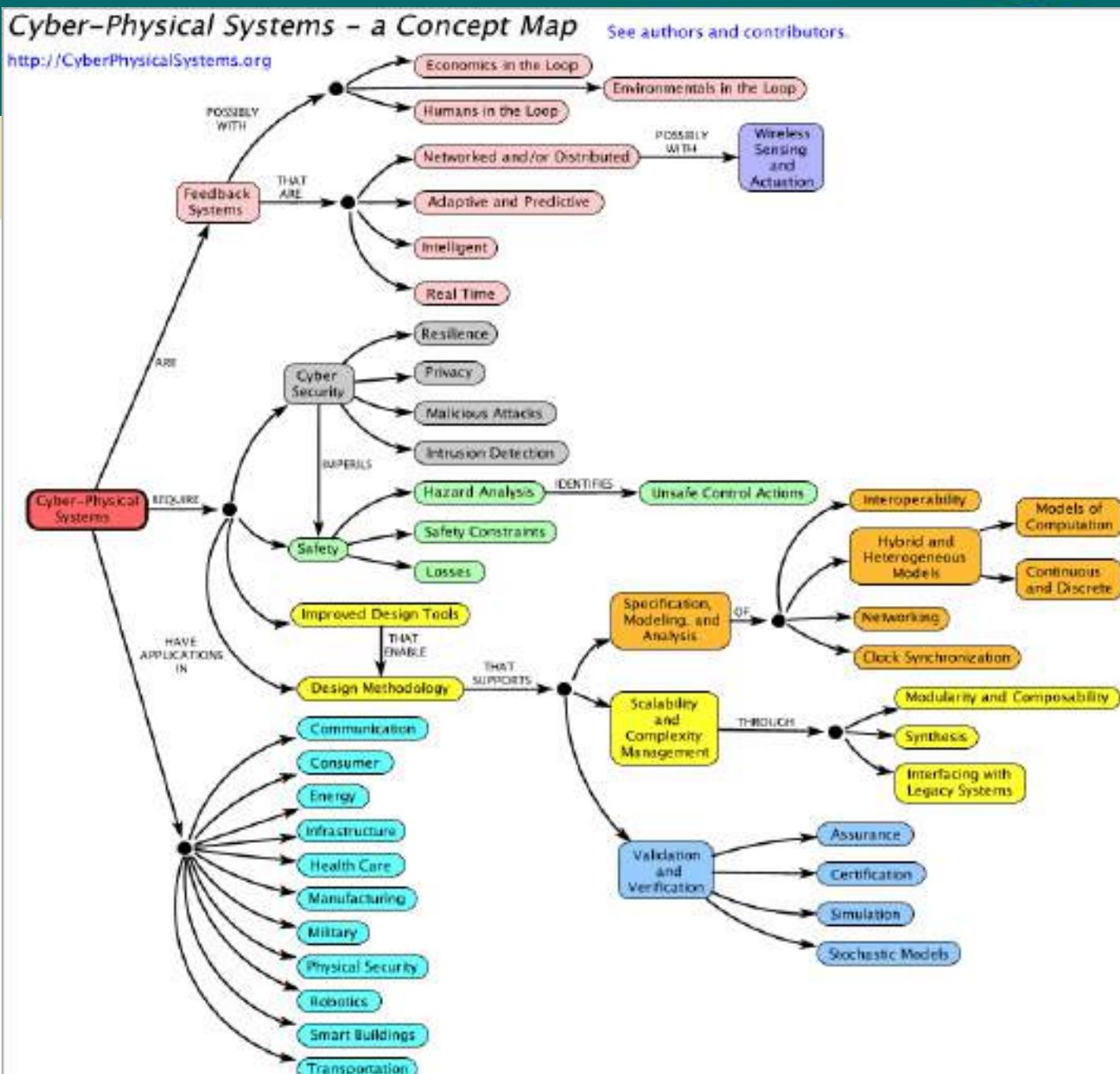


## Definitions



# Cyber-Physical Systems

## CPS Model



Ref:

<https://ptolemy.berkeley.edu/projects/cps/>



## Online Talks about CPS by Prof. Edward, A. Lee

- [Models of Timed Systems](#), Mälardalens University, Västerås, Sweden, Sep. 26, 2018.  
[https://play.mdh.se/media/t/0\\_tvle844g](https://play.mdh.se/media/t/0_tvle844g)
- [Resurrecting Laplace's Demon: The Case for Deterministic Models](#), CS Distinguished Speaker Series, Univ. of Houston, Nov. 8, 2017. <https://www.youtube.com/watch?v=jGQKj-USUOc>
- [What Does 'Real Time' Mean?](#) Simons Institute, UC Berkeley, June 28, 2016.  
<https://www.youtube.com/watch?v=G8aofqZxVZY>
- [Cyber-Physical Systems: A Fundamental Intellectual Challenge](#), College de France, Paris, France, Dec. 11, 2013.  
<https://www.college-de-france.fr/site/gerard-berry/guestlecturer-2013-12-11-17h00.htm>

Ref:

<https://ptolemy.berkeley.edu/projects/cps/>

<https://ptolemy.berkeley.edu/~eal/>

<https://www.youtube.com/channel/UCYywSsaTcxhIRE1fwp5mhYg>

<https://blog.engineering.vanderbilt.edu/what-is-the-difference-between-cps-and-iot>

[https://vimeo.com/703370628?embedded=true&source=vimeo\\_logo&owner=4730653](https://vimeo.com/703370628?embedded=true&source=vimeo_logo&owner=4730653)

<https://cacm.acm.org/magazines/2022/5/260355-explorations-in-cyber-physical-systems-education/fulltext>

## Online Talks about CPS by Prof. Edward, A. Lee

- (Network) Autonomous systems are independent networks controlled by different organizations, each running an interior gateway protocol of their choice.
- Autonomous system does not even require the controller to realize the proper decision but can react on environmental changes.
- Examples:
  - Self-driving cars and some robots, operate without human intervention.
  - An autonomous vacuum cleaner can clean our house while it detects that we aren't present at home, and the floor needs to be cleaned.
  - The fridge can order our favourit beverage once it detects that the last bottle is almost empty

# Autonomous System





## Overview





## Overview





## Definitions

- **Embedded systems** are an **Information processing system embedded into a larger product.** [Peter Marwedel, TU Dortmund]
- **Embedded software** is software integrated with physical processes. The technical problem is managing time and concurrency in computational system with feedback loops. [Edward A. Lee, UC Berkeley]
- **Embedded systems** are **engineering artifact** involving computation that is subject to **physical constraints.**
  - These **constraints** affect available processor speeds, **power**, and hardware failure rates.
  - Many embedded systems feature **very tight power budget** (an order of some **Watts**) while **ultra low-power ones** (e.g. **wearable systems**) only a **few milli watts.**

Table 1. Comparison of baseline processors and the state-of-the-art cores.

Core Name	Pipe No.	Hard Mul Div	Technology	Power Supply	Area (KGE)	Perf (CortMark @MHz)	Power (uW /MHz)
TinyCore	/	Y	40 nm	1.1 V	19.3	2.88	2.12
LittleCore	2	Y	40 nm	1.1 V	19.8	3.44	2.29
PipeCoreA	4	Y	40 nm	1.1 V	21.2	2.75	3.63
PipeCoreG	4	Y	40 nm	1.1 V	24.8	2.91	5.32
micro-riscy	2	N	65 nm	1.2 V	11.6	0.91	2.33
zero-riscy	2	Y	65 nm	1.2 V	18.9	2.44	2.81
riscy	4	Y	65 nm	1.2 V	40.7	3.19	6.98
Cortex-M0+	2	Y	40 nm	1.1 V	12.5	2.46	3.8
Cortex-M3	3	Y	40 nm	1.1 V	37.9	3.34	11
Cortex-M4	3	Y	40 nm	1.1 V	53	3.42	12.26

Ref:

W. Ma, Q. Cheng, Y. Gao, L. Xu, and N. Yu, "An Ultra-Low-Power Embedded Processor with Variable Micro-Architecture, *Micromachines*., MDPI, 2021.  
<https://users.ece.utexas.edu/~valvano/>

## Characteristics (1/2)

- **Job Specific:** designed for a specific task, has strong correlation with the application. e.g. ECG can only be used for cardio signal acquisition and processing. Complete redesign is needed if using the microcontroller for other purpose.
- **Invisibility:** it is a part of the whole system, people mostly will only focus on the system function, performance and usage instead of the embedded system inside it.
- **Real time:** provide response within a limited period of time. strong: us  $\sim$ ms; typical: ms  $\sim$ s; weak: $\sim$ s
- **Limited resources:** the available hardware resources of embedded system is limited due to its low-cost, small form factor, low-power, etc.

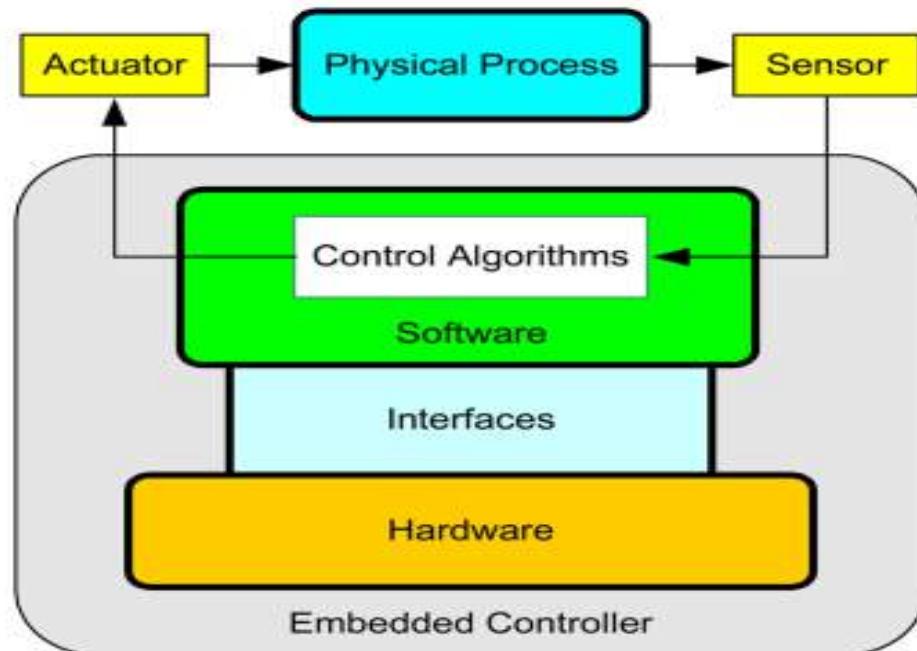
## Characteristics (2/2)

- An embedded system is a **specific purpose system** in which the computer is completely encapsulated by the device it controls.
- An embedded system performs **predefined tasks** usually with very specific requirements.
- A combination of **hardware and software**, and perhaps additional mechanical or other parts, designed to perform a dedicated function.
- They can range from having **no user interface** to having **complex graphical(GUI)** interface like mobile devices.
- Traditional mobile phones, Music players, Digital cameras, TVs are **very common embedded systems in our life.**
- An embedded system might contain **one or more programmable chips** such as microcontroller ( $\mu$ C, uc, or MCU), microprocessor or digital signal processor.
- The word “embedded” indicates that computer unit (**e.g. microcontroller**) is fully surrounded by the device it controls, and is invisible to the user of the device.

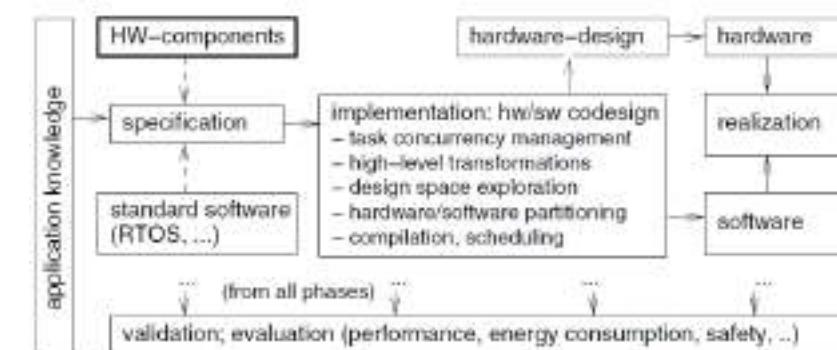


## Layer Architecture of Embedded System

An embedded system is a tightly coupled Hardware (HW) + Software (SW) system to perform a dedicated system.



Simplified Design Information Flow



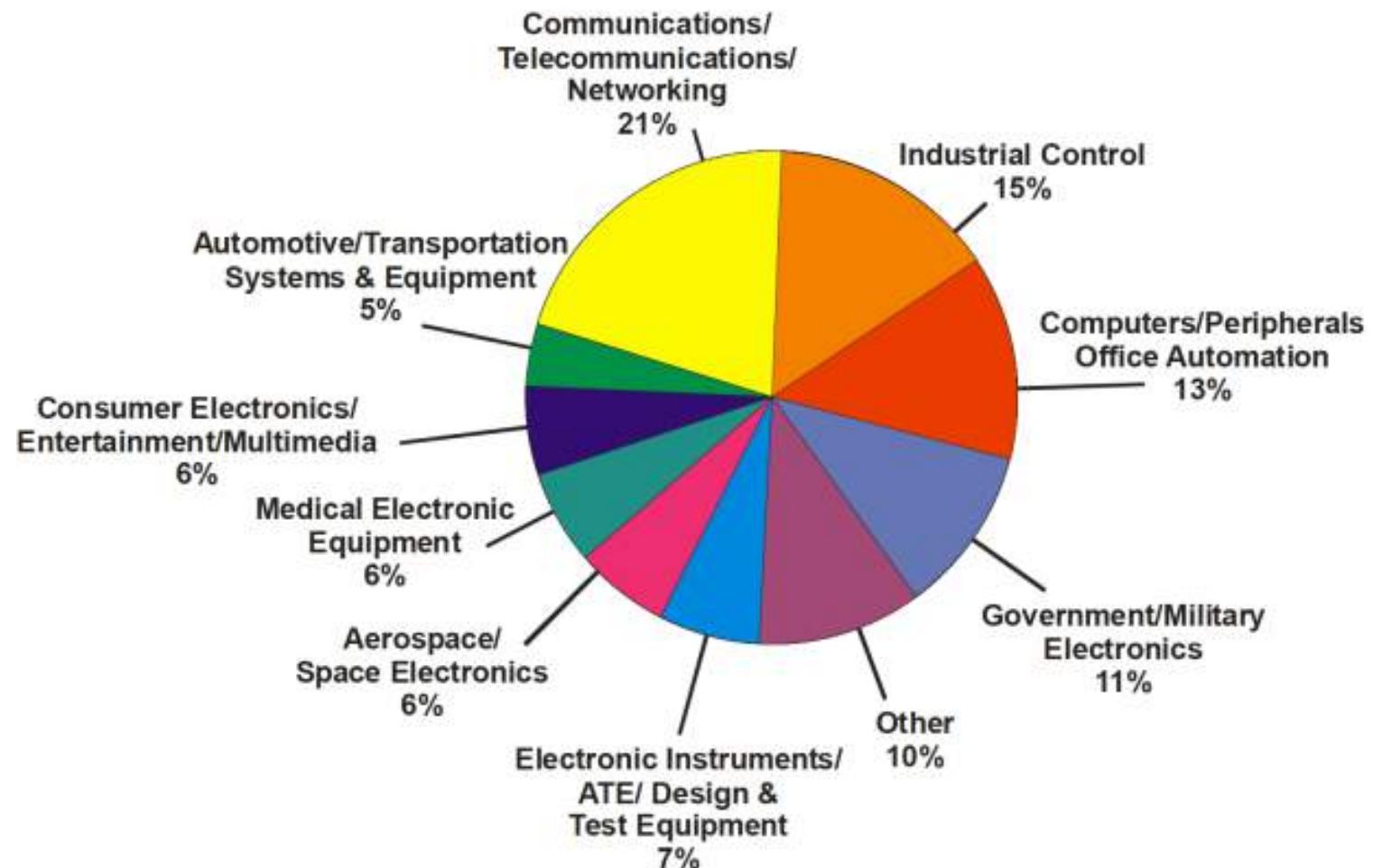
Automotive Electronics



Average car has about 50 MCUs.

S-class has 63; a 1999 BMW 7-series has 65 [2009]

## Embedded System Applications





## Evolution

- **Year 1833-** Michael Faraday described electronic conduction increasing with temperature in silver sulfide crystals.
- Year 1874- Ferdinand Braun noted that current flowed freely only in one direction at the contact between the metal point and galena crystal.
- **Year 1961-** Charles Stark Draper, in order to reduce the size and weight of the Appollo Guidance Computer, developed and Integrated Circuit(IC). It became the first computer to use Ic and it helped in collecting real-time flight data.
- **Year 1965-** Autonetics developed the D-17B, missile guidance system in Minuteman 1. It marked the first mass production of the Embedded Systems.
- **Year 1968-** The Volkswagen 1600 used a Microprocessor to control its electronic fuel injection system. It was the first Embedded System developed for a vehicle and marked an important contribution towards the Evolution of Embedded Systems.
- **Year 1970-** Texas Instruments developed the first Microcontroller. Also, in 1971, Intel developed the first commercially available processor, the 4004. It was a 4-bit processor and was designed for use in calculators and small electronics.
- **year 1987-** Wind River released the first embedded operating system, the real-time VxWorks. Also, in 1996, Microsoft's Windows Embedded CE was released.
- **Late 1990s-** Embedded Linux products began to become popular and today Linux is used in almost all embedded devices.

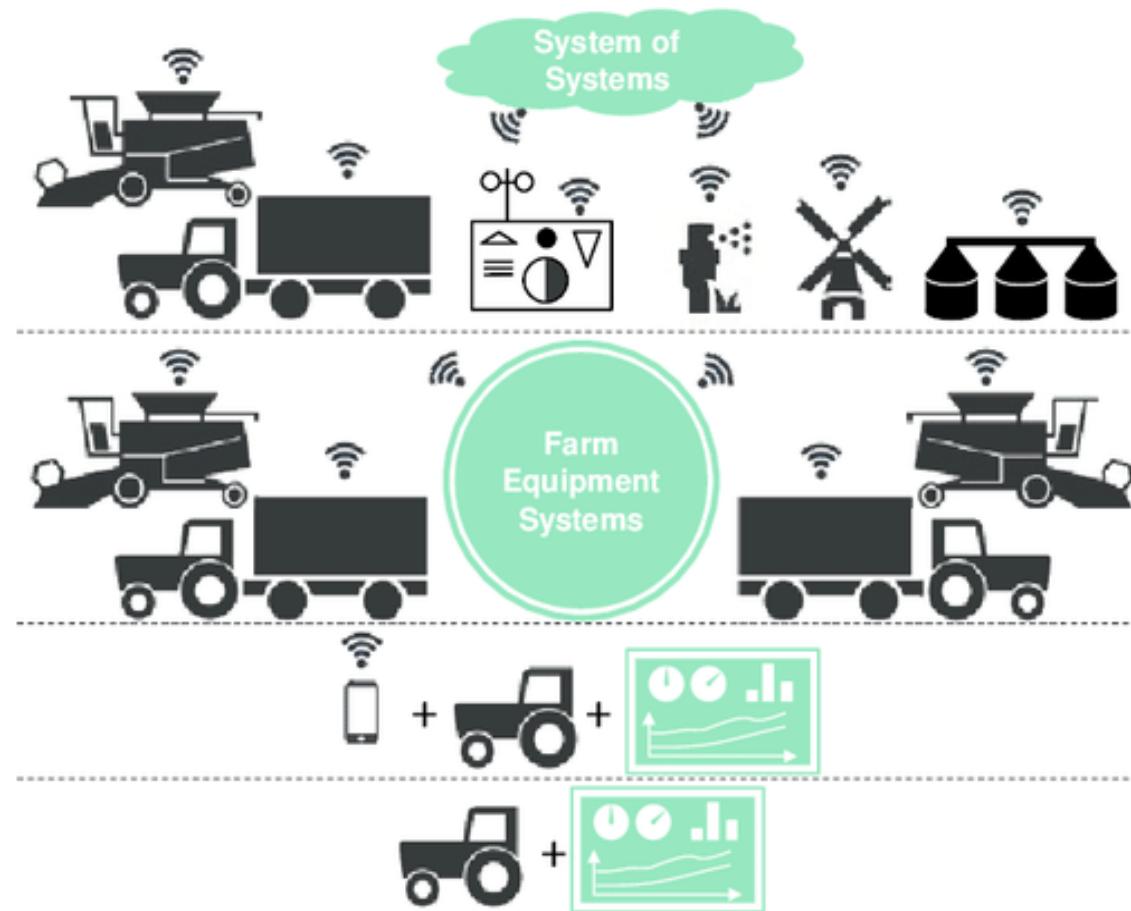
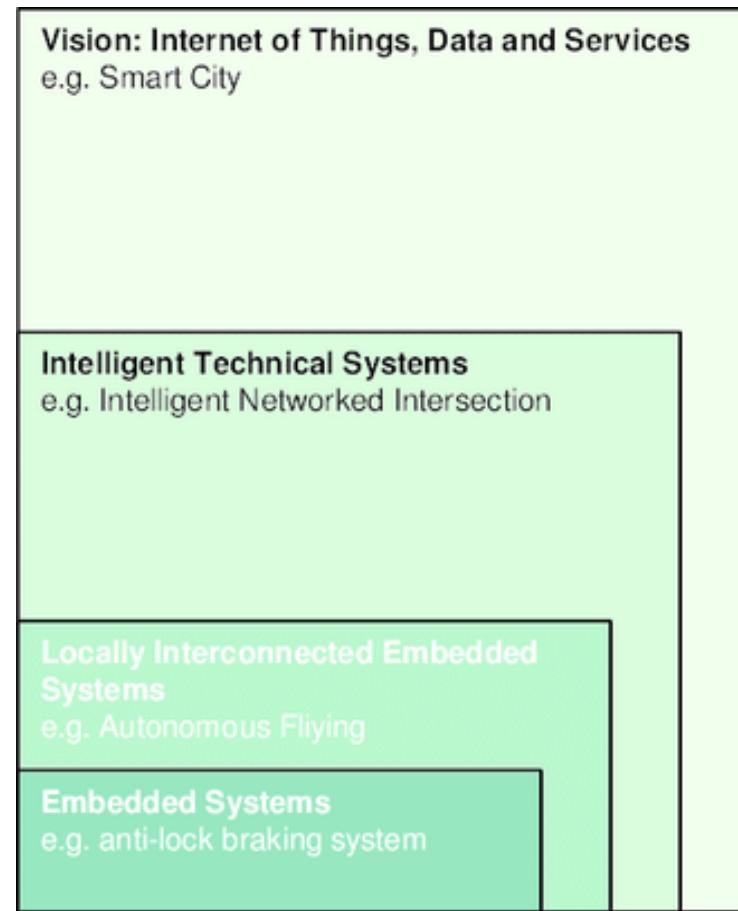
### Note:

- All these significant contributions towards the Evolution of Embedded Systems have contributed to modern-day gadgets and technologies all of which are dependent on embedded systems.
- Moore's Law (PhD Gordon Moore, 1965) describes the increased capacity in ICT, which becomes clear by watching trends like these over time: operations per second of a processor, available volatile memory for processing, non-volatile storage space, and interconnection speed in networks

### Ref

<https://whataftercollege.com/robotics-embedded-system/chronological-evolution-of-the-embedded-systems/>

## Evolution



D. Kliewe, et.al. "Model-based representation of protection measures as Solution Patterns", In Procedia Technology (26), Elsevier, 2016.

Geisberger, E.; Broy, M. (editors): agendaCPS, Integrierte Forschungsagenda Cyber-Physical Systems (acatech Studie). Springer Verlag, Berlin, 2012.

Porter M. E.; Heppelmann J.E.: How Smart, Connected Products are Transforming Com-petition. In: Harvard Business Review, Ausgabe November, 2014.

## Technologies

- Today's embedded systems can be divided into :
  - 1) Real-Time Embedded System
  - 2) Stand-alone Embedded System
  - 3) Networked Embedded System
  - 4) Mobile embedded System
  - 5) Small-scale Embedded System
  - 6) Medium-Scale Embedded System
  - 7) Sophisticated or Complex Embedded system
- The latest technology in Embedded Systems in the present time are:
  - 1) Artificial intelligence and Machine Learning
  - 2) Deep Learning
  - 3) Embedded Security
  - 4) Cloud Connectivity
  - 5) Augmented Reality and Virtual Reality

Ref

<https://whataftercollege.com/robotics-embedded-system/chronological-evolution-of-the-embedded-systems/>



- These are computer systems that monitor, control, and respond as per need to a change or event in the external event. It has a **low latency response to an input event**. These systems work with strict time-constraints and can provide guaranteed **worst-case response time to critical events** and **acceptable average-case response time to non-critical events**
- **Real-Time Embedded Systems can be divided into 2 types:**
  - 1) Hard Real-Time Embedded System
  - 2) Soft Real-Time Embedded System
- **Hard Real-Time Embedded System:** The primary function of these types of systems is **to ensure that all critical functions/processes are completed within the stipulated time frame** and missing a deadline is considered a **system failure**. All delays in the systems are strictly time-bound. **Data is stored in short-term or read-only memory as there is no secondary memory.** These types of systems are used in various missiles and airplanes.
- **Soft Real-Time System:** Even though the basic functionality is to ensure that critical processes have to be completed within a given time frame, **they are less constrictive than Hard-Time Systems**. Soft Real-Time tries to reach a deadline but **does not consider it a system failure** if one or two deadlines are missed. They are a little bit more flexible than Hard Real-time Systems. **They are used in various areas like multimedia, scientific projects, seismic sensors, etc.**
- **Three Major Challenges with Real-Time Embedded Systems**
  - 1) Managing software engineering Organizations
  - 2) Ensuring the development of a performance relevant architecture
  - 3) Finding suitable tools

#### Applications of Real-Time Embedded Systems

- Vehicle control system for automobiles, ships.
- Military operations
- Multimedia systems that provide graphics, video, and audio
- Space operations like spaceship launch, monitoring and space station control
- Medical treatments for radiation therapy, heart operations, etc.

Ref

<https://whataftercollege.com/robotics-embedded-system/chronological-evolution-of-the-embedded-systems/>

## System On Chip (SOC) Technologies

Component	Main application	Main characteristics
Embedded RISC processor (ex. ARM)	General computing	Low performance, high flexibility, low cost
Microcontroller	Control-dominated computing	High performance, good flexibility, high cost
DSP: digital signal processor	Data-dominated computing	High performance, good flexibility, high cost
ASIC: application-specific integrated circuit	Specific computing	Very high performance, low flexibility, very high cost
ASIP: application-specific instruction set processor	Specific application domain	High performance, good flexibility, high cost
FPGA	Reconfigurable computing	Good performance, high flexibility, high cost

Ref:

<https://cdn.intechopen.com/pdfs/58179.pdf>

## System On Chip (SOC) Architectures

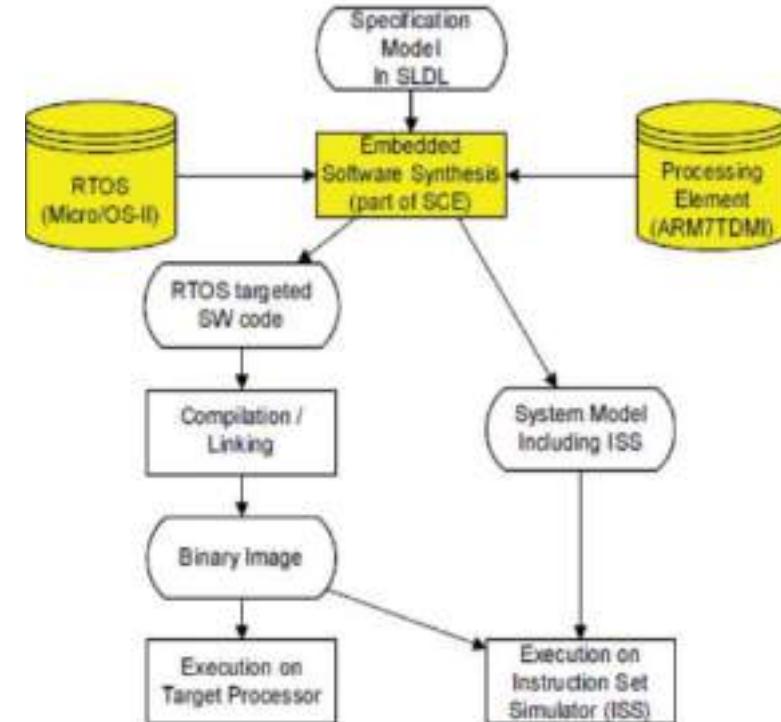
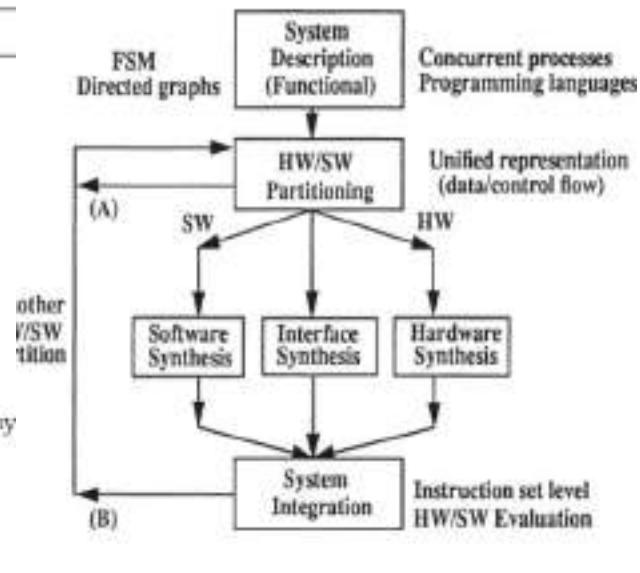
SOC class	Main characteristics
TSOC: traditional SOC	One central processor (master) with many hardware accelerators (slaves)
MPSOC: multiprocessor SOC	Multiprocessors architecture
SSOC: software-oriented SOC	SOC where software implementation is the prominent part; the architecture is mainly composed of ISA processors
HSOC: hardware-oriented SOC	SOC where hardware implementation is the prominent part; the architecture is mainly composed of ASICs
RSOC: reconfigurable SOC	FPGA-based
NOC: network on chip	A microcommunication network
PPSOC: plug and play SOC	SOC with IP reuse
PNOc: photonic NOC	Photonic technology
WNOC: wireless NOC	A combination between wired and wireless communications
QSOC: quantum SOC	SOC that contains all the components needed for a quantum information processor
CSOC: chaotic SOC	Chaotic computing-based

Ref:

<https://cdn.intechopen.com/pdfs/58179.pdf>

## Codesign Flow

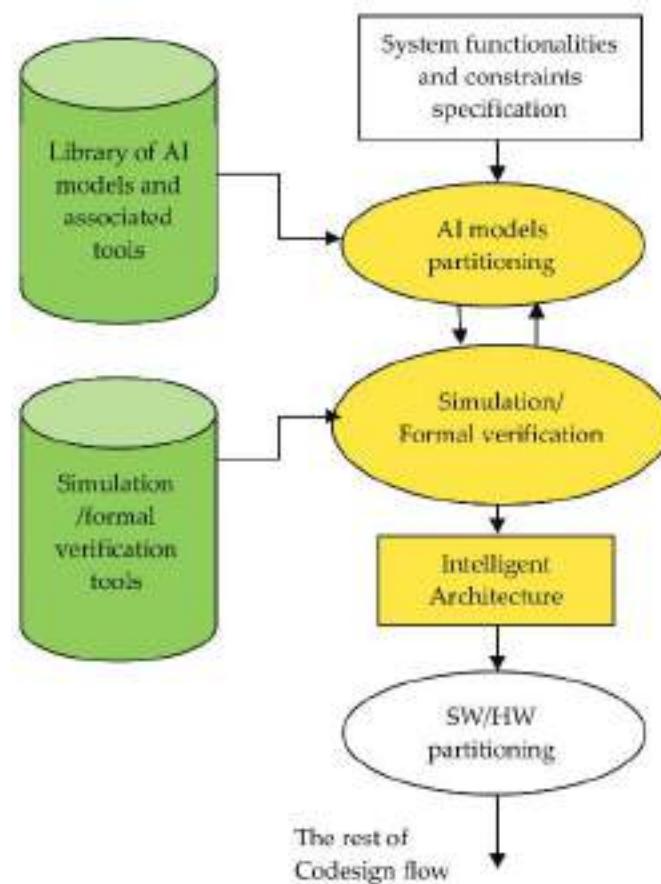
Model of computation	Application
DF: data flow	Data-oriented processing
SDF: synchronous DF	Data-oriented processing when the input/output size is known
KPN: Kahn process network	Deterministic data-oriented processing with infinite buffer
DE: discrete event	Discrete time processing
CT: continuous time	Continuous time processing (analog parts)
FSM: finite-state machine	Control-oriented sequential processing
DFSM: data path fsm	Control-/data-oriented processing
Statecharts	Control-oriented processing supporting concurrency and hierarchy
RS: reactive synchronous	Reactive systems with zero delay computing assumptions
Petri nets	Reactive systems (formal specification/verification)
Multi-MOC	Heterogeneous systems



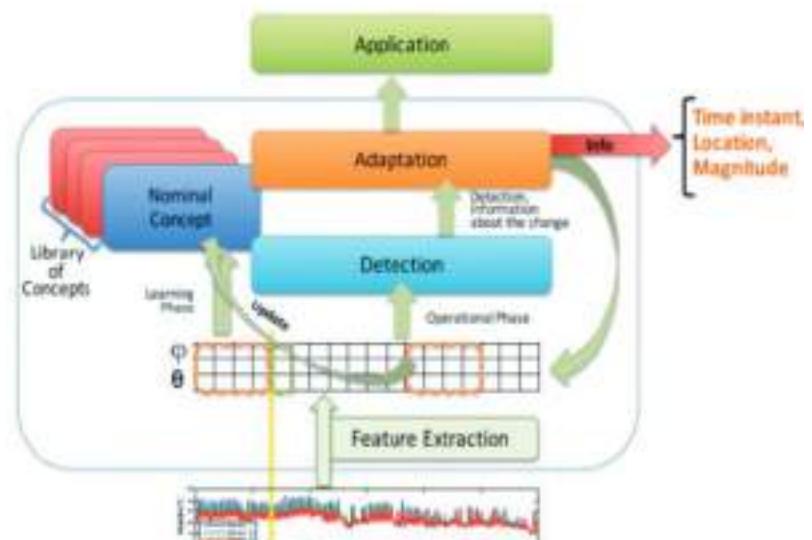
Ref:

<https://cdn.intechopen.com/pdfs/58179.pdf>

## Intelligent Architecture Based Intelligent Embedded Software Codesign



Embedded computing	Intelligent computing
Software and hardware are both first class	Software is first class
Resources constrained	Unlimited resources
Simple tasks	Very complex tasks
Small computing power	Significant computing power
Reactive	Cognitive
Low-level models and programming languages (Assembly, C)	High-level programming models beyond procedural paradigm
Static and completely specified environment	Dynamic and imprecise environments
Human intervention is weak	Human intervention is prominent



Ref:

<https://cdn.intechopen.com/pdfs/58179.pdf>



## Operating Systems

Operating System	Overview	Characteristics	Language	Open Source
Contiki	An open-source multitasking OS designed for wireless sensor network (WSN) and memory-efficient embedded systems network.	Modular structure, multithreading, event-driven.	C	✓
TinyOS	An open-source OS, intended for the low-power wireless devices	Monolithic structure, multithreading, event-driven, support for TOS threads.	NesC	✓
RIOT OS	Real time	Modular structure, multithreading	C and C++	✓
Mantis	An open-source operating system designed for WSN. It presents C API with Linux and Windows development environments.	Threads	C	✓
Nano-RK	This OS has a lightweight embedded resource kernel (RK) and networking support to be used in WSN.	Threads	C	✗
LiteOS	An open-source UNIX-like OS for WSN	Threads and events	LiteC++	✓
FreeRTOS	A real-time OS designed for embedded devices		C	✗
Linux		Monolithic structure, event-driven	C and C++	✓



## Introduction

- RTOS is an open source [operating system for embedded devices](#) developed by **RT-Thread**. It provides a standardized, friendly foundation for developers to program a variety of devices and includes a large number of useful libraries and toolkits to make the process easier.
- Like Linux, RTOS uses a modular approach, which makes it easy to extend. Packages enable developers to use RTOS for any device they want to target. One of RTOS's most popular extensions is the AT device package, which includes porting files and sample code for different AT (Command) devices (i.e., modems).
  - AT commands were originally a protocol to control old dial-up modems. As modem technology moved on to higher bandwidths, it remained useful to have a light and efficient protocol for device control, and major mobile phone manufacturers jointly developed a set of AT commands to control the GSM module on mobile phones.
  - Today, the AT protocol is still common in networked communication, and there are many devices, including WiFi, Bluetooth, and 4G, that accept AT commands.
  - If you're creating purpose-built appliances for edge computing input, monitoring, or the Internet of Things (IoT), some of the AT devices supported by RTOS that you may encounter include ESP8266, ESP32, M26, MC20, RW007, MW31, SIM800C, W60X, SIM76XX, A9/A9G, BC26, AIR720, ME3616, M 6315, BC28, and EC200X.

<https://www.guru99.com/real-time-operating-system.html>

# Difference between GPOS and RTOS



## General-Purpose Operating System (GPOS)

It used for desktop PC and laptop.

### Process-based Scheduling.

Interrupt latency is not considered as important as in RTOS.

No priority inversion mechanism is present in the system.

Kernel's operation may or may not be preempted.

Priority inversion remain unnoticed

## Real-Time Operating System (RTOS)

It is only applied to the embedded application.

Time-based scheduling used like round-robin scheduling.

Interrupt lag is minimal, which is measured in a few microseconds.

The priority inversion mechanism is current. So it can not modify by the system.

Kernel's operation can be preempted.

No predictability guarantees

<https://www.guru99.com/real-time-operating-system.html>

# Real Time Operating System (RTOS)







## Introduction

- ROS is an open-source, meta-operating system (middleware) for your robot.
  - Provides the services you would expect from an operating system, including hardware abstraction, low level device control, implementation of commonly-used functionality, message-passing between processes, and package management.
  - provides tools and libraries for obtaining, building, writing, and running code across multiple computers (robot software development)
- ROS is similar in some respects to 'robot frameworks,' such as Player, YARP, Orocros, CARMEN, Orca, MOOS, and Microsoft Robotics Studio).
- The ROS runtime "graph" is a peer-to-peer network of processes (potentially distributed across machines) that are loosely coupled using the ROS communication infrastructure.
- ROS implements several different styles of communication, including synchronous RPC-style communication over services, asynchronous streaming of data over topics, and storage of data on a Parameter Server.
- Originally developed in 2007 at the Stanford Artificial Intelligence Laboratory and development continued at Willow Garage
- Since 2013 managed by OSRF (Open Source Robotics Foundation)
- Note: ROS is not a real-time framework, though it is possible to integrate ROS with real-time code.



## Introduction

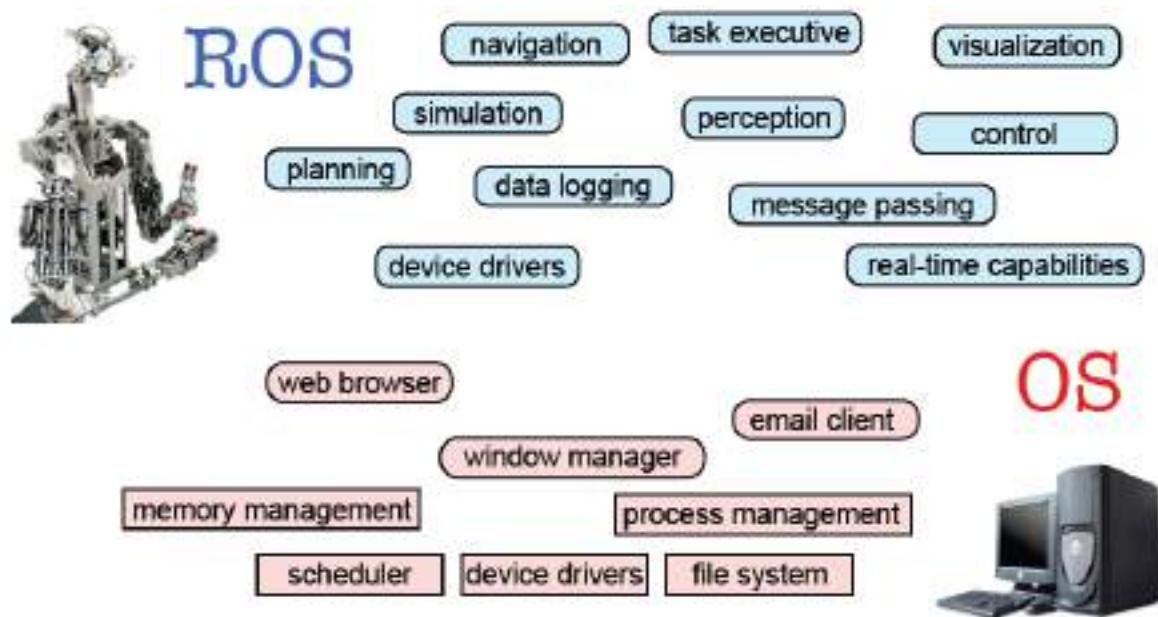




## Introduction

## ROS vs OS

- ROS provides standard operating system services such as
  - hardware abstraction, (USB drive)
  - low-level device control,
  - implementation of commonly-used functionality,
  - message-passing between processes, and
  - package management
  - provides tools and libraries for obtaining, building, writing, and running code across multiple computers





## ROS Philosophy

- **Peer to Peer**
  - ROS systems consist of numerous small computer programs which connect to each other and continuously exchange messages
- **Tools-based**
  - There are many small, generic programs that perform tasks such as visualization, logging, plotting data streams, etc.
- **Multi-Lingual**
  - ROS software modules can be written in any language for which a *client library has been written*. Currently client libraries exist for C++, Python, LISP, Java, JavaScript, MATLAB, Ruby, and more.
- **Thin**
  - The ROS conventions encourage contributors to create stand-alone libraries and then *wrap those libraries so they send and receive messages to/from other ROS modules*.
- **Free and open source**



## ROS References (ros.org)

- <http://wiki.ros.org/>
- Installation: <http://wiki.ros.org/ROS/Installation>
- Tutorials: <http://wiki.ros.org/ROS/Tutorials>
- ROS Tutorial Videos
  - <http://www.youtube.com/playlist?list=PLDC89965A56E6A8D6>
- ROS Cheat Sheet e.g.
  - <https://mirror.umd.edu/roswiki/attachments/de/ROScheatshheet.pdf>

<http://wiki.ros.org/>



<https://www.youtube.com/watch?v=xHw2WFe6GBs>

<https://www.youtube.com/channel/UC2aPsByptP3HXobjXgsXQsA>

## Logic Control

- Logic control is an essential part to develop an intelligence device
- Logic control can be developed by
  - Truth table
    - You only need to know the corresponding input/output relation
    - As there is **no memory**, only simple operations can be achieved
  - Finite State Machine (FSM) or State Machine
    - Decision is based on what it has done i.e. the system has memory (history). Therefore the performance can be more complex.
    - A sequential system contains state stored in memory elements internal to the system.
    - Its behavior depends both on the set of inputs supplied and on the contents of the internal memory, or state of the system.
    - Thus, the sequential system cannot be described with a truth table.

# Case Study: ESP32 & Arduino Programming



- Arduino ESP32 core is built over FreeRTOS. If you look into the `cores/esp32` folder, you will see a file called “`main.cpp`”.
  - Here you can see our Arduino program is just a task of FreeRTOS.
  - When the task is invoked it will invoke “`setup()`” function and then invoke the “`loop()`” in an infinite loop.
- Suppose that we implement an application that have 2 jobs:
  - + job A to scan the input from keyboard.
  - + job B to send a message and wait for response.
    - If job B takes long time to finish, it will block job A.
    - So job A may miss some keyboard pressing event. It is not a good design

```
void loop(){
    jobA_scan_the_keyboard();
    jobB_send_message();
    while(1){
        if(jobB_get_response()){
            break;
        }
    }
}
```

<http://www.iotsharing.com/2017/05/how-to-apply-finite-state-machine-to-arduino-esp32-avoid-blocking.html>



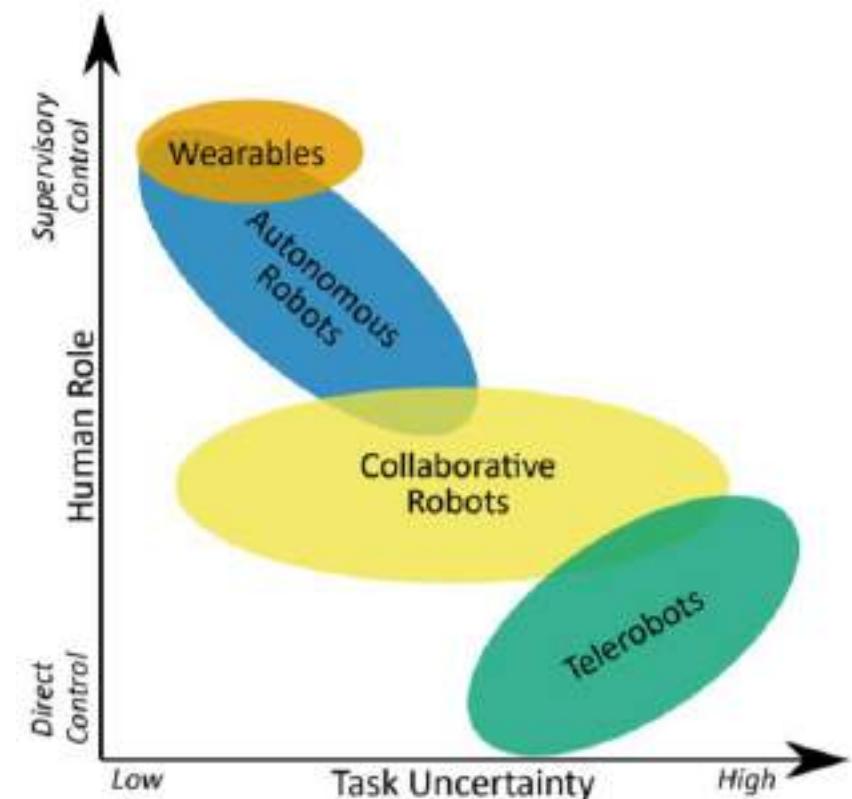
- [Mar06] Peter Marwedel, “**Embedded System Design**”, Springer, 2006.
- [VB17] Ovidiu Vermesan and Joel Bacquet, “**Cognitive Hyperconnected Digital Transformation: Internet of Things Intelligence Evolution**”, River Publishers, 2017.
- [BHC+18] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, “**The Industrial Internet of Things (IIoT): An Analysis Framework**”, Computers in Industry 101, October 2018.
- [BS19] Rajkumar Buyya and Satish Narayana Srirama, “**Fog and Edge Computing: Principles and Paradigms**”, Wiley Series on Parallel and Distributed Computing, 2019.
- [RS19] Ammar Rayes and Samer Salam, “**Internet of Things from Hype to Reality: The Road to Digitization**”, 2<sup>nd</sup> Edition, Springer, 2019.
- [BD16] R. Buyya and A. V. Dastjerdi, “**Internet of Things: Principles and Paradigms**”, 2016.
- [DF20] John Davies and Carolina Fortuna, “**Internet of Things: From Data to Insight**”, Wiley, 2020.
- [LEA20] Perry Lea, “**IoT and Edge Computing for Architects: Implementing edge and IoT systems from sensors to clouds with communication systems, analytics, and security**”, 2<sup>nd</sup> Edition, Packt Publishing, 2020.
- [GG20] Patrick Grossete and Stephen Goodman, “**Wireless Technologies and Use Cases in Industrial IoT**”, BKKIOT-1775, Cisco Live, January 2020.

# Automation vs Autonomy



## Human Factors Comparative Analysis

Characteristics of Human Factors	Non-AI-Based Automated Systems (with varied levels of automation)	AI-Based Autonomous Systems (with varied levels of autonomy)
	Examples: Office software, clothes washing machines, elevators, automated manufacturing lines	Examples: smart speakers, intelligent decision systems, autonomous vehicles, intelligent robots
Human-like sensing ability	Yes	Yes (advanced technologies)
Human-like cognitive abilities (perceptual integration, pattern recognition, learning, reasoning, etc.)	No	Yes (abilities vary across design scenarios; advanced systems may set self-defined goals, adjust strategy)
Human-like execution ability	No (require human manual activation and intervention according to predefined rules)	Yes (execute independently in specific situations, abilities vary across scenarios)
Human-like adaptive ability to unpredictable environments	No	Yes (abilities vary across scenarios)
Operation outcomes	Deterministic	Non-deterministic (machine behaviors may evolve in unexpected ways)
Need for human manual intervention		Needs vary across levels of automation/autonomy; humans must be the ultimate decision makers



Modalities of Physical Human-Robot Interaction



# **ITCS447**

## **Lecture 3.1**

### **Introduction to ESP32**

**Asst. Prof. Dr. Thitinan Tantidham**



มหาวิทยาลัยมหิดล  
Mahidol University

ห้ามมิให้นักศึกษาทำข้า ดัดแปลง หรือใช้ประโยชน์จากการอันมีลิขสิทธิ์ปราการอยู่ในระบบการศึกษาอิเล็กทรอนิกส์ (E-Learning) ของมหาวิทยาลัย ไม่ว่าจะทั้งหมดหรือบางส่วน โดยไม่ได้รับอนุญาตจากมหาวิทยาลัย นอกจากนี้จากการศึกษาส่วนบุคคล ทั้งนี้ การทำข้า ดัดแปลง หรือเผยแพร่ต่อสาธารณะนั้นงานอันมีลิขสิทธิ์ จะมีโทษปรับตั้งแต่ 20,000 บาท ถึง 200,000 บาท และหากเป็นการกระทำเพื่อการค้า จะมีโทษจำคุกตั้งแต่ 6 เดือน ถึง 10 ปี หรือปรับตั้งแต่ 100,000 บาท ถึง 800,000 บาท หรือทั้งจำทั้งปรับ

Unless allowed by Mahidol University, the User shall not copy, modify, or exploit in part or in whole of the copyrighted materials on the Platform, other than for your own individual study. The copy, modify or communication to public of the copyrighted materials shall be inflicted with a fine from 20,000 Baht up to 200,000 Baht. If the offence is committed with the commercial purpose, the offender shall be inflicted with imprisonment for a term from 6 months up to 10 years or a fine from 100,000 Baht up to 800,000 Baht or both imprisonment and fine.



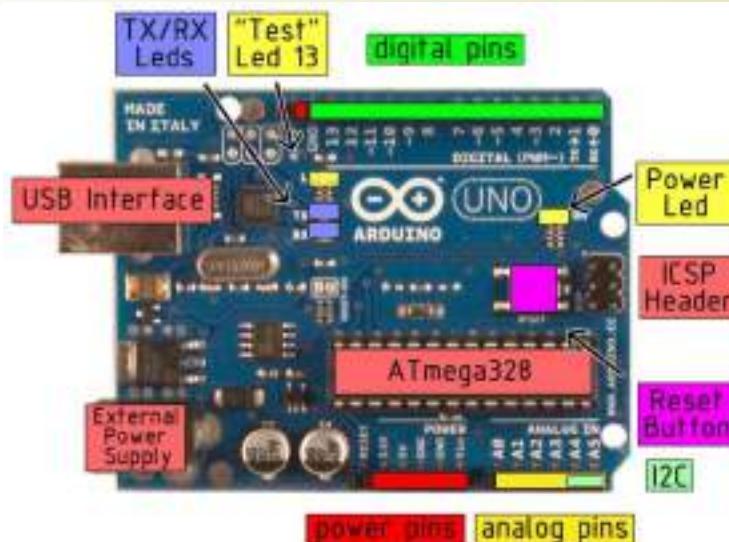
# Part I: Outline



- Arduino Uno R3/Due
- ESP32



## Atmega328P/168PA Pin Mapping



Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB of which 0.5 KB used by bootloader
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

### Digital pins:

- Pins 0 – 7: PORT D [0:7]
- Pins 8 – 13: PORT B [0:5]
- Pins 14 – 19: PORT C [0:5] (Arduino analog pins 0 – 5)
- digital pins 0 and 1 are RX and TX for serial communication
- digital pin 13 connected to the base board LED

### Arduino Evolution

<https://www.youtube.com/watch?v=ujiuXj8lrA>

### Atmega168 Pin Mapping

Arduino function		Arduino function
reset	(PCINT14/RESET) PC6	PC5 (ADC5/SCL/PCINT13)
digital pin 0 (RX)	(PCINT16/RXD) PD0	PC4 (ADC4/SDA/PCINT12)
digital pin 1 (TX)	(PCINT17/TXD) PD1	PC3 (ADC3/PCINT11)
digital pin 2	(PCINT18/INT0) PD2	PC2 (ADC2/PCINT10)
digital pin 3 (PWM)	(PCINT19/OC2B/INT1) PD3	PC1 (ADC1/PCINT9)
digital pin 4	(PCINT20/XCK/T0) PD4	PC0 (ADC0/PCINT8)
VCC	VCC	GND
GND	GND	GND
crystal	(PCINT6/XTAL1/TOSC1) PB6	AREF
crystal	(PCINT7/XTAL2/TOSC2) PB7	AVCC
digital pin 5 (PWM)	(PCINT21/OC0B/T1) PD5	PB5 (SCK/PCINT5)
digital pin 6 (PWM)	(PCINT22/OC0A/AIN0) PD6	PB4 (MISO/PCINT4)
digital pin 7	(PCINT23/AIN1) PD7	PB3 (MOSI/OC2A/PCINT3)
digital pin 8	(PCINT0/CLKO/ICP1) PB0	digital pin 11(PWM)
		digital pin 10 (PWM)
		digital pin 9 (PWM)

### Electronic stuff

Output pins can provide 40 mA of current

Writing HIGH to an input pin installs a 20KΩ pullup

Digital Pins 11,12 & 13 are used by the ICSP header for MOSI, MISO, SCK connections (Atmega168 pins 17,18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.



ATmega16u2 replaces FT232RL for USB-serial comms

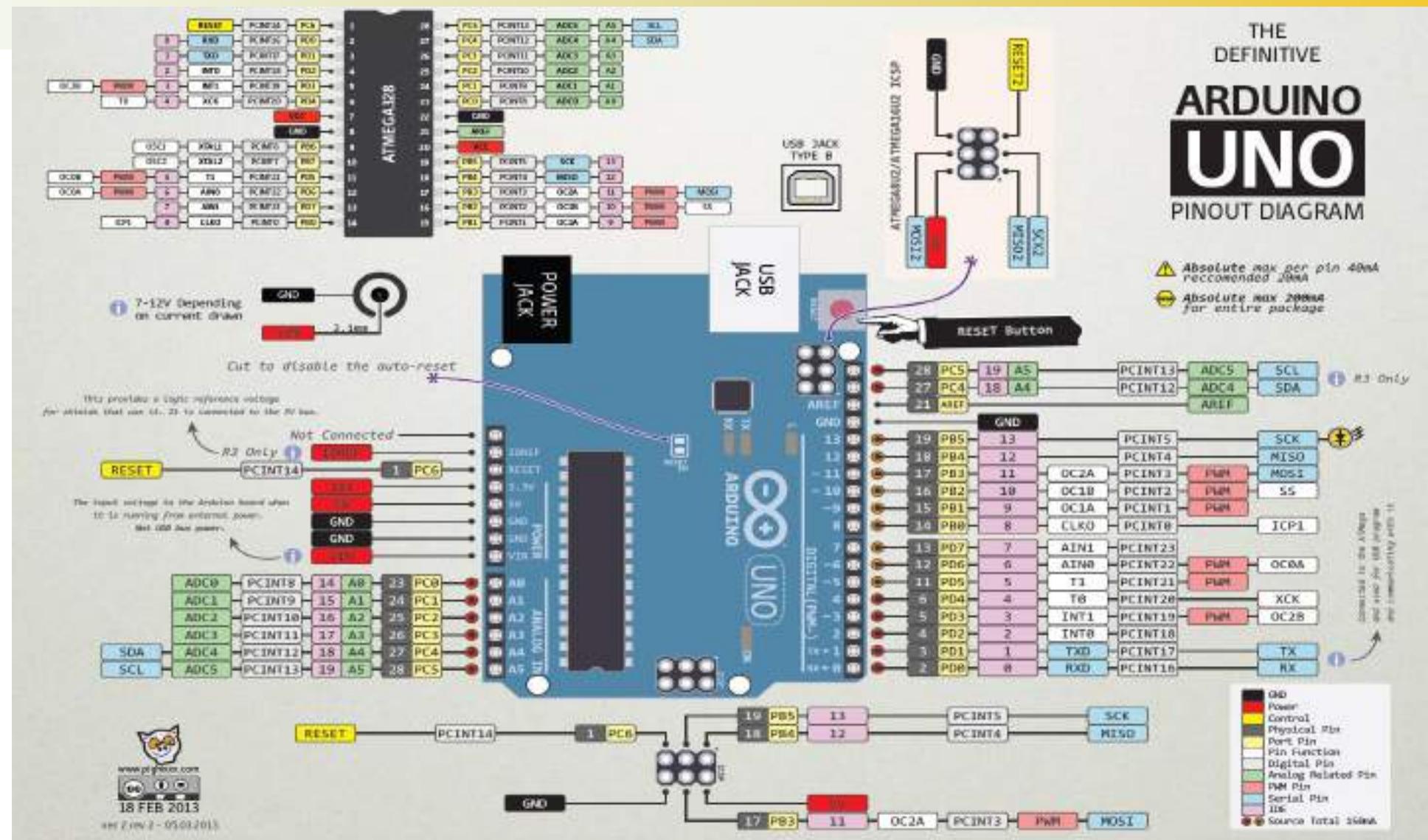


- Power
  - VIN, 5V, 3.3V, GND
- Control
  - RESET
- Analog In
  - A0, A1, A2, A3, A4, A5
- Digital I/O
  - 0,1,2,3,4,5,6,7,8,9,10,11,12,13
- Reference for Shields
  - AREF, IOREF
- Programmable LED
  - 13

[http://www.adafruit.com/index.php?main\\_page=popup\\_image&pID=50](http://www.adafruit.com/index.php?main_page=popup_image&pID=50)

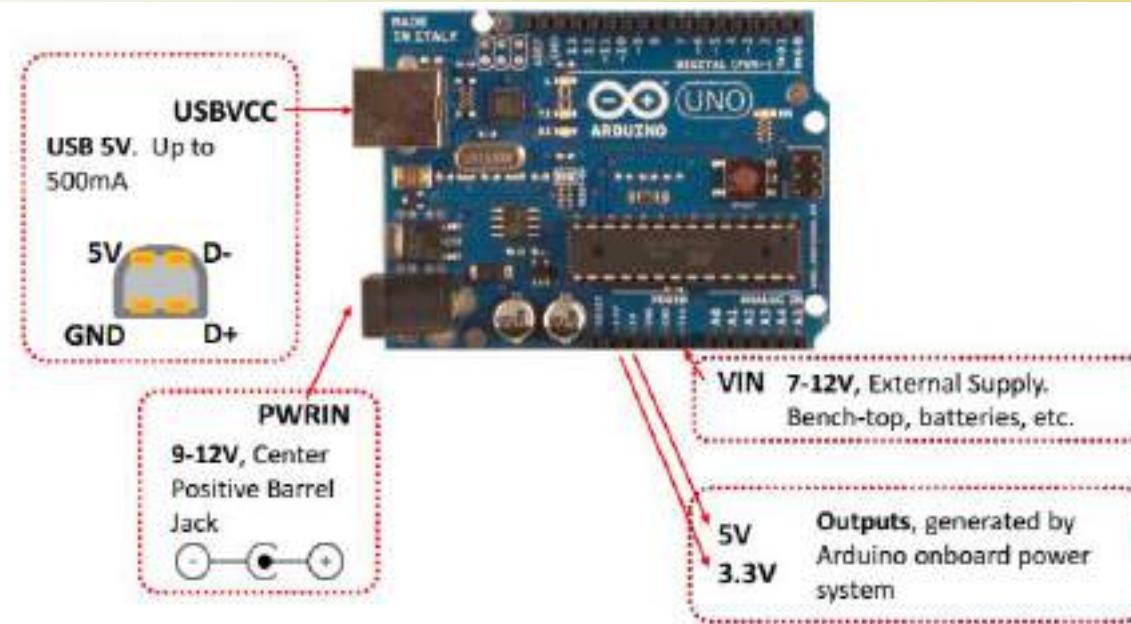
See: <http://learn.adafruit.com/arduino-tips-tricks-and-techniques/arduino-uno-faq>

# Arduino Uno R3





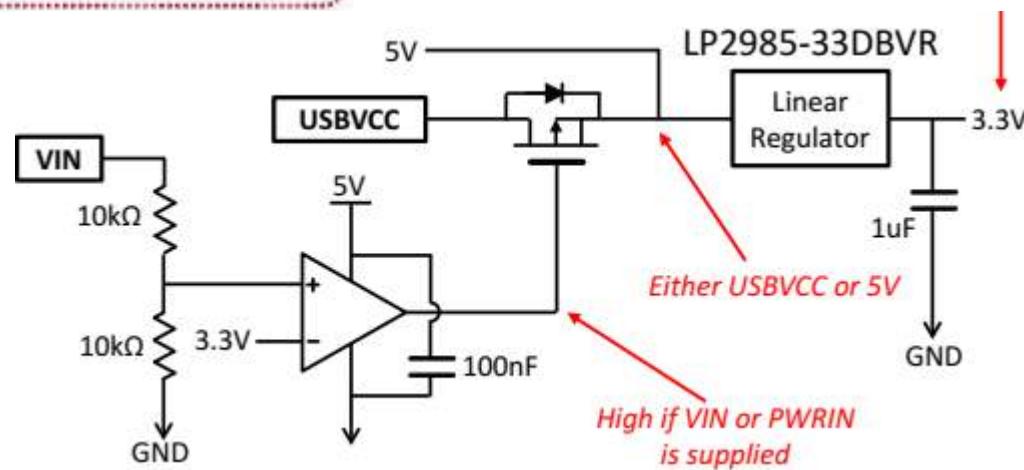
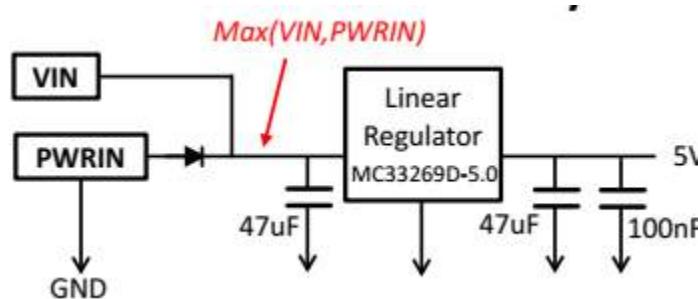
## Power System



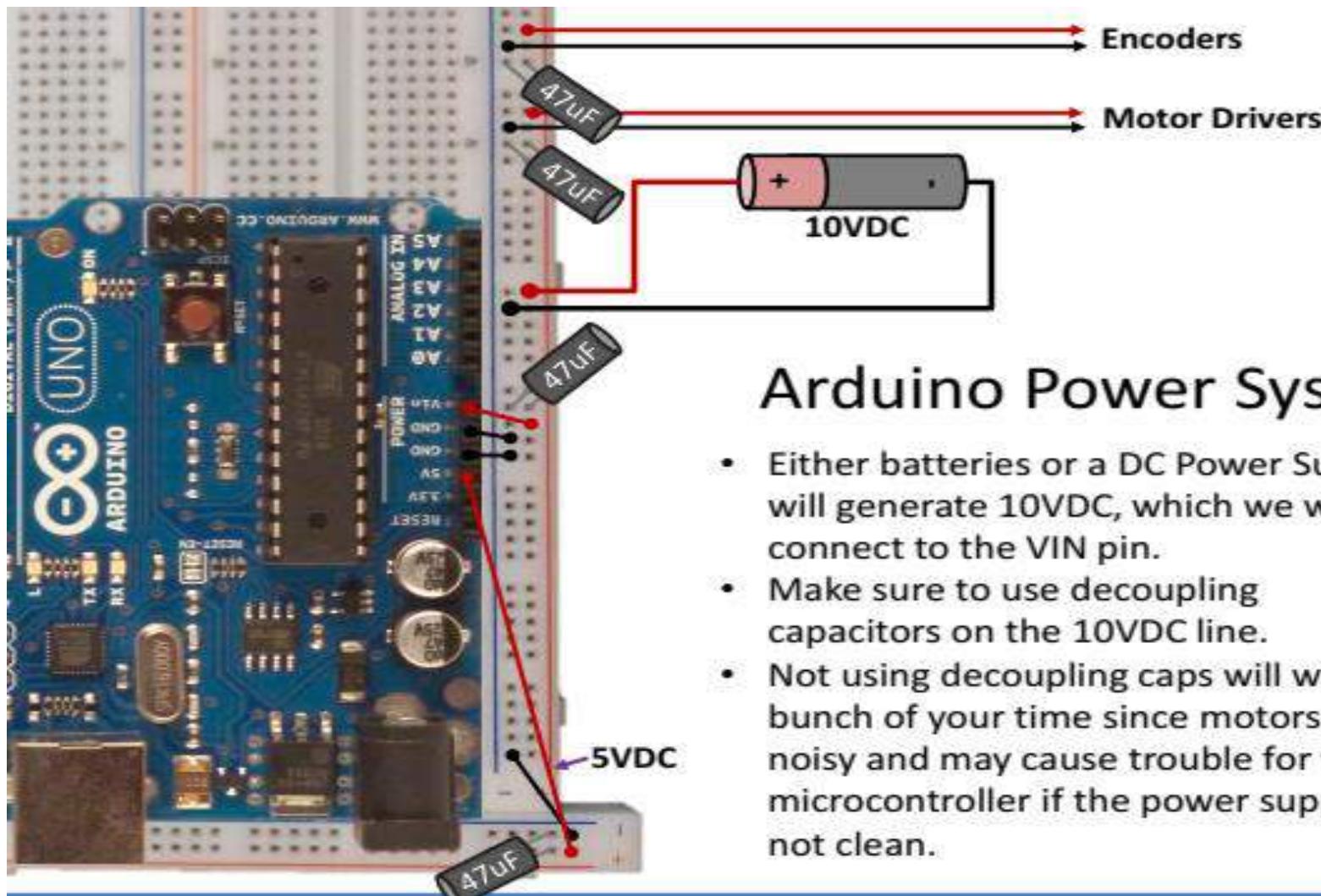
Three power inputs:

- USB (provides 5VDC),
- VIN pin (7-12VDC),
- A barrel jack (9-12VDC)

Two power outputs: 5V, 3.3V



## Power System

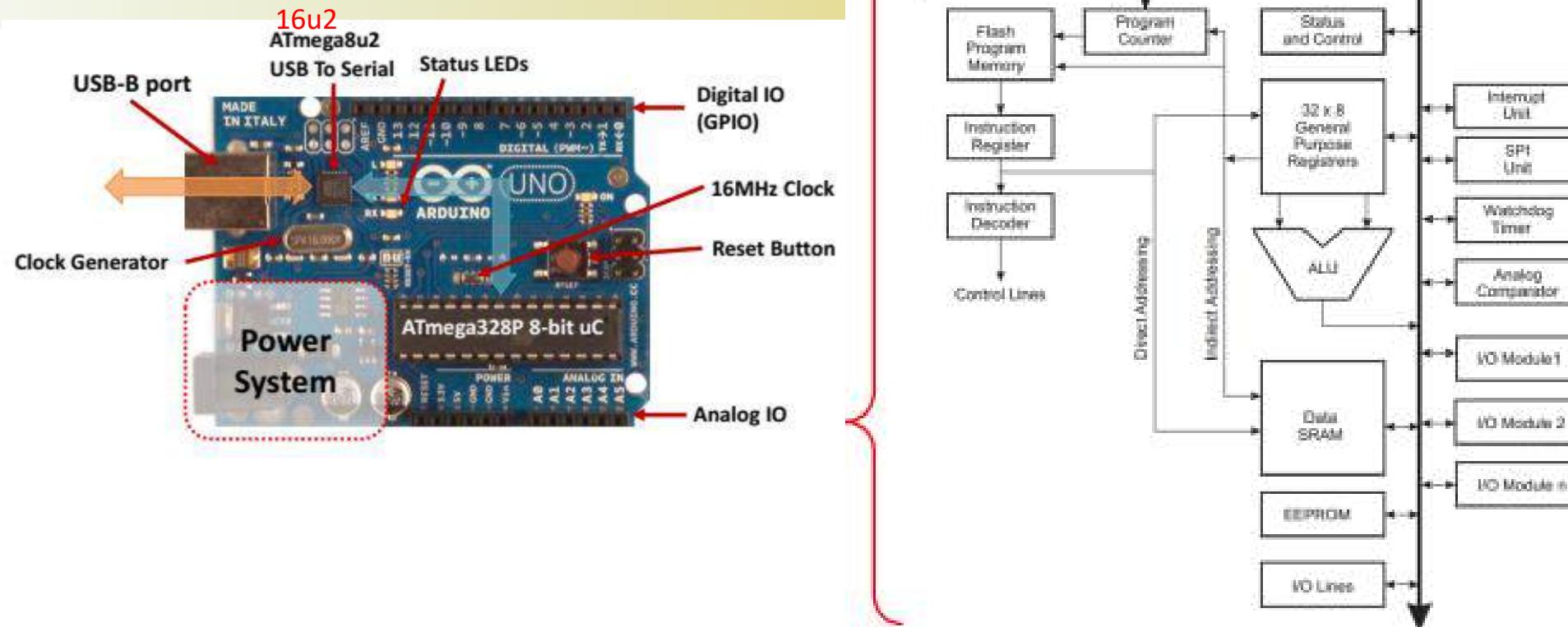


### Arduino Power System

- Either batteries or a DC Power Supply will generate 10VDC, which we will connect to the VIN pin.
- Make sure to use decoupling capacitors on the 10VDC line.
- Not using decoupling caps will waste a bunch of your time since motors are noisy and may cause trouble for your microcontroller if the power supply is not clean.



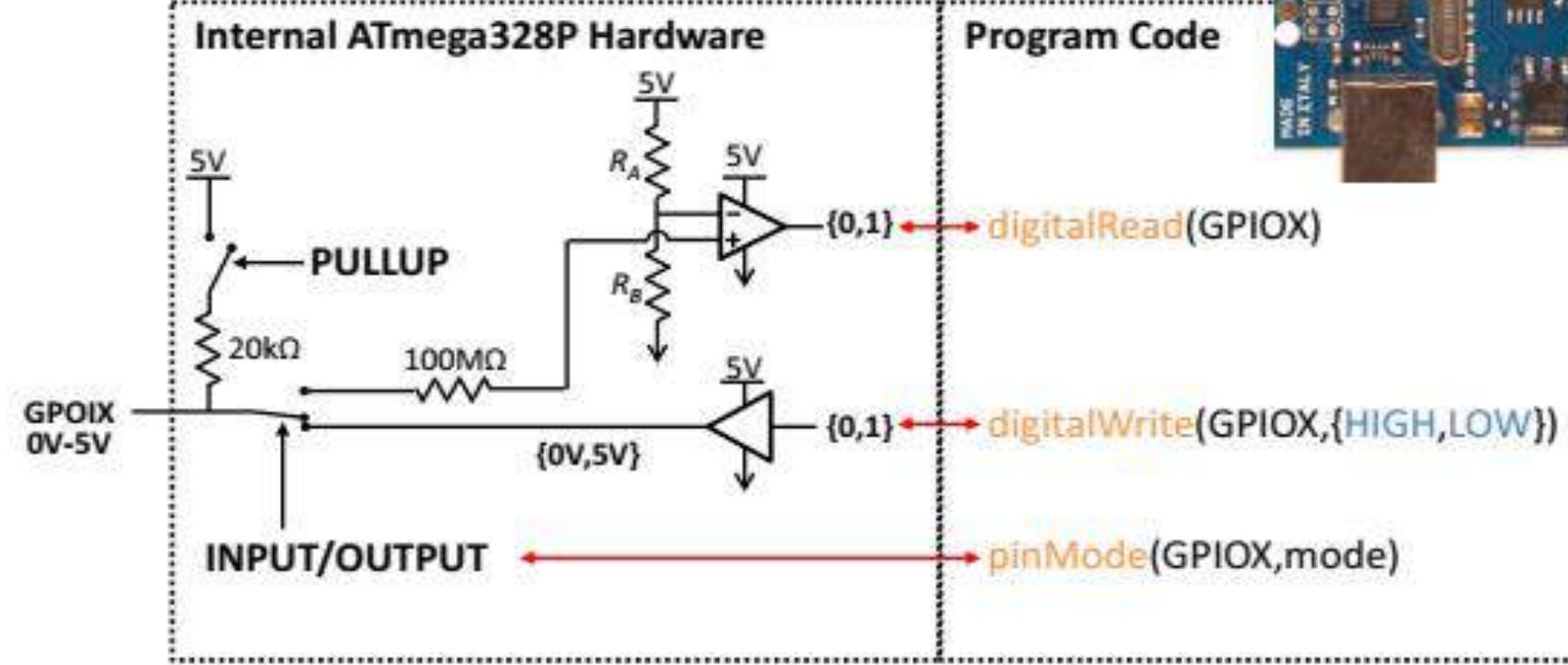
## 8-bit microcontroller



- An Arduino Uno/Mega has two microcontrollers: Atmega328 and 16u2.
- The 16u2 is normally used for USB to Serial communication.  
PC ←usb→ 16u2 ←hw serial→ 328
- Hoodloader2 gives you the option to reprogram the 16u2 of a normal Arduino Uno/Mega R3 with Custom sketches. We can use the 16u2 as a normal USB AVR like an Arduino Leonardo board

## Digital Inputs/Outputs (GPIO)

GPIO: General Purpose Inputs/Outputs. These are digital outputs, so only voltages of 0V or 5V may be generated. Pins marked with (~) may be used to output PWM signals.  
Maximum current of 40mA per output. Pins default to Inputs



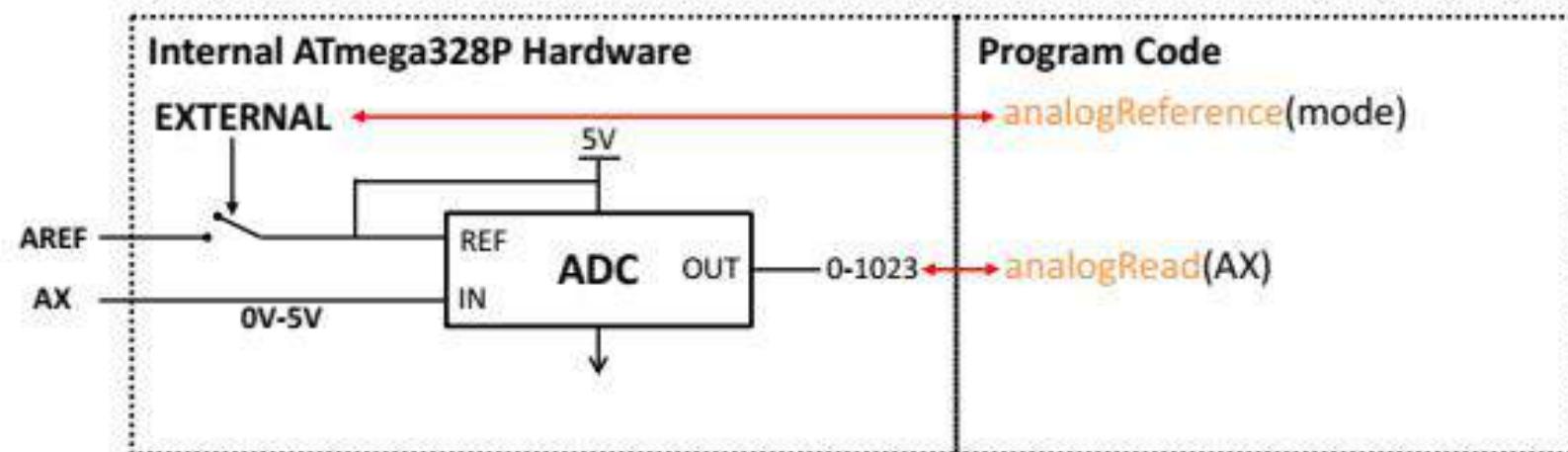
## Analog Inputs



Analog Inputs: 6 Analog inputs are available. These pins convert an analog voltage in the range 0V-AREF to an integer in the range 0-1023. The value read by the ATmega328 can be calculated as:

$$VAL_X = \text{Integer} \left[ \frac{V_A}{AREF} (1023) \right]$$

Analog reference AREF defaults to 5V. It takes approximately 100  $\mu$ s to perform an analog read. Analog inputs A0-A5 can be configured to act exactly like the GPIO pins if needed.

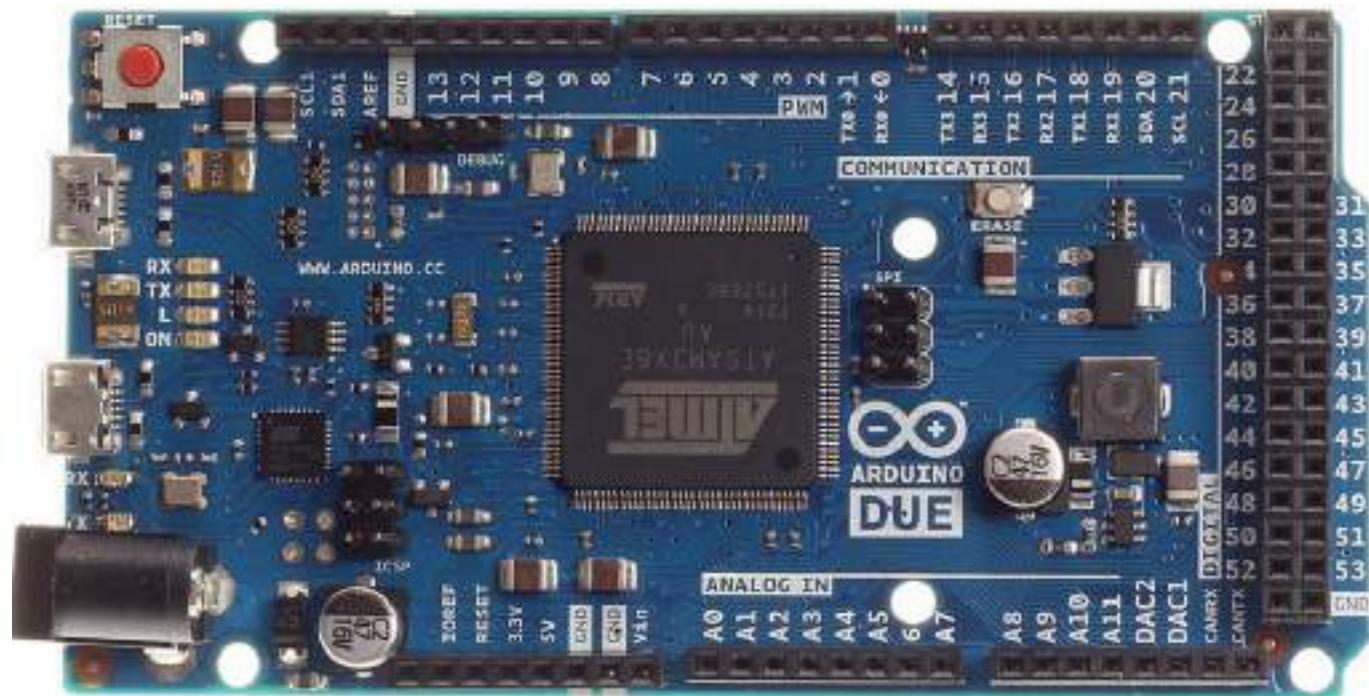


# Arduino Due



Note: **3.3 V !!**

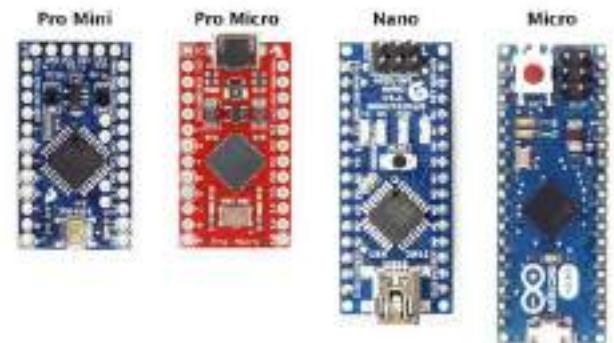
Atmel SAM3X8E processor (32 bit ARM Cortex M3 architecture, 84MHz)



[http://www.adafruit.com/index.php?main\\_page=popup\\_image&pID=1076](http://www.adafruit.com/index.php?main_page=popup_image&pID=1076)

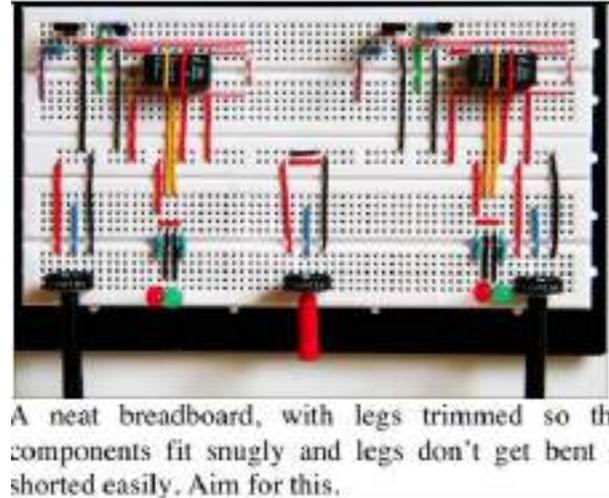
See: <https://www.arduino.cc/en/pmwiki.php?n>Main/arduinoBoardDue>

# Arduino Products



<https://www.arduino.cc/en/Main/Products>

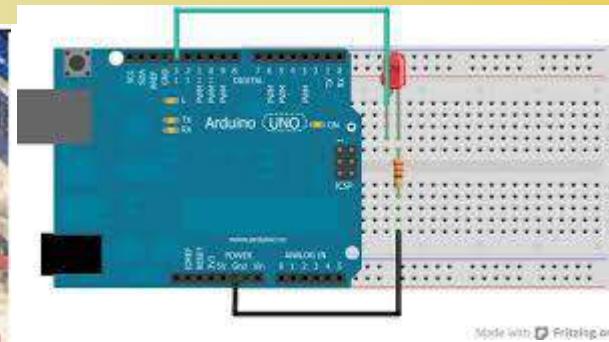
# The Neat Breadboard



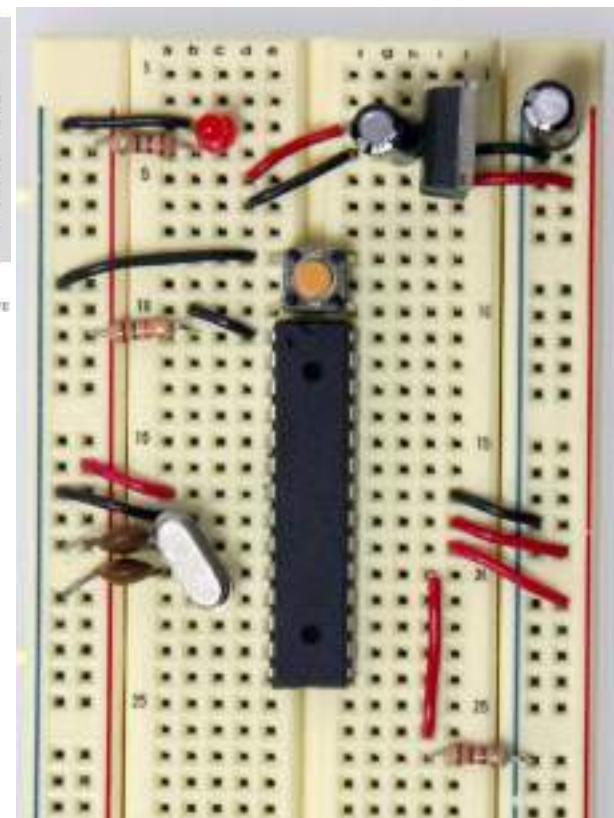
A neat breadboard, with legs trimmed so that components fit snugly and legs don't get bent or shorted easily. Aim for this.



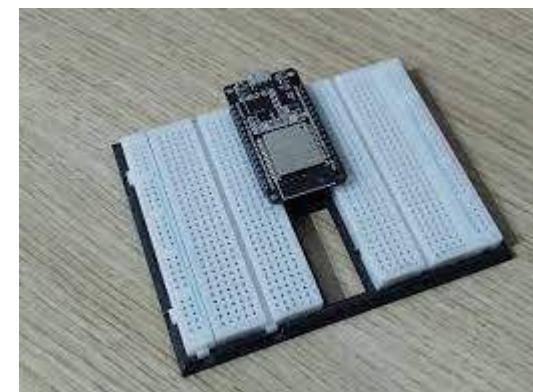
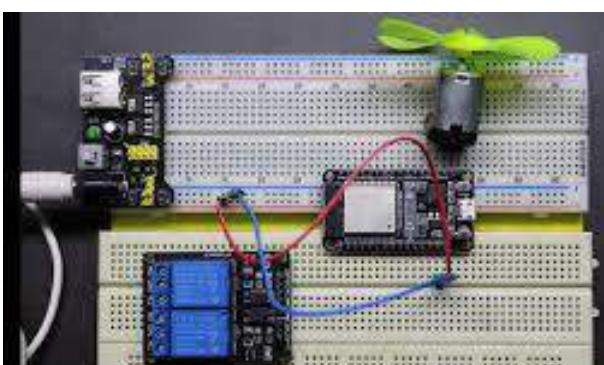
The alternative. Don't do this.



Made with Fritzing.org



Arduino on board

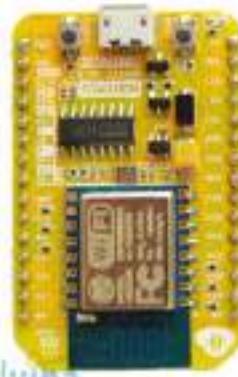


<https://www.arduino.cc/en/main/standalone>



<http://esp32.net/usb-uart/>

V1 or V0.9 (official)  
ESP-12,  
USB to serial CH340



V2 or V1.0 (official)  
ESP-12E,  
USB to serial CP2102



V3 produced by Lolin  
ESP-12E,  
USB to serial CH340



<http://espressif.com/>

[https://www.eurotronix.com/files/shares/ESSPRESIF%20Company%20Brochure\\_EN.pdf](https://www.eurotronix.com/files/shares/ESSPRESIF%20Company%20Brochure_EN.pdf)



ESP32-C3

ESP32-S2

ESP32

ESP8285

ESP8266

## Product details

CPU	RISC-V 32-bit Single-Core @ 160 MHz	Tensilica Xtensa LX7 32 bit Single-Core @ 240Mhz	Tensilica Xtensa LX6 32 bit Dual-Core @ 160 / 240Mhz	Tensilica LX106 32 bit @ 80 MHz (up to 160 MHz)	Tensilica LX106 32 bit @ 80 MHz (up to 160 MHz)
Coprocessor		ULP (RISC-V)	ULP		
RAM	400 KB	320 KB	520 KB	160 KB	160 KB (36 KB available to user)
ROM (Flash)	384 KB	128 KB	448 KB	1,024 KB	
UART	2	2	3	2	2
SPI	3	4 (QSPI)	4	2	2
Wi-Fi	802.11b/g/n	802.11b/g/n	802.11b/g/n	802.11b/g/n (max 65Mbps)	802.11b/g/n (max 65Mbps)
Bluetooth®	5.0 + BLE	—	4.2 BR/EDR + BLE	—	—
USB Host	—	—	—	—	—
Ethernet (LAN, RJ45)	—	—	10/100Mbps	—	—
RTC Memory	8 KB	16 KB	16 KB		
External SPIRAM		128 MB up to	16 MB up to	16 MB up to	16 MB up to
External Flash		1 GB up to			
ESP-MESH	—	—	—	—	—
CAN			v2.0		
GPIO	22 (up to)	43	32 (up to)	17 (up to)	17 (up to)
Hall effect sensor	—	—	—	—	—
Temperature sensor	—	—	—	—	—
Touch sensors		14	10	No	No
I²C	up to 1	up to 2	up to 2	up to 1	up to 1
I²S	1	1	2	2	2
ADC	2 × 12-bit SAR ADCs, up to 6 channels	12 bit SAR ADC up to 20 channels	12 bit SAR ADC up to 18 channels	10 bit	10 bit
DAC		8 bit up to 2 channels	8 bit up to 2 channels	—	—
PWM	up to 6	8	up to 8	—	—
SDMMC		—	—	—	—
RMT (remote control)		—	—	—	—
LCD Interface	—	—	—	—	—
Camera Interface	—	—	—	—	—
Secure flash	4096-bit OTP	4096-bit OTP secure boot	1024-bit OTP secure boot		
Cryptography support	AES-128/256, RSA Accelerator, SHA Accelerator, Random Number Generator (RNG), HMAC	AES-128/192/256, SHA-2, RSA, RNG, HMAC, Digital Signature	AES, SHA-2, RSA, ECC, RNG		
Deep sleep consumption		5 µA	10 µA	20 µA	20 µA

## Dimensions

Height	0.19685 in (5mm)	0.27559 in (7 mm)	0.23622 in (6mm)	0.19685 in (5mm)	0.19685 in (5mm)
Width	0.19685 in (5mm)	0.27559 in (7 mm)	0.23622 in (6mm)	0.19685 in (5mm)	0.19685 in (5mm)

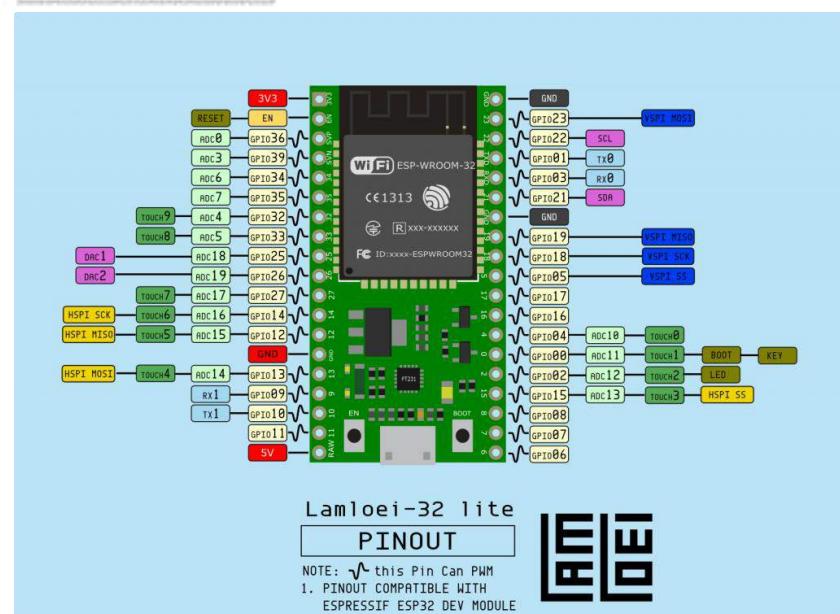
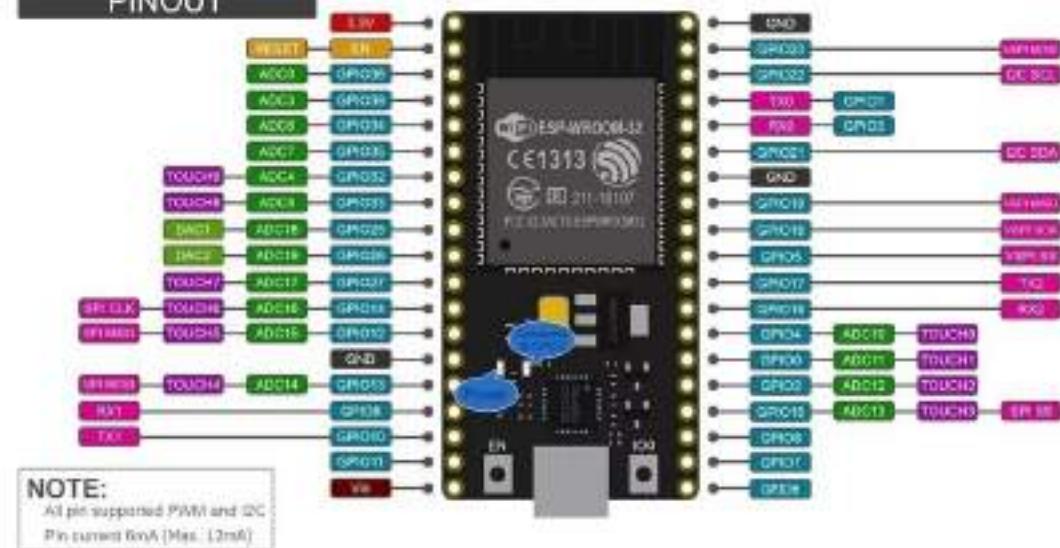
<https://socialcompare.com/en/comparison/esp8266-vs-esp32-vs-esp32-s2><https://www.espressif.com/en/products/modules>

# ESP32 Series



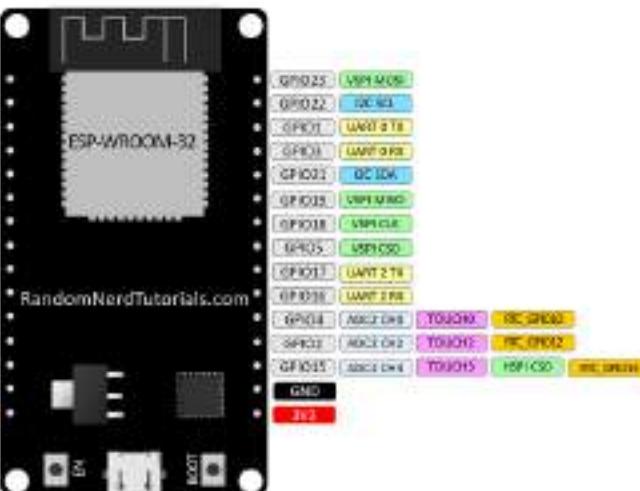
NodeMCU-32S

## PINOUT

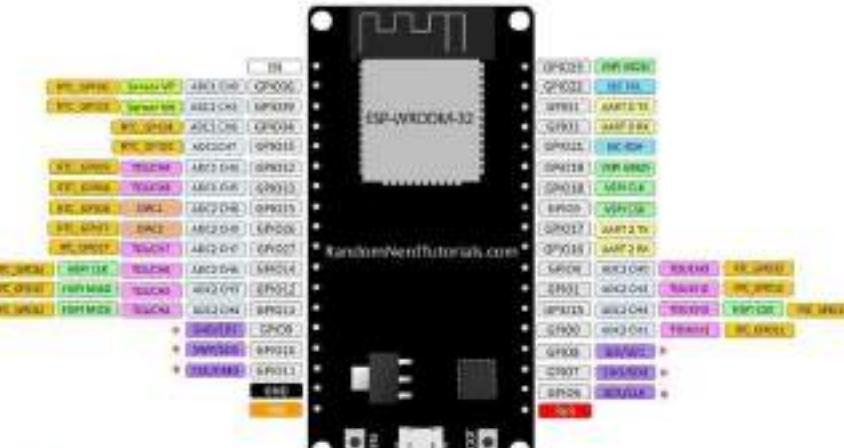


ESP32 DEVKIT V1 - DOIT

version with 30 GPIOs



ESP32 DEVKIT V1 - DOIT



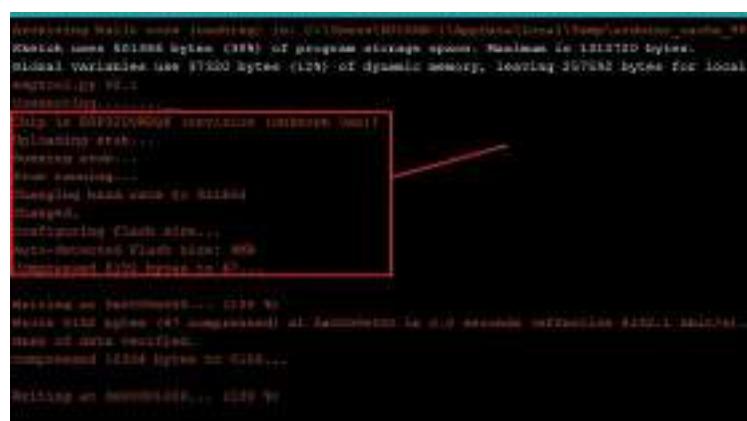
PN1304/13K, 3003/13R, 304/13L, 3H01/13R, 3W01/13R and 3C5/13R, namely, RP106-10, RP1011, are connected to the integrated SPI Bus, integrated on ESP\_WIRDOM\_32 and are not recommended for other uses.

## Troubleshooting (1)



If you try to upload a new sketch to your ESP32 and you get this error message “*A fatal error occurred: Failed to connect to ESP32: Timed out... Connecting...*”. It means that your ESP32 is not in flashing/uploading mode.

- To ensure that you have the right board name and COM port selected, follow these steps:
    - ✓ Hold-down the “**BOOT**” button in your ESP32 board
    - ✓ Press the “**Upload**” button in the Arduino IDE to upload your sketch
    - ✓ After you see the “Connecting ...” message in your Arduino IDE, release the finger from the “**BOOT**” button.
    - ✓ After that, you should see the “Done uploading” message.





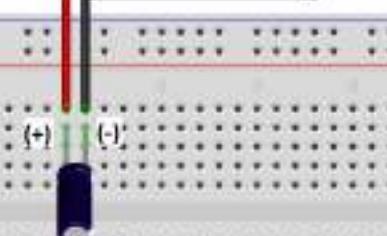
# Troubleshooting (2)

```
All error occurred while uploading the sketch
C:\Users\Sara\AppData\Local\Arduino15\packages\esp32\hardware\esp32\1.0.1/tools/gen_esp32part.exe" -q "C:\Users\Sara
"C:\Users\Sara\AppData\Local\Arduino15\packages\esp32\tools\espota_py\2.6.0/espota.exe" --chip esp32 elf2Image --flash
espota.py v2.6-beta1

"C:\Users\Sara\AppData\Local\Arduino15\packages\esp32\tools\xtensa-esp32-elf-gcc\1.22.0-80-g6c4433a-5.2.0/bin/xtensa-esp
Sketch uses 190944 bytes (14%) of program storage space. Maximum is 1310720 bytes.
Global variables use 12632 bytes (3%) of dynamic memory, leaving 315048 bytes for local variables. Maximum is 327680 bytes.
C:\Users\Sara\AppData\Local\Arduino15\packages\esp32\tools\espota_py\2.6.0/espota.exe --chip esp32 --port COM7 --baud 921600 --
espota.py:2:4: error: syntax error
syntax error: CNT
Connecting..... An error occurred while uploading the sketch

A fatal error occurred: Failed to connect to ESP32: Timed out waiting for packet header

```



## Problem:

Always hold-down the “**BOOT/FLASH**” button in your ESP32 board while uploading a new sketch at the same time. But having to worry about this every time you want to upload new code can be tedious, specially when you’re testing and debugging your code.

## Solution:

To make your ESP32 board go into flashing/uploading mode automatically, you can **connect a 10 uF electrolytic capacitor between the EN pin and GND**.

<https://randomnerdtutorials.com/solved-failed-to-connect-to-esp32-timed-out-waiting-for-packet-header/>

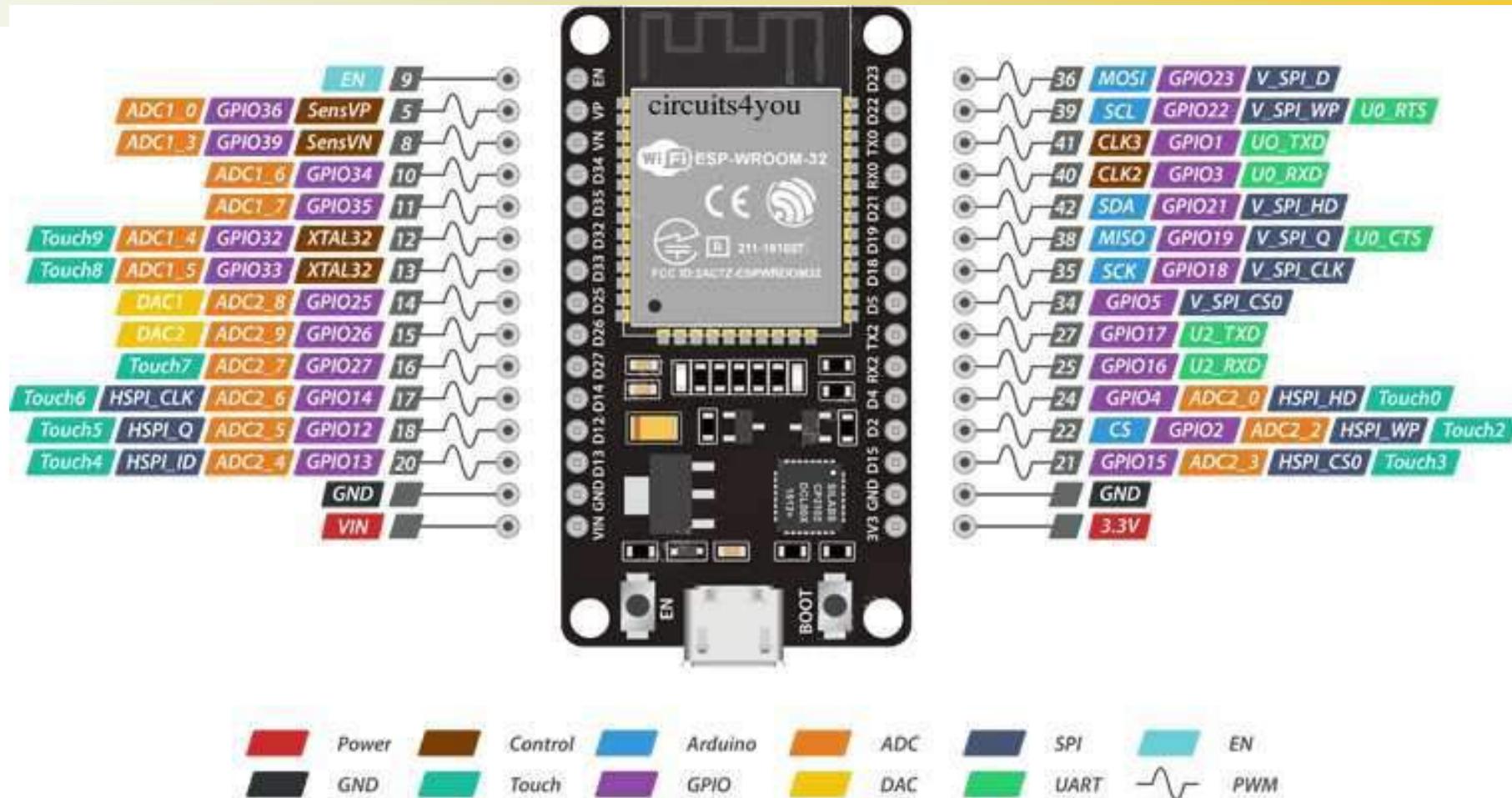
## Since 2016

- ESP32-D0WDQ6 Chip  
[https://www.espressif.com/sites/default/files/documentation/esp32\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf)
- two CPU cores that can be individually controlled
- CPU clock frequency is adjustable from 80 MHz to 240 MHz.
- The user may also power off the CPU and make use of the low-power co-processor to constantly monitor the peripherals for changes or crossing of thresholds.
- ESP32 integrates a rich set of peripherals, ranging from capacitive touch sensors, Hall sensors, SD card interface, Ethernet, high-speed SPI, UART, I2S and I2C.
- Support Bluetooth, Bluetooth LE, Wi-Fi
- The sleep current is less than 5  $\mu$ A
- Support a data rate of up to 150 Mbps and 20.5 dBm output power at the antenna
- Support freeRTOS with LWIP
- Secure (encrypted) over the air (OTA) upgrade so that developers can continually upgrade their products even after their release





## DevKit Pinout (Pin Layout)



<http://esp32.net/#Hardware>

ESP32 Dev. Board / Pinout

## ESP Peripherals Feature

- 18 x Analog to-Digital Converter (ADC) channels
- 10 x Capacitive sensing GPIOs
- 3 x UART interfaces
- 3 x SPI interfaces
- 2 x I2C interfaces
- 16 x PWM output channels
- 2 x Digital-to-Analog Converters (DAC)
- 2 x I2S interfaces



## 4 x Input Pins + 25 x GPIO Pins

- **Input Only Pins:**
  - GPIOs 34,35, 36, 39
- **Other GPIOs can be used for input or output**
  - Pins with internal Pull Up “INPUT\_PULLUP”
    - GPIOs 14, 16, 17, 18, 19, 21, 22, 23
  - **Pins without internal Pull Up**
    - GPIOs 13, 25, 26, 27, 32, 33

For example: To make GPIO22 as input and GPIO23 as output

```
pinMode(22, INPUT_PULLUP);  
pinMode(23, OUTPUT);  
digitalWrite(23, HIGH);
```

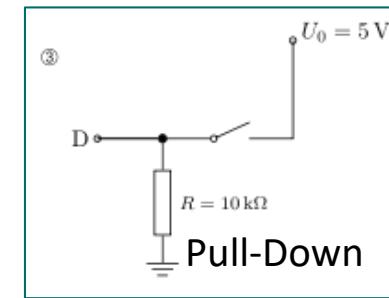
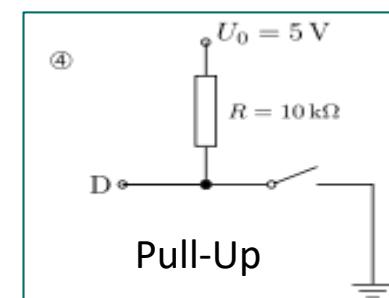
### Note:

Pull-Up (Active Low)

- It is to force GPIO to be HIGH state until press button it will change to LOW

Pull-Down (Active High)

- It is to force GPIO to be LOW state until press button it will change to HIGH



- **Analog Input Pins**
  - Resolution of up to 12 bits – 4096 distinct values (0 – 4095) and voltage ranges between these will produce a correspondingly scaled digital value.
  - The available scales beyond the 0-1V, 0-1.34V, 0-2V and 0-3.6V

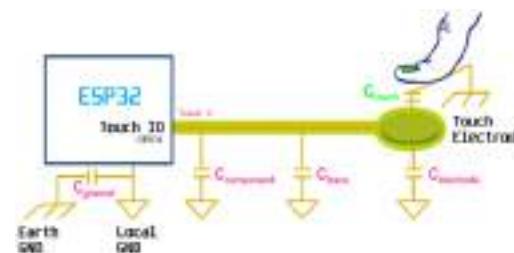
[https://circuits4you.com/2018/12/31/esp32-devkit-esp32-wroom-gpio-pinout/?fbclid=IwAR3Drl\\_Ls-rLJLCydPWNxMlkslbEUxiLOP3i5gqSBYYHEVnkRtQ0Swfdc88](https://circuits4you.com/2018/12/31/esp32-devkit-esp32-wroom-gpio-pinout/?fbclid=IwAR3Drl_Ls-rLJLCydPWNxMlkslbEUxiLOP3i5gqSBYYHEVnkRtQ0Swfdc88)

- Digital to Analog Converter (DAC)
  - GPIOs 25, 26 for DAC1, DAC2
- RTC GPIOs can be used to wake up the ESP32 from deep sleep
  - GPIOs 36, 39, 34, 35, 25, 26, 33, 32, 04, 0, 2, 15, 13, 12, 14, 27 for RTC\_GPIOs 0, 3-17 respectively.
- All pins that can act as outputs can be used as PWM pins with 8, 10, 12, 15-bit resolution for the parameters:
  - Signal's frequency, Duty cycle, PWM channel, GPIO
  - e.g. ledcWrite(pinChannel, dutyCycle); [1]
- 1 + 2 serial ports for programming
  - GPIOs 3, 1 (U0Rx, U0Tx) and GPIOs 16, 17 (U2Rx, U2Tx)
- I2C (supported by the Wire library)
  - GPIOs 21, 22 (SDA, SCL)
- SPI

SPI	MOSI	MISO	CLK	CS
VSPI	GPIO 23	GPIO 19	GPIO 18	GPIO 5
HSPI	GPIO 13	GPIO 12	GPIO 14	GPIO 15

[1] <https://techexplorations.com/guides/esp32/begin/pwm/>

- Capacitive Touch GPIOs
  - GPIOs 4, 0, 2, 15, 13, 12, 14, 27, 33, 32 for T0 – T9 respectively
- Interrupts
  - All GPIOs can be configured as interrupts
- Enable (EN)
  - Enable (EN) is the 3.3V regulator's enable pin. It's pulled up, so connect to ground to disable the 3.3V regulator. This means that you can use this pin connected to a pushbutton to restart your ESP32.
- GPIO Current drawn
  - The absolute maximum current drawn per GPIO is source 40mA and sink 28mA



- [Mar11] Michael Margolis, **Arduino Cookbook**, O'Reilly, 2011
- [Mon18] Simon Monk, **Programming Arduino Next Steps: Going Further with Sketches**, 2<sup>nd</sup> Edition, 2018.
- [OK13] Robert Oshana, Mark Kraeling, **Software Engineering for Embedded Systems**, e-ISBN: 978-0-12-415941-9, Newnes, 2013.
- [SS\*] Rui Santos and Sara Santos, **Learn ESP32 with Arduino IDE**, v1.4
- [Whi11] Elecia White, **Making Embedded Systems**, O'Reilly, 2011
- <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/>
- <https://randomnerdtutorials.com/getting-started-with-esp32/>
- <https://mjrobot.org/iot-made-simple-playing-with-the-esp32-on-arduino-ide/>
- <https://www.instructables.com/IOT-Made-Simple-Playing-With-the-ESP32-on-Arduino-/>
- <http://www.energiazero.org/meccatronica/IOT-Made-Simple-Playing-With-the-ESP32-on-Arduino-.pdf>



# ITCS447

## Lecture 3.2

### Arduino IDE Programming for ESP32 in Practice

Asst. Prof. Dr. Thitinan Tantidham



มหาวิทยาลัยมหิดล  
Mahidol University

ห้ามมิให้นักศึกษาทำข้า ดัดแปลง หรือใช้ประโยชน์จากการอันมีลิขสิทธิ์ปราการอยู่ในระบบการศึกษาอิเล็กทรอนิกส์ (E-Learning) ของมหาวิทยาลัย ไม่ว่าจะทั้งหมดหรือบางส่วน โดยไม่ได้รับอนุญาตจากมหาวิทยาลัย นอกจากนี้จากการศึกษาส่วนบุคคล ทั้งนี้ การทำข้า ดัดแปลง หรือเผยแพร่ต่อสาธารณะนั้นงานอันมีลิขสิทธิ์ จะมีโทษปรับตั้งแต่ 20,000 บาท ถึง 200,000 บาท และหากเป็นการกระทำเพื่อการค้า จะมีโทษจำคุกตั้งแต่ 6 เดือน ถึง 10 ปี หรือปรับตั้งแต่ 100,000 บาท ถึง 800,000 บาท หรือทั้งจำทั้งปรับ

Unless allowed by Mahidol University, the User shall not copy, modify, or exploit in part or in whole of the copyrighted materials on the Platform, other than for your own individual study. The copy, modify or communication to public of the copyrighted materials shall be inflicted with a fine from 20,000 Baht up to 200,000 Baht. If the offence is committed with the commercial purpose, the offender shall be inflicted with imprisonment for a term from 6 months up to 10 years or a fine from 100,000 Baht up to 800,000 Baht or both imprisonment and fine.





- Programming for ESP32
- Arduino IDE
- ESP32 Installation on Arduino IDE
- Digital Input/Output
- Analog Input/Output
- Basic Arduino Programming
- Pull-up/Pull-down Resistor Concept
- GPIO: Active Low/Active High
- `pinMode(GPIO,INPUT_PULLUP)`
- Pushbutton Switch or Momentary Button





- Arduino IDE
- Espressif IoT Development Framework (IDF) – official Espressif development framework for ESP32
- MicroPython A lean implementation of Python 3 for microcontrollers
- Espressif Mesh Development Framework
- Espruino – JavaScript SDK and firmware closely emulating Node.js
- Lua RTOS for ESP32
- Moddable SDK — includes JavaScript language and library support for the ESP32
- Mongoose OS – an operating system for connected products on microcontrollers; programmable with JavaScript or C. A recommended platform by Espressif Systems,<sup>[39]</sup> AWS IoT,<sup>[40]</sup> and Google Cloud IoT.<sup>[41]</sup>
- mruby for the ESP32
- .NET nano Framework - Coding in .NET C#, deploy and debug from Visual Studio<sup>[42]</sup>
- NodeMCU – Lua-based firmware
- PlatformIO Ecosystem and IDE:
- Pymakr IDE – IDE designed for use with Pycom devices; handles firmware upgrades and includes MicroPython REPL console
- Simba Embedded Programming Platform
- Whitecat Ecosystem Blockly Based Web IDE
- Zerynth – Python for IoT and microcontrollers, including the ESP32
- AtomVM -- Erlang/Elixir Abstract machine (BEAM) for ESP32

<http://esp32.net/>

<https://docs.espressif.com/projects/arduino-esp32/en/latest/>

[https://docs.espressif.com/projects/arduino-esp32/en/latest/getting\\_started.html](https://docs.espressif.com/projects/arduino-esp32/en/latest/getting_started.html)

<https://docs.platformio.org/en/latest/platforms/espressif32.html>

[https://docs.platformio.org/en/latest/tutorials/espressif32/arduino\\_debugging\\_unit\\_testing.html#tutorial-espressif32-arduino-debugging-unit-testing](https://docs.platformio.org/en/latest/tutorials/espressif32/arduino_debugging_unit_testing.html#tutorial-espressif32-arduino-debugging-unit-testing)

Arduino web sites:

<http://arduino.cc/en/Guide/Environment>  
<http://arduino.cc/en/Tutorial/HomePage>

Arduino Programming tutorials

- [Arduino Tutorial \(http://www.ladyada.net/learn/arduino/lesson2.html\)](http://www.ladyada.net/learn/arduino/lesson2.html)
- [Beginning Arduino Programming](#)
- [Arduino Projects for Dummies](#)
- [Arduino Cookbook](#)
- [Arduino Project Handbook](#)

# Getting Started and Testing a New Board



Check out: <https://www.arduino.cc/en/Guide>

1. Download & install the Arduino environment (IDE)
2. Connect the board to your computer via the USB cable
3. If needed, install the drivers
4. Launch the Arduino IDE
5. Select your board
6. Select your serial port
7. **Open the blink example for many boards**
8. Upload the program

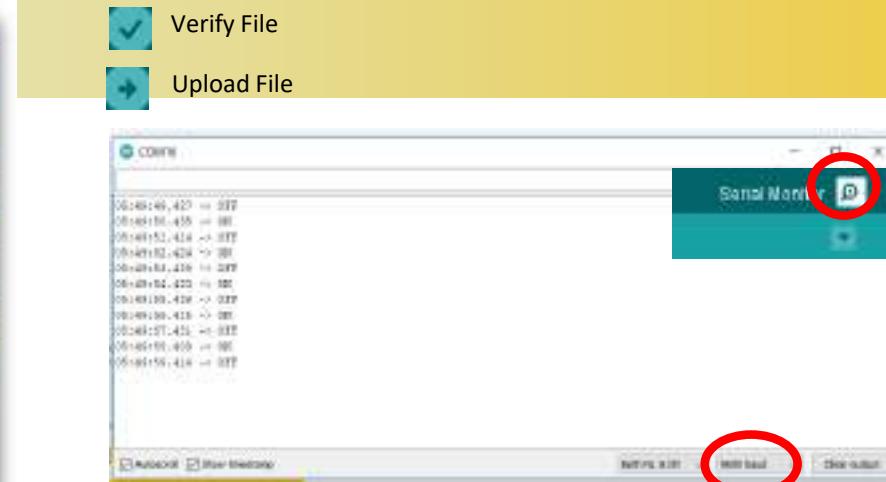
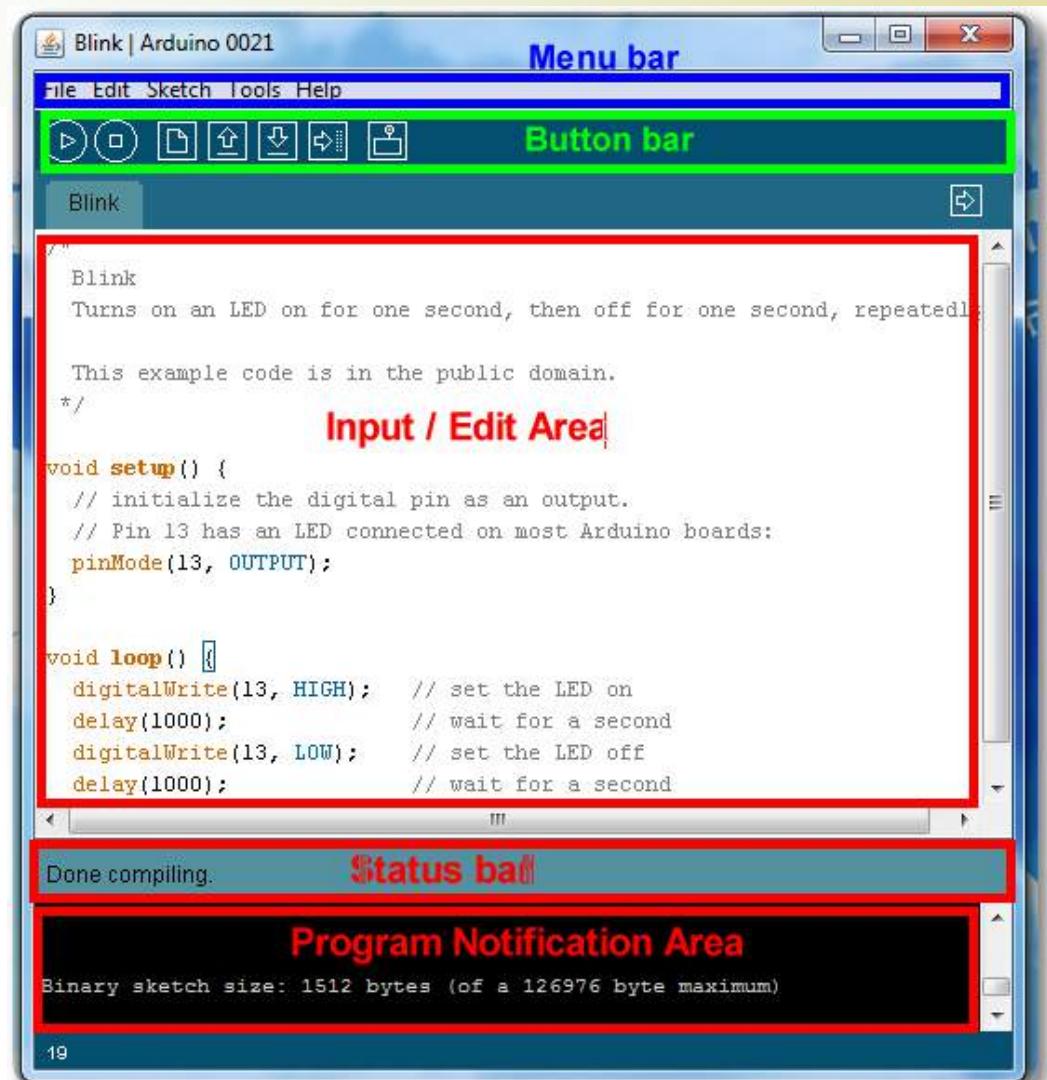
Note:

ESP32: Additionally test WiFi and BLE

Open WiFiScan for ESP32 or ESP8266

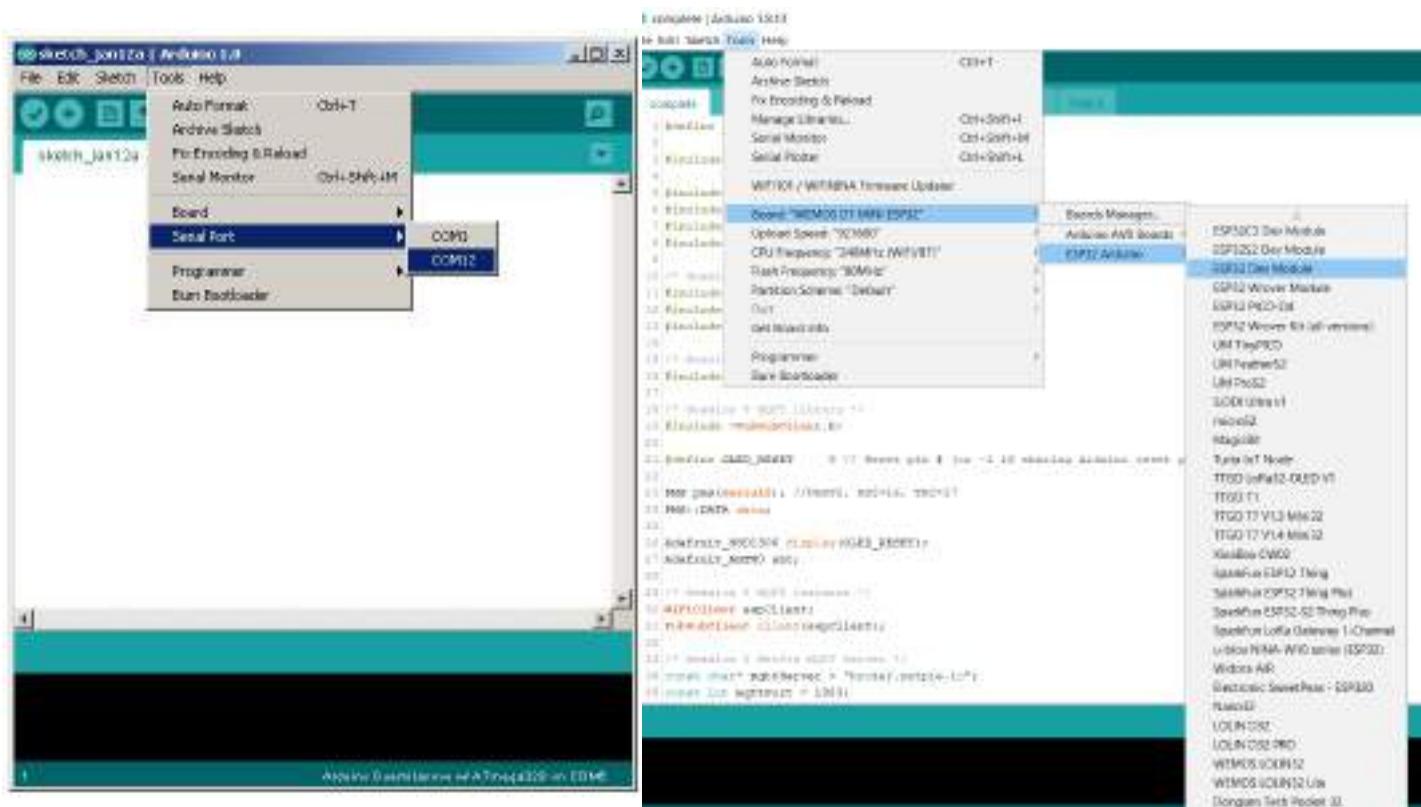
**File > Examples > WiFi (ESP32) > WiFiScan**

# Arduino IDE Environment



See: <http://arduino.cc/en/Guide/Environment> for more information

# Serial Port (Console Port) and (Development) Board

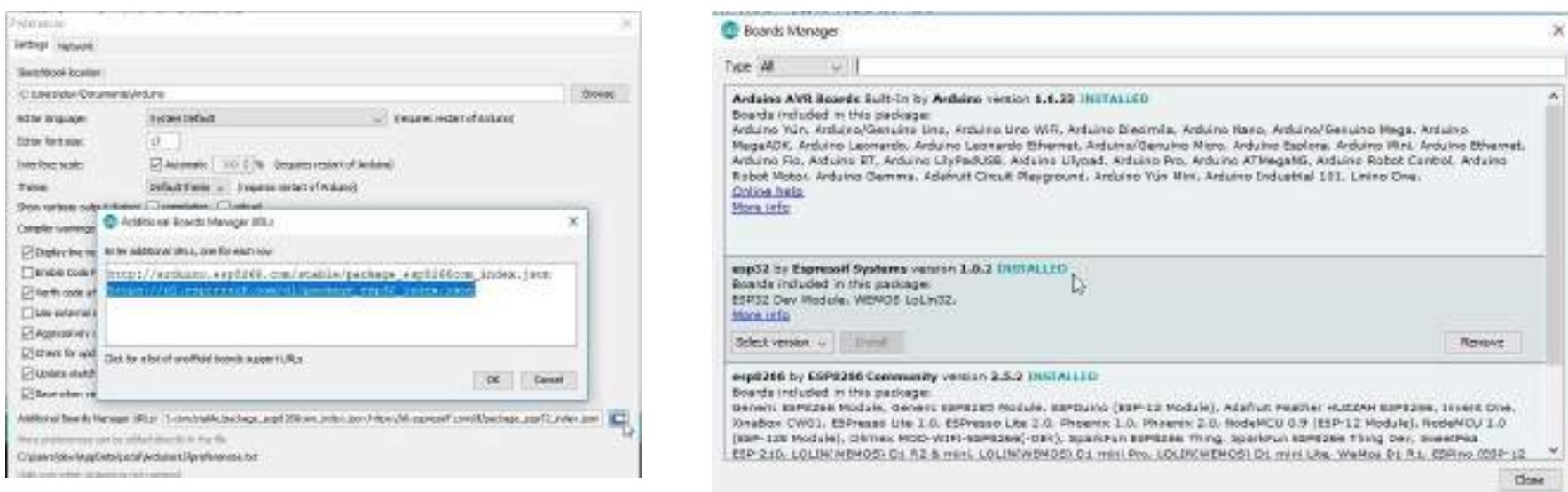




# ESP32 Installation on Arduino IDE

## Step 1

- Download Arduino IDE from <https://www.arduino.cc/en/Main/Software>
- Download and run installer software or download zip and install from an install file from unzip folder
- Install USB driver for NodeMCU-32S or other ESP32\* boards – “CP210x”  
<https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers>
- Install Arduino Core for ESP32
  - File-> Preferences: [https://dl.espressif.com/dl/package\\_esp32\\_index.json](https://dl.espressif.com/dl/package_esp32_index.json)
  - Tools-> Manage Libraries: search ESP32 (by Espressif System)

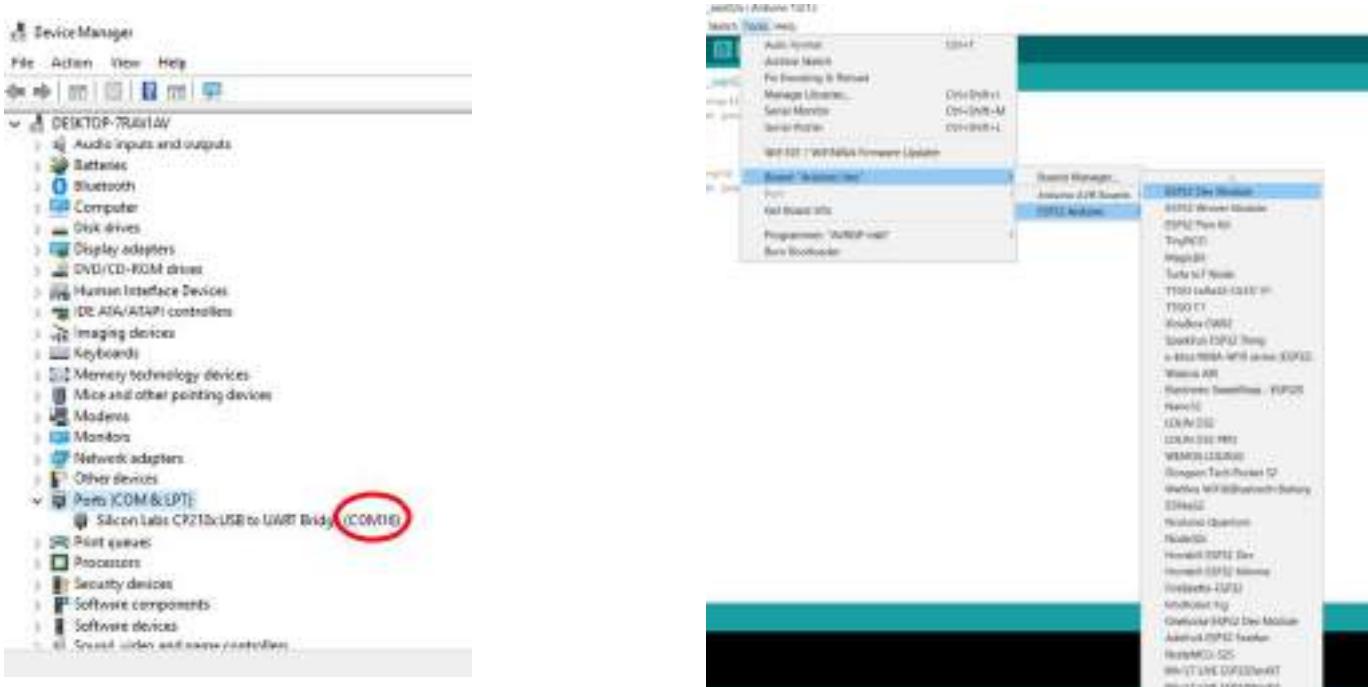


Ref: <https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/>



## Step 2: Check/Choose Port and Board

Tools->Board->ESP32->ESP32 Dev Module





## Step 3: Create a New folder and save a your example program over there.

int LED\_BUILTIN = 2;

Blink

File Edit Sketch Tools Help

New Ctrl+N  
Open... Ctrl+O  
Open Recent  
Sketchbook Examples

Close Ctrl+W  
Save Ctrl+S  
Save As... Ctrl+Shift+S  
Page Setup Ctrl+Shift+P  
Print Ctrl+P  
Preferences Ctrl+Comma  
Quit Ctrl+Q

Examples

01.Basics

AnalogReadSerial  
BlinkWithoutInterrupts

Blink

DigitalReadSerial  
Fade  
ReadAnalogVoltage

02.Digital  
03.Analog  
04.Communication  
05.Control  
06.Sensors  
07.Display  
08.Strings  
09.USB  
10.StarterKit\_BasicKit  
11.ArduinoISP

Examples for any board

Adafruit Circuit Playground

Bridge  
Explore  
Ethernet  
Firmata  
GSM  
LiquidCrystal  
Robot Control  
Robot Motor  
SD  
Servo  
Spacebrewfun  
Stepper  
Tremolo  
RETIRED

Examples for Arduino Uno

EEPROM  
SoftwareSerial

WiFiScan | Arduino 1.8.13

File Edit Sketch Tools Help

New Ctrl+N  
Open... Ctrl+O  
Open Recent  
Sketchbook Examples

Close Ctrl+W  
Save Ctrl+S  
Save As... Ctrl+Shift+S  
Page Setup Ctrl+Shift+P  
Print Ctrl+P  
Preferences Ctrl+Comma  
Quit Ctrl+Q

Bridge  
Ethernet  
firmata  
LiquidCrystal  
SD  
Stepper  
Tremolo  
RETIRED

Transfers to ESP32 Dev Module

WiFiScan (WIFI\_STA);  
WiFi.disconnect();  
delay(1000);

Serial.println(F("Hello"));

void loop()

{

Serial.println(F("Scan"));

// WiFi.scanNetworks

int n = WiFi.scanResults;

Serial.println(F("Results: "));

if (n == 0) {

Serial.println(F("No networks found"));

} else {

Serial.print(n);

Serial.println(F(" networks found"));

for (int i = 0; i < n; i++) {

// Print SSID

Serial.print(F("["));

Serial.print(i);

Serial.print(F("] "));

Serial.print(WiFi.SSID(i));

Serial.print(F(" ["));

Serial.print(WiFi.RSSI(i));

Serial.print(F(" dBm) "));

Serial.print(WiFi.channel(i));

Serial.print(F(" channel) "));

Serial.print(WiFi.encryptionType(i));

Serial.print(F(" encryption) "));

}

}

}

Learning...

Open Serial Monitor  
Setup boadrate as same as  
Serial.begin(115200);

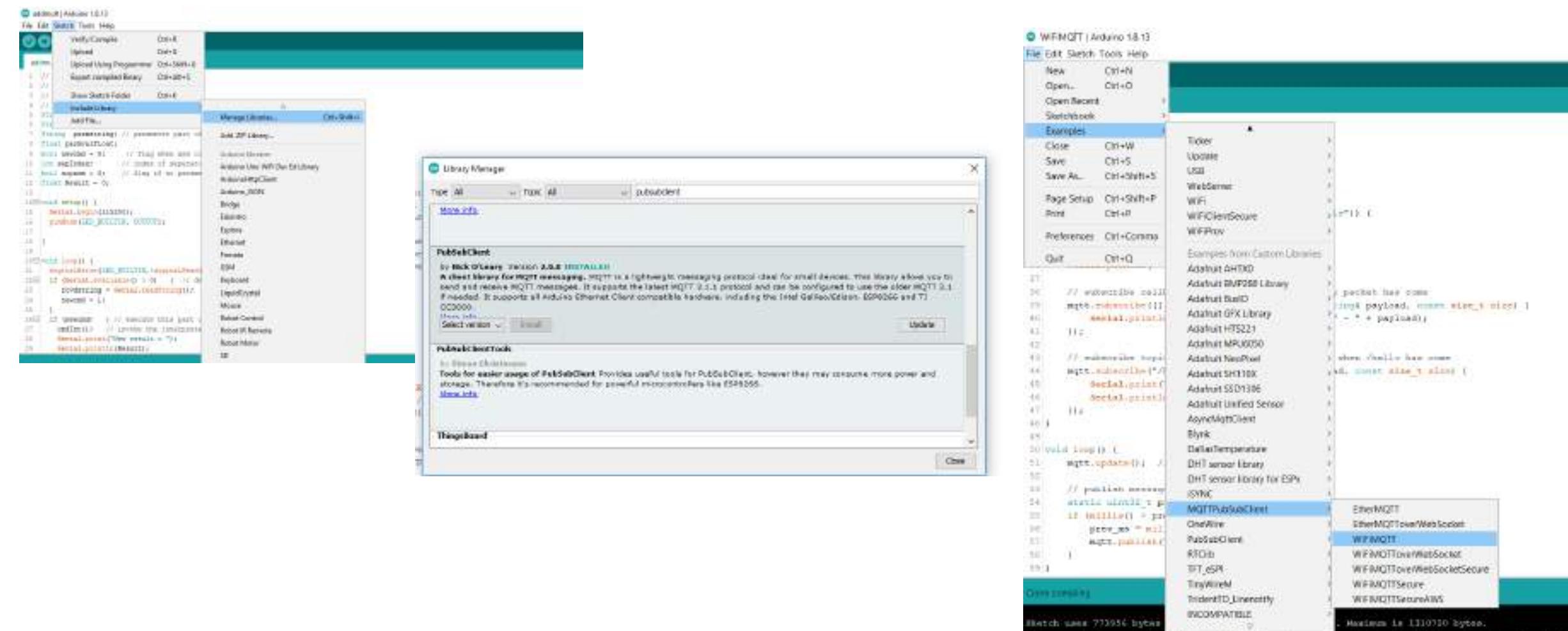




# Add New Library

## Step 4: Add PubSubClient Library + Test WiFiMQTT

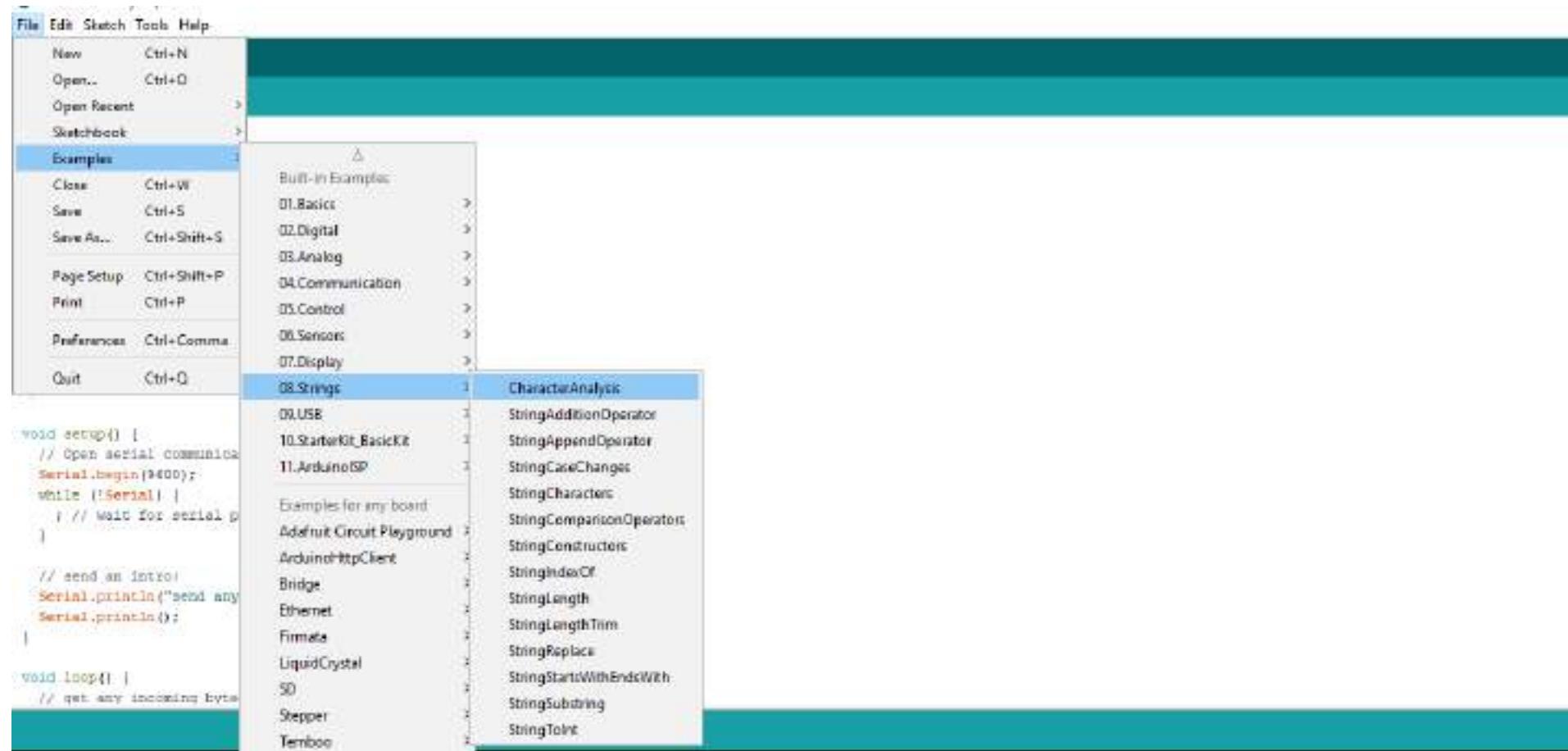
Sketch->Include Library->Manage Libraries or Add .zip library ...



# Use Serial.Print to Debug Program



## Step 5: Learn how to use Serial Print from Examples: Strings



# Digital Input/Output



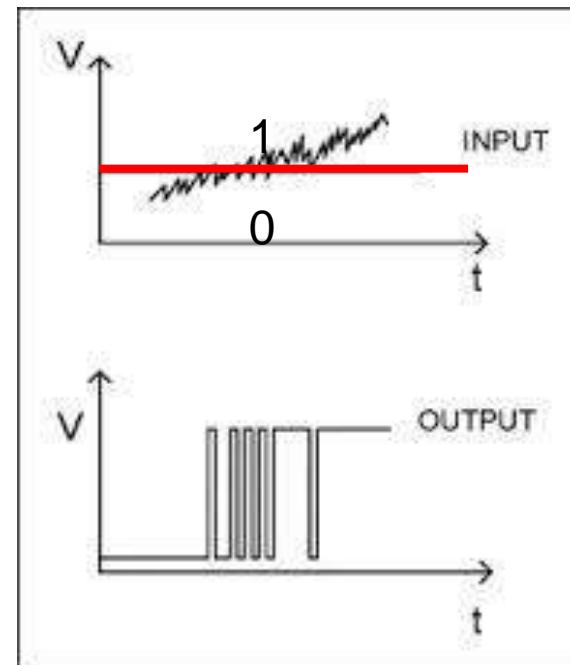
- Digital I/O is binary valued:
  - on or off, 1 or 0, HIGH or LOW
- Internally, all microprocessors are digital, **why?**
- `digitalRead(GPIO)`, `digitalWrite(GPIO)`
- For examples:

Digital Input

- Push button switch on/off
- Proximity sensor

Digital Output

- Led on/off
- Buzzer
- Relay



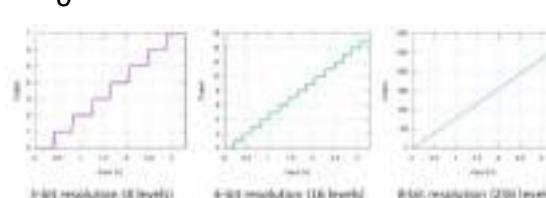
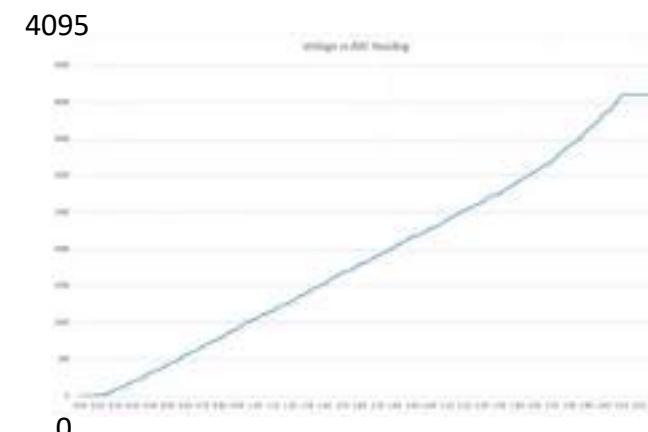
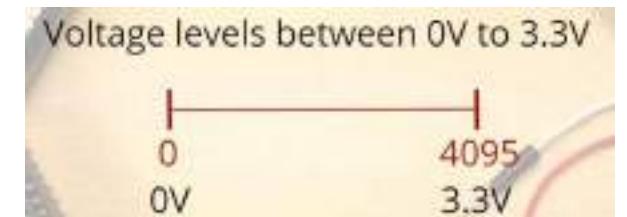
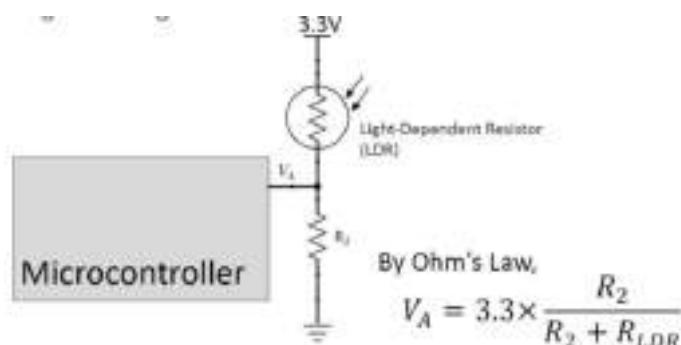
# Analog Input



- **Analog Inputs (ADC GPIOs)**

- Reading an analog value with the ESP32 means you can measure varying voltage levels between 0 V and 3.3 V.
- The voltage measured is then assigned to a value between 0 and 4095, in which 0 V corresponds to 0, and 3.3 V corresponds to 4095. Any voltage between 0 V and 3.3 V will be given the corresponding value in between.
- `analogRead(GPIO)`
- For examples:

Microphone, Potentiometer, Light sensor,  
Temperature sensor

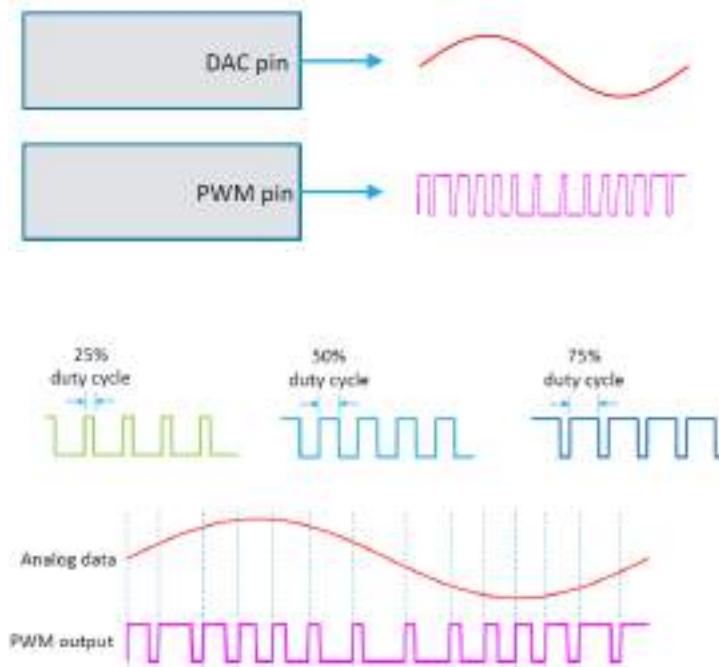


# Analog Output



- **Analog Outputs (PWM GPIOs)**

- ESP32 provides **DAC** (Digital-to-Analog Converter) pins to produce multi-level output signals
  - PWM (Pulse-Width Modulation) utilizes duty cycle control to simulate analog output
  - For examples:
    - Servo motors (control speed and rotation)
    - Dimmable LED control
- ```
led = PWM(Pin(23), freq=5000)
```



```
ledcAttachPin(X_PWM_OUT, X_PWM_CHANNEL);  
ledcSetup(X_PWM_CHANNEL, X_PWM_FREQ, PWM_RESOLUTION);
```

# Basic Arduino Programming



## void setup( ) and void loop

- An arduino program == 'sketch'
  - Must have:
    - `setup()`
    - `loop()`
  - `setup()`
    - configures pin modes and registers
  - `loop()`
    - runs the main body of the program forever
      - like `while(1) {...}`
  - Where is `main()` ?
    - Arduino simplifies things
    - Does things for you

Uses C, C++

```
/* Blink - turns on an LED for DELAY_ON msec,
then off for DELAY_OFF msec, and repeats
*/
const byte ledPin = 13; // LED on digital pin 13
const int DELAY_ON = 1000;
const int DELAY_OFF = 1000;

// setup() method runs once, when the sketch starts

void setup()
{
    // initialize the digital pin as an output:
    pinMode(ledPin, OUTPUT);
}

// loop() method runs forever,
// as long as the Arduino has power

void loop()
{
    digitalWrite(ledPin, HIGH); // set the LED on
    delay(DELAY_ON); // wait for DELAY_ON msec
    digitalWrite(ledPin, LOW); // set the LED off
    delay(DELAY_OFF); // wait for DELAY_OFF msec
}
```

# Basic Arduino Programming



## void setup()

- A digital pin can either be an output or an input

- Output

- your program determines what the voltage on a pin is (either 0V (LOW or logic 0) or 5V (HIGH or logic 1))
      - Information is sent out

- Input

- the world outside the microcontroller determines the voltage applied to the pin
      - Information is taken in

Where can you find out about the commands e.g. `digitalRead`, `digitalWrite`, etc?

<https://www.arduino.cc/reference/en/>

```
const byte ledPin = 13; // LED on digital pin 13
```

```
void setup()
```

```
{
```

```
// initialize the digital pin as an output:  
pinMode(ledPin, OUTPUT);
```

```
}
```

- `pinMode()`

- sets whether a pin is an input or an output

- `ledPin` ⇒ byte constant assigned the value of 13

- `OUTPUT` is a macro defined constant

- Which has the value 1

- `INPUT` is a macro ... ⇒ ?



## Basic Digital Pin I/O Functions

- `pinMode(pin, mode)`
  - Sets pin to INPUT or OUTPUT mode
  - Writes 1 bit in the DDRx register
- `digitalWrite(pin, value)`
  - Sets pin value to LOW or HIGH (0 or 1)
  - Writes 1 bit in the PORTx register
- `int value = digitalRead(pin)`
  - Reads back pin value (0 or 1)
  - Read 1 bit in the PINS register
- `Serial.println(value)`
  - Prints the value to the Serial Monitor on your computer

DDR = Data Direction Register

# Basic Arduino Programming



## Serial Monitor Example

```
// sketch 01_04_serial
int switchPin = 7;

void setup()
{
    pinMode(switchPin, INPUT_PULLUP);
    Serial.begin(9600);
}

void loop()
{
    if (digitalRead(switchPin) == LOW)
    {
        Serial.println("Paperclip connected");
    }
    else
    {
        Serial.println("Paperclip NOT connected");
    }
    delay(1000);
}
```

# Basic Arduino Programming



## Data Types

| Type          | Memory (bytes) | Range                            | Notes                                                                                                                                                                          |
|---------------|----------------|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| boolean       | 1              | true or false (0 or 1)           | Used to represent logical values.                                                                                                                                              |
| char          | 1              | -128 to +127                     | Used to represent an ASCII character code; for example, A is represented as 65. Negative numbers are not normally used.                                                        |
| byte          | 1              | 0 to 255                         | Often used for communicating serial data, as a single unit of data. See Chapter 9.                                                                                             |
| int           | 2              | -32768 to +32767                 | These are signed 16 bit values.                                                                                                                                                |
| unsigned int  | 2              | 0 to 65536                       | Used for extra precision when negative numbers are not needed. Use with caution as arithmetic with <code>ints</code> may cause unexpected results.                             |
| long          | 4              | 2,147,483,648 to 2,147,483,647   | Needed only for representing very big numbers.                                                                                                                                 |
| unsigned long | 4              | 0 to 4,294,967,295               | See <code>unsigned int</code> .                                                                                                                                                |
| float         | 4              | -3.4028235E+38 to +3.4028235E+38 | Used to represent floating point numbers.                                                                                                                                      |
| double        | 4              | as float                         | Normally, this would be 8 bytes and higher precision than <code>float</code> with a greater range. However, on Arduino <code>double</code> is the same as <code>float</code> . |

# Basic Arduino Programming



## Arduino Commands (1)

| Command      | Example                                     | Description                                                                                                                                                                           |
|--------------|---------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Digital I/O  |                                             |                                                                                                                                                                                       |
| pinMode      | <code>pinMode(8, OUTPUT);</code>            | Sets pin 8 to be an output. The alternative is to set it to be INPUT or INPUT_PULLUP.                                                                                                 |
| digitalWrite | <code>digitalWrite(8, HIGH);</code>         | Sets pin 8 high. To set it low, use the constant LOW instead of HIGH.                                                                                                                 |
| digitalRead  | <code>int i;<br/>i = digitalRead(8);</code> | Sets the value of i to HIGH or LOW, depending on the voltage at the pin specified (in this case, pin 8).                                                                              |
| pulseIn      | <code>i = pulseIn(8, HIGH)</code>           | Returns the duration in microseconds of the next HIGH pulse on pin 8.                                                                                                                 |
| tone         | <code>tone(8, 440, 1000);</code>            | Makes pin 8 oscillate at 440 Hz for 1000 milliseconds.                                                                                                                                |
| noTone       | <code>noTone();</code>                      | Cuts short the playing of any tone that was in progress.                                                                                                                              |
| Analog I/O   |                                             |                                                                                                                                                                                       |
| analogRead   | <code>int r;<br/>r = analogRead(0);</code>  | Assigns a value to r of between 0 and 1023; 0 for 0V, 1023 if pin0 is 5V (3.3V for a 3V board).                                                                                       |
| analogWrite  | <code>analogWrite(9, 127);</code>           | Outputs a PWM signal. The duty cycle is a number between 0 and 255, 255 being 100%. This must be used by one of the pins marked as PWM on the Arduino board (3, 5, 6, 9, 10, and 11). |



## Arduino Commands (2)

| Time Commands              |                                                          |                                                                                                                                                                                                                    |
|----------------------------|----------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| millis                     | <code>unsigned long l;<br/>l = millis();</code>          | The variable type <b>long</b> in Arduino is represented in 32 bits. The value returned by <b>millis()</b> is the number of milliseconds since the last reset. The number wraps around after approximately 50 days. |
| micros                     | <code>long l;<br/>l = micros();</code>                   | See <b>millis</b> , except this is microseconds since the last reset. It wraps after approximately 70 minutes.                                                                                                     |
| delay                      | <code>delay(1000);</code>                                | Delays for 1000 milliseconds or 1 second.                                                                                                                                                                          |
| delayMicroseconds          | <code>delayMicroseconds(100000);</code>                  | Delays for 100,000 microseconds. Note the minimum delay is 3 microseconds; the max is around 16 milliseconds.                                                                                                      |
| Interrupts (see Chapter 3) |                                                          |                                                                                                                                                                                                                    |
| attachInterrupt            | <code>attachInterrupt(1,<br/>myFunction, RISING);</code> | Associates the function <b>myFunction</b> with a rising transition on interrupt 1 (D3 on an Uno).                                                                                                                  |
| detachInterrupt            | <code>detachInterrupt(1);</code>                         | Disables any interrupt on interrupt 1.                                                                                                                                                                             |

# Basic Arduino Programming



## General Arduino Commands (3)

### Structure

```
void setup()
void loop()
```

### Control Structures

```
if(x > 1)
for(i = 0; i < 255; i++)
while(x < 0)
```

### Further Syntax

Single line comment:  
// ...  
Multi line comment:  
/\* ... \*/  
Define ANSWER 42  
#include <avr-libc.h>

### General Operators

- assignment  
+ - addition, subtraction  
\* / multiplication, division  
% modulo  
== equal to  
!= not equal to  
< less than  
<= less than or equal to

### Pointer Access

\* reference operator  
-> dereference operator

### Bitwise Operators

& bitwise AND  
| bitwise OR  
^ bitwise XOR  
~ bitwise NOT

### Compound Operators

++ Increment  
-- Decrement  
+= Compound addition  
&= Compound bitwise AND

### Constants

HIGH, LOW  
true, false  
3.14... Decimal  
01001000 : Binary  
0x5BA4 : Hexadecimal

### Data Types

bool boolean 0, 1, false, true  
char e.g. 'a' -128 → 127  
signed char 0 → 255  
int -32,768 → 32,767  
unsigned int 0 → 65,535  
long -2,147,483,648 → 2,147,483,647  
float -3.402823E-38 → 3.402823E+38  
sizeof (inbytes) returns 2 bytes

### Arrays

```
int myInts[5];
int myBins[] = {2,4,8,5,6};
int myVals[5] = {2,4,9,3,5};
```

### Strings

```
char S1[5];
char S2[5] = 'A','B','C','D','\0';
char S3[8] = 'A','T','G','C','T','A','T','\0';
char S4[] = "Arduino";
char S5[8] = {"Arduino"};
char S6[15] = "Arduino";
```

### Conversion

```
char() int() long()
byte() word() float()
```

### Qualifiers

static Persist between calls  
volatile Use RAM (not for ISR)  
const Mark read-only  
PROGMEM Use flash memory

### Interrupts

```
attachInterrupt(interrupt, function, type)
detachInterrupt(interrupt)
boolean (interrupt)
interrupts()
noInterrupts()
```

### Advanced I/O

tone(pin, freq);
tone(pin, freq, duration, ms);
softTone(pin);
shiftOut(dataPin, clockPin, how, value);
signed long pulseInPin, [HIGH,LOW];

### Time

unsigned long millis();
30 days overflow
unsigned long micros();
20 min. overflow
delay(ms);
delayMicroseconds(us);

### Math

min(x,y);
max(x,y);
abs(x);
sin(rad);
cos(rad);
tan(rad);
pow(base,exponent);
expval, fromL, fromH, toL, toH;
sqrt(max,fromL,toH);

### Pseudo Random Numbers

```
randomSeed(seed)
long random(max)
long random(min, max)
```

### ATmega328 Pinout



### IO Pins

|                   | Uno             | Mega            |
|-------------------|-----------------|-----------------|
| # of IO           | 14 + 6          | 54 + 11         |
| Serial Pins 3     | 0 - RX, 1 - TX  | RX1 → RX4       |
| Interrupts        | 2,3             | 2,3,18,19,20,21 |
| PWM Pins          | 3,6 - 9,10 - 13 | 0 - 13          |
| SPI Communication | 10 - 13         | 50 - 53         |
| I2C               | A4, A5          | 20,21           |

### Analog I/O

```
analogReference(INTERNAL, INTERNAL)
analogRead(pin)
analogWrite(pin, value)
```

### Digital I/O

```
pinMode(pin, [INPUT, OUTPUT])
digitalRead(pin)
digitalWrite(pin, value)
```

### Serial Communication

```
Serial.begin(baud)
Serial.print("test")
Serial.println("test")
```

### Websites

CODE: [arduino.cc](http://arduino.cc)  
playground.arduino.cc  
[arduino.cc/en/FaQ](http://arduino.cc/en/FaQ)

### Arduino Uno Board



See more commands from: [http://www.sel.eesc.usp.br/jcarmo/pdfs/Arduino\\_SimonMonk\\_2011.pdf](http://www.sel.eesc.usp.br/jcarmo/pdfs/Arduino_SimonMonk_2011.pdf)

# Basic Arduino Programming



## Blinking the LED in loop()

- `digitalWrite()`
  - Causes the voltage on the indicated pin to go HIGH (+5V) or LOW (0V)
  - Note: must first configure the pin to be an output
    - To make pin go to 5V (high):
      - `digitalWrite(pin_num,HIGH);`  
» Best to `#define` pin num.
    - To make pin go to 0V (low):
      - `digitalWrite(pin_num,LOW);`
- `delay()`
  - Causes the program to wait for a specified time in milliseconds

```
#define LED_PIN 13 // LED on digital pin 13
#define DELAY_ON 500 // in ms
#define DELAY_OFF 100

void setup()
{
    // initialize the digital pin as an output:
    pinMode(LED_PIN, OUTPUT);
}

void loop()
{
    digitalWrite(LED_PIN, HIGH); // turn LED on
    delay(DELAY_ON); // wait for DELAY_ON ms
    digitalWrite(LED_PIN, LOW); // turn LED off
    delay(DELAY_OFF); // wait for DELAY_OFF ms
}
```

<http://arduino.cc/en/Reference/Extended>

# Basic Arduino Programming



## Using Switches and Buttons

```
const int inputPin = 2; // choose the input pin

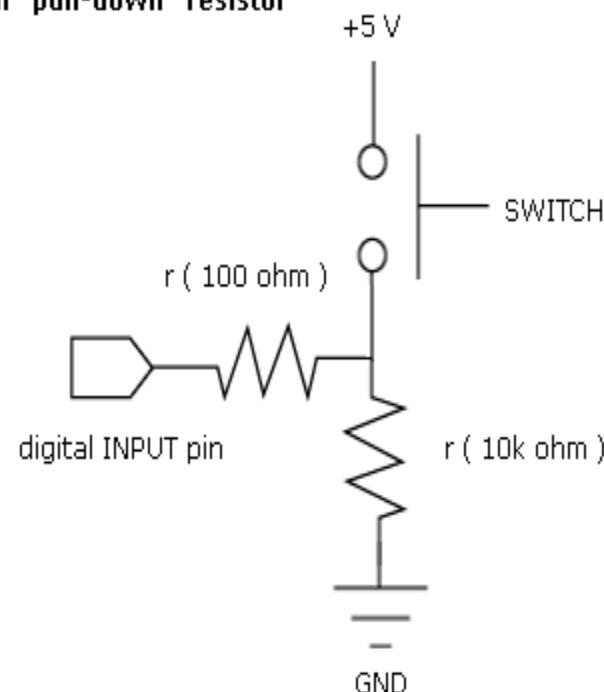
void setup() {
    pinMode(inputPin, INPUT); // declare pushbutton as input
}

void loop(){
    int val = digitalRead(inputPin); // read input value
}
```

# Pull-down/ Pull-up Resistor Concept



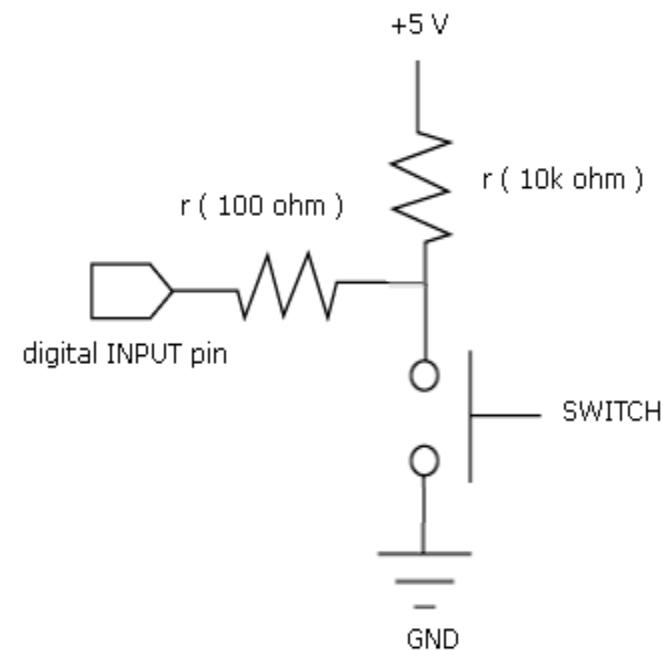
Switch with "pull-down" resistor



Button NOT Press: 0

Button Press:

Switch with "pull-up" resistor



Button NOT Press: 1

Button Press:

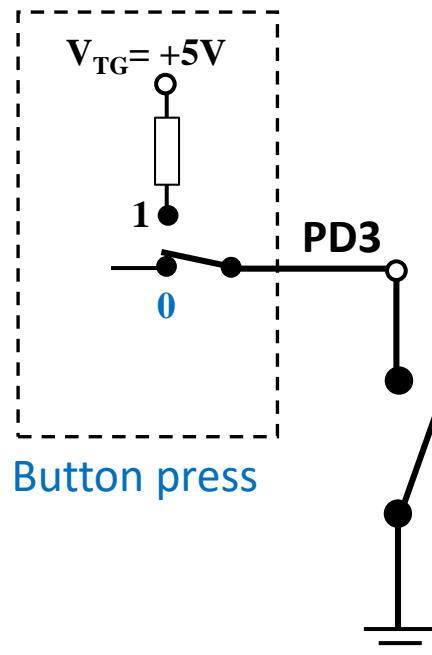
0

# Pull-up Resistor Concept



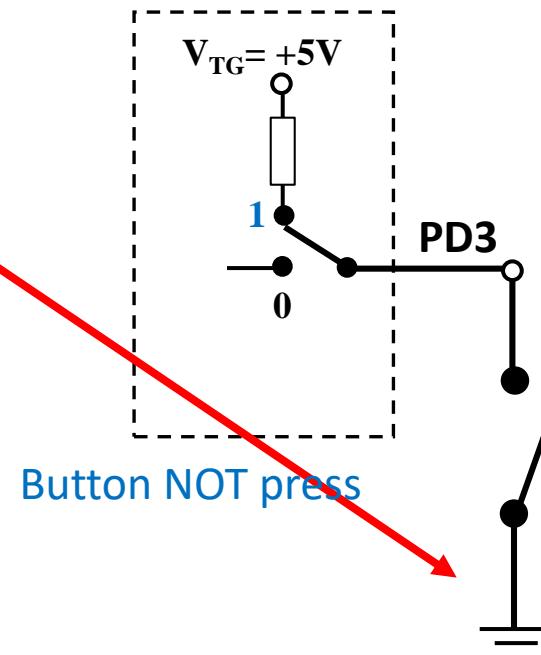
**Pull-up resistor OFF**

ATmega328



**Pull-up resistor ON**

ATmega328

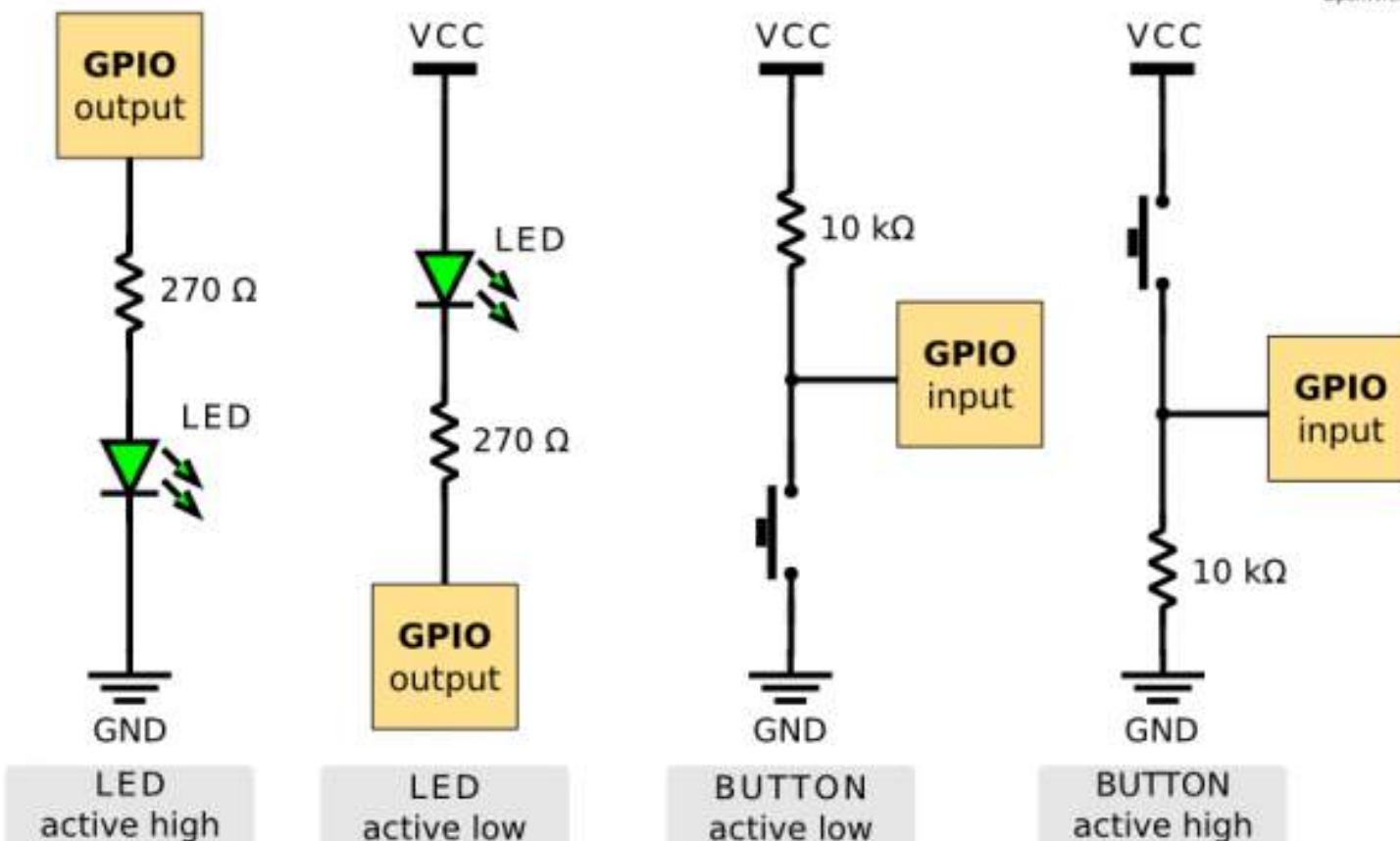


The AVR chip has, internally, a 20 kOhm pull-up resistor that can be connected to the pin (internally). The pin must be configured as input, and its value, in this situation, tells whether the pull-up is connected (otherwise the value defines, when the pin is configured as output, its output state).

# GPIO: Active Low/Active High



OpenWrt.org



# Pushbutton Switch or Momentary Button



- A momentary button is a “Biased Switch”
- Pushing the button changes state
- State is reversed (return to biased position) when button is released
- Two types
  - NO: normally open
  - NC: normally closed

Normally Open



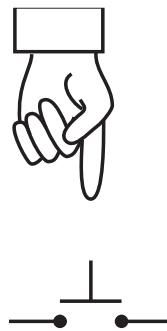
Normally Closed



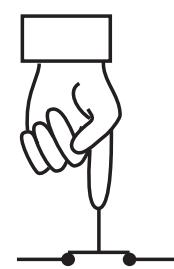
# Pushbutton Switch or Momentary Button



- **Normally open**
  - electrical *contact is made* when button is pressed
- **Normally closed**
  - electrical *contact is broken* when button is pressed
- Internal spring returns button to its un-pressed state



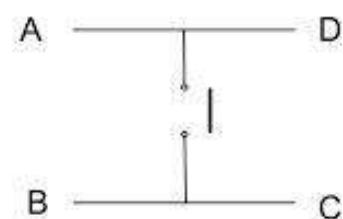
Open



Closed



Image from sparkfun.com



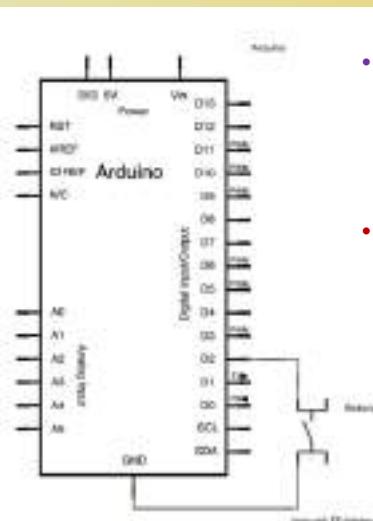
**pinMode(GPIO, INPUT\_PULLUP)**



## Example 1:

```
void setup() {
  //start serial connection
  Serial.begin(9600);
  pinMode(2, INPUT_PULLUP); //configure pin2 as an input and enable the internal pull-up resistor
  pinMode(13, OUTPUT);
}
void loop() {
  int sensorVal = digitalRead(2); //print out the value of the pushbutton
  Serial.println(sensorVal);
  // Keep in mind the pullup means the pushbutton's logic is inverted.
  // It goes HIGH when it's open, and LOW when it's pressed.
  // Turn on pin 13 when the button's pressed, and off when it's not:
  if (sensorVal == HIGH) {
    digitalWrite(13, LOW);
  } else {
    digitalWrite(13, HIGH);
  }
}
```

[https://circuits.io/circuits/2678082-arduino-input\\_pullup](https://circuits.io/circuits/2678082-arduino-input_pullup)



- Connect two wires to the Arduino board.
    - The **black wire** connects **ground** to one leg of the pushbutton.
    - The **second wire** goes from digital **pin 2** to the other leg of the pushbutton.
  - Pushbuttons or switches connect **two points** in a circuit when you press them.
    - When the pushbutton is open (unpressed) there is no connection between the two legs of the pushbutton. Because the internal pull-up on pin 2 is active and connected to 5V, we read HIGH when the button is open.
    - When the button is closed, the Arduino reads LOW because a connection to ground is completed.

<https://www.arduino.cc/en/Tutorial/InputPullupSerial>

**pinMode(GPIO, INPUT\_PULLUP)**



## Example 2: handleSW() function

Objective: This program is to design for having a pushbutton as input for turning off and on an LED.

Pin 13 is the onboard LED

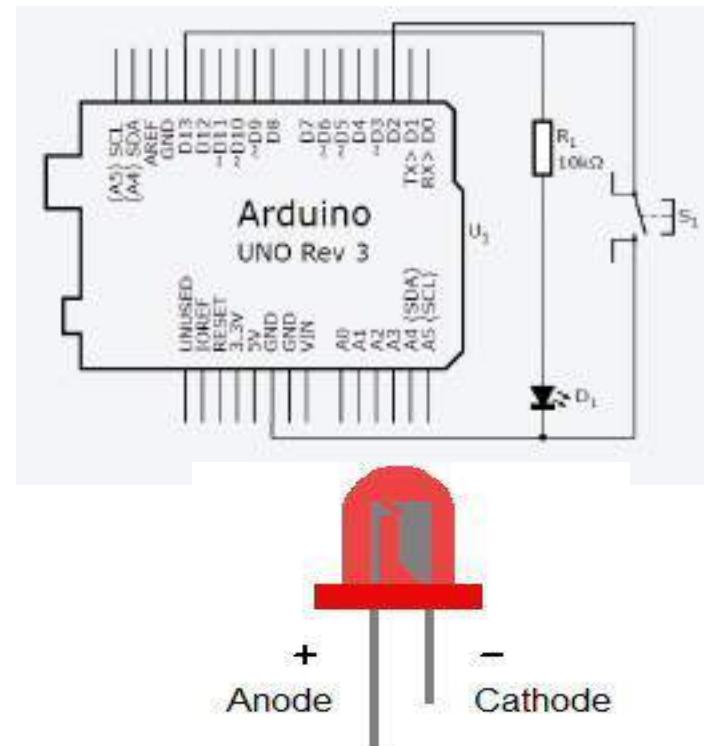
Pin 2 has a NO momentary switch connected to ground. Note that the internal pullup is enabled.

```
const byte LED = 13, SW = 2;

void setup() {
    pinMode(LED, OUTPUT); //toggles an LED
    pinMode(SW, INPUT_PULLUP);
    //reads a switch
}

void handleSW() {
    digitalWrite(LED, digitalRead(SW));
    //reads pin 2 and writes the result to pin 13
}

void loop() {
    handleSW();
}
```



[https://circuits.io/circuits/2679697-arduino\\_handle\\_sw](https://circuits.io/circuits/2679697-arduino_handle_sw)

## Example 3: handleSW() function

```
const byte LED = 13, SW = 2;

void handleSW() {
    digitalWrite(LED, digitalRead(SW));
}

void handleOtherStuff() {
    delay(250);
}

void setup() {
    pinMode(LED, OUTPUT);
    pinMode(SW, INPUT_PULLUP);
}

void loop() {
    handleSW();
    handleOtherStuff();
}
```

# Button Switch Operations



## Example: user input features of the fan

- Potentiometer for speed control
  - Continually variable input makes sense for speed control
- Start/stop
  - Could use a conventional power switch
  - Push button (momentary) switch
- Lock or limit rotation angle
  - Button click to hold/release fan in one position
  - Potentiometer to set range limit

# Putting Buttons Into Action

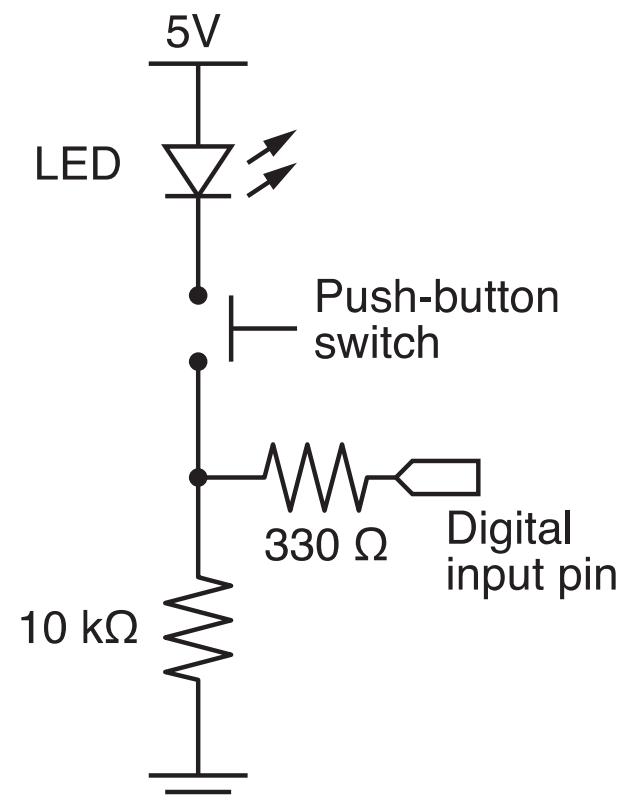


- Build the circuit: same one is used for all examples
  - Test with LED on/off
  - LED is only controlled by the button, not by Arduino code
- Create a “wait to start” button
  - Simplest button implementation
  - Execution is blocked while waiting for a button click
- Use an “interrupt handler”
  - Most sophisticated: Don’t block execution while waiting for button input
  - Most sophisticated: Requires good understanding of coding
  - Requires “de-bouncing”
  - Not too hard to use as a black box

# Momentary Button and LED Circuit



- Digital input with a pull-down resistor
  - When switch is open (button not pressed):
    - Digital input pin is tied to ground
    - No current flows, so there is no voltage difference from input pin to ground
    - Reading on digital input is LOW
  - When switch is closed (button is pressed):
    - Current flows from 5V to ground, causing LED to light up.
    - The 10k resistor limits the current draw by the input pin.
    - The  $330\Omega$  resistor causes a large voltage drop between 5V and ground, which causes the digital input pin to be closer to 5V.
    - Reading on digital input is HIGH



# Momentary Button



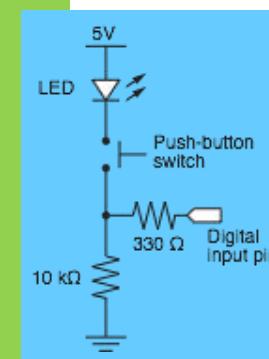
## Continuous Monitor of Button State

```
int button_pin = 4;           // pin used to read the button

void setup() {
    pinMode(button_pin, INPUT);
    Serial.begin(9600);          // Button state is sent to host
}

void loop() {
    int button;
    button = digitalRead(button_pin);

    if (button == HIGH) {
        Serial.println("on");
    } else {
        Serial.println("off");
    }
}
```



Serial monitor shows a continuous stream of “on” or “off”

This program *does not* control the LED

# Momentary Button



## Wait for Button Input

```
int button_pin = 4;           // pin used to read the button

void setup() {
    int start_click = LOW;      // Initial state: no click yet
    pinMode( button_pin, INPUT );
    Serial.begin(9600);

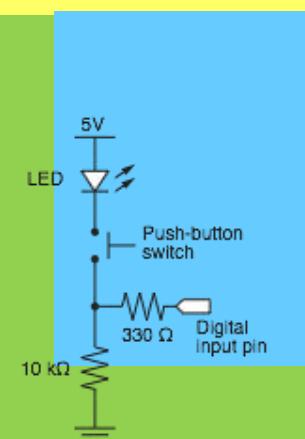
    while ( !start_click ) {
        start_click = digitalRead( button_pin );
        Serial.println("Waiting for button press");
    }
}

void loop() {
    int button;

    button = digitalRead( button_pin );
    if ( button == HIGH ) {
        Serial.println("on");
    } else {
        Serial.println("off");
    }
}
```

Same loop() function as in the preceding sketch

while loop continues as long as start\_click is FALSE



- [Mar11] Michael Margolis, **Arduino Cookbook**, O'Reilly, 2011
- [Mon18] Simon Monk, **Programming Arduino Next Steps: Going Further with Sketches**, 2<sup>nd</sup> Edition, 2018.
- [OK13] Robert Oshana, Mark Kraeling, **Software Engineering for Embedded Systems**, e-ISBN: 978-0-12-415941-9, Newnes, 2013.
- [SS\*] Rui Santos and Sara Santos, **Learn ESP32 with Arduino IDE**, v1.4
- [Whi11] Elecia White, **Making Embedded Systems**, O'Reilly, 2011
- <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/>
- <https://randomnerdtutorials.com/getting-started-with-esp32/>
- <https://mjrobot.org/iot-made-simple-playing-with-the-esp32-on-arduino-ide/>
- <https://www.instructables.com/IOT-Made-Simple-Playing-With-the-ESP32-on-Arduino-/>
- <http://www.energazero.org/meccatronica/IOT-Made-Simple-Playing-With-the-ESP32-on-Arduino-.pdf>



# **ITCS447**

## **Lecture 4/8**

# **Multitasking Programming**

## **Interrupt-FSM-PID**

Asst. Prof. Dr. Thitinan Tantidham



มหาวิทยาลัยมหิดล  
Mahidol University

ห้ามมิให้นักศึกษาทำข้า ดัดแปลง หรือใช้ประโยชน์จากการอันมีลิขสิทธิ์ปราภูอยู่ในระบบการศึกษาอิเล็กทรอนิกส์ (E-Learning) ของมหาวิทยาลัย ไม่ว่าจะทั้งหมดหรือบางส่วน โดยไม่ได้รับอนุญาตจากมหาวิทยาลัย นอกจากนี้จากการศึกษาส่วนบุคคล ทั้งนี้ การทำข้า ดัดแปลง หรือเผยแพร่ต่อสาธารณะชื่องานอันมีลิขสิทธิ์ จะมีโทษปรับตั้งแต่ 20,000 บาท ถึง 200,000 บาท และหากเป็นการกระทำเพื่อการค้า จะมีโทษจำคุกตั้งแต่ 6 เดือน ถึง 10 ปี หรือปรับตั้งแต่ 100,000 บาท ถึง 800,000 บาท หรือทั้งจำทั้งปรับ

Unless allowed by Mahidol University, the User shall not copy, modify, or exploit in part or in whole of the copyrighted materials on the Platform, other than for your own individual study. The copy, modify or communication to public of the copyrighted materials shall be inflicted with a fine from 20,000 Baht up to 200,000 Baht. If the offence is committed with the commercial purpose, the offender shall be inflicted with imprisonment for a term from 6 months up to 10 years or a fine from 100,000 Baht up to 800,000 Baht or both imprisonment and fine.



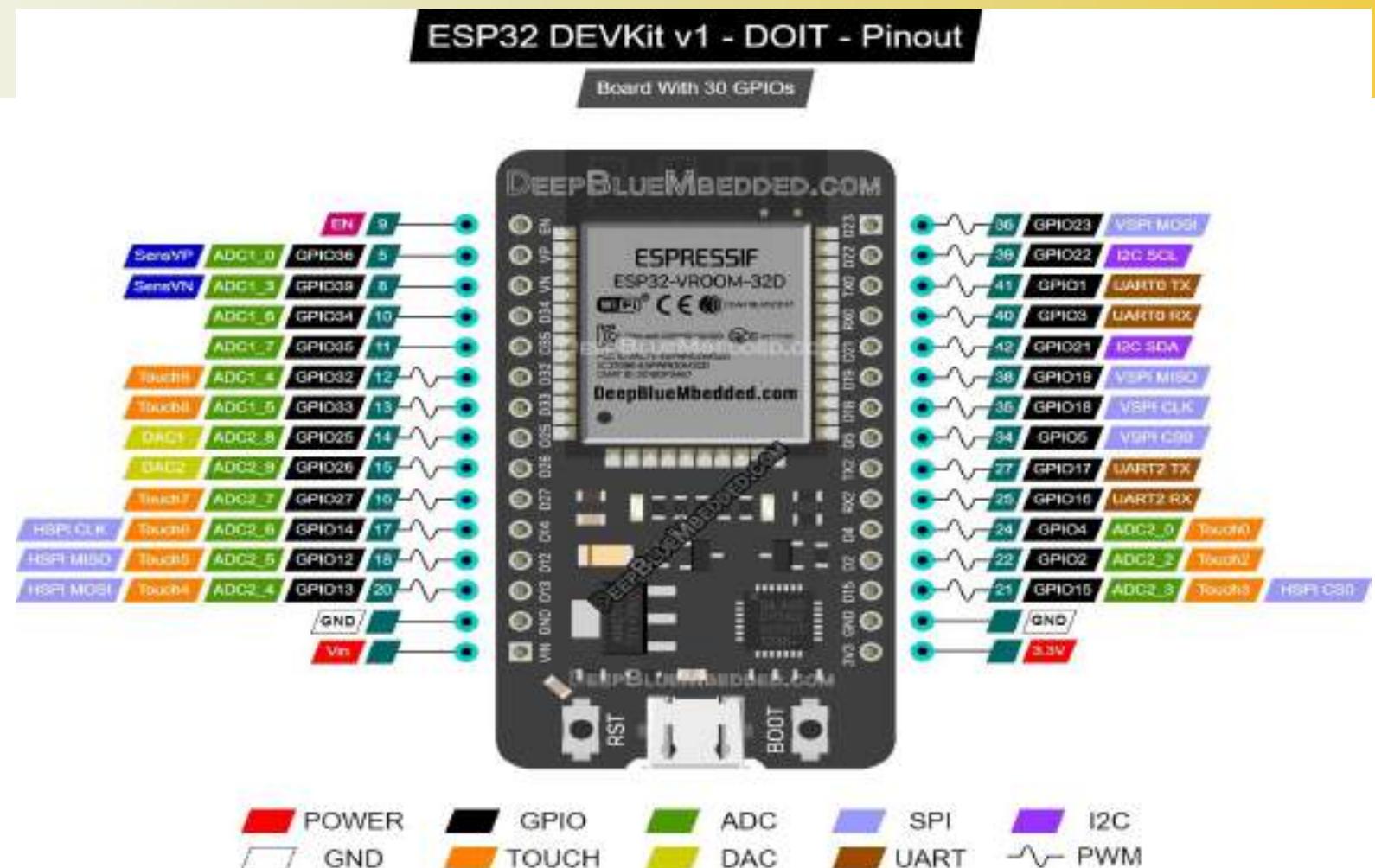
- Interrupt
- Finite State Machine
- Proportional Integral Derivative (PID)



# Interrupt

- Review: ESP32 Pin Layout
- What is Interrupt?
- ISR
- Interrupts in ESP32
- External Interrupts on Different Microcontrollers
- External Interrupt Programming:  
`attachInterrupt(GPIO, ISR, mode)`
- Examples

# Review: ESP32 Pin Layout



38 Pins including 3v3, En, 5v, 3xGNDs  
32 Pins for GPIOs



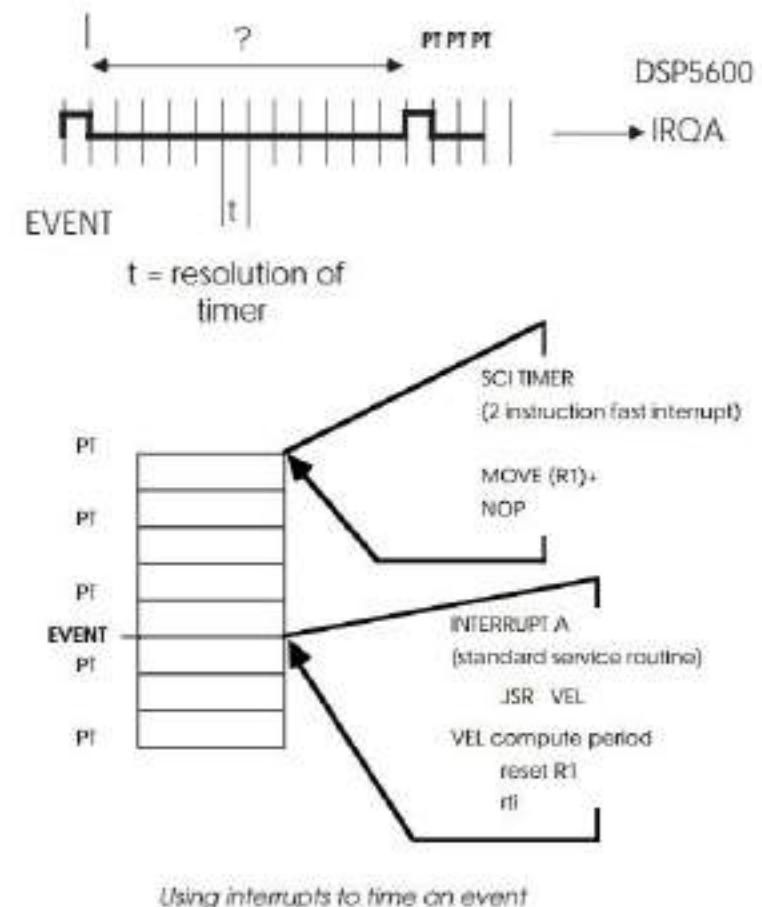
# What is Interrupt?

- An interrupt is a signal that interrupts the current processor activity.
- It may be triggered by an **external event** (change in pin state) or an **internal event** (a timer or a software signal).
- Once triggered, an interrupt pauses the current activity and causes the program to execute a different function or jump into their ISR function.
  - Interrupt Service Routine (ISR) function is also called an interrupt handler.
- Once the function is completed, the program returns to what it was doing before the interrupt was triggered.

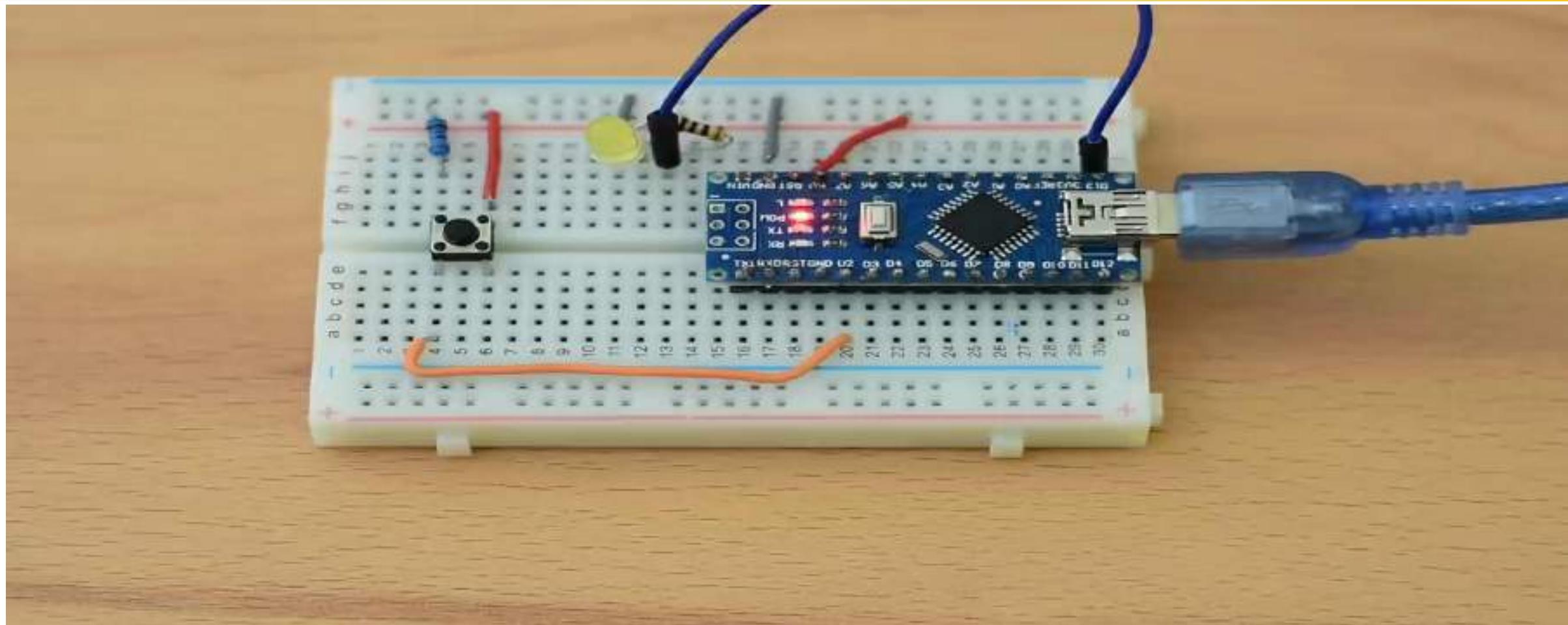
# Interrupt Service Routine (ISR)



- Interrupts are used to detect important real-time events, which occur during the normal code execution of the code, without continuously checking them, like pushing a button.
- ISR should be as short as possible and the return type of it is void.
- ISR should have the `IRAM_ATTR` attribute to declare that the compiled code will be placed in the Internal RAM (IRAM) in order to execute quickly as possible.
  - Some of normal functions do not work or behave differently in the ISR, for example, `delay()` function does not work in the ISR.
  - `delay()` is a blocking function. Blocking functions prevent a program from doing anything else until that particular task is completed. If you need multiple tasks to occur at the same time, you cannot use `delay()`, but uses `millis()` function instead
  - Variables used in the ISR must be volatile variables.



# Interrupts





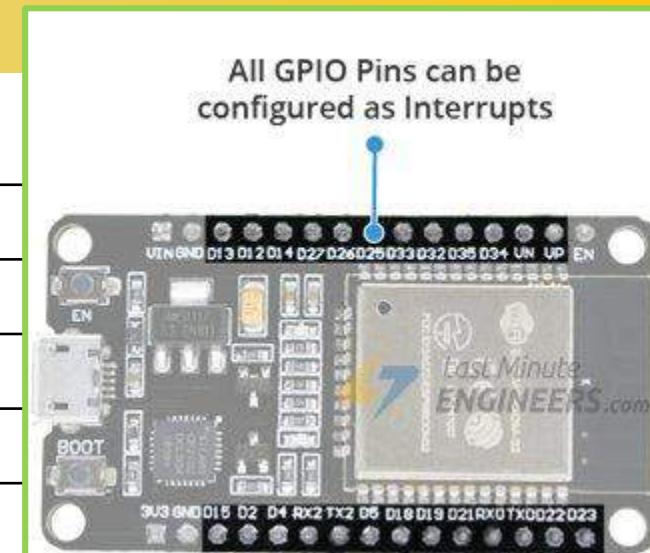
- The ESP32 offers up to 32 interrupt slots for each core.
- Each interrupt has a certain priority level and can be categorized into two types:
  - **Hardware Interrupts**
    - These occur in response to an external event by hardware peripherals or sources, external GPIO pins.
    - For example, GPIO Interrupt(when a key is pressed down) or a Touch Interrupt(when touch is detected)
  - **Software Interrupts**
    - These occur in response to a software instruction by user or programmer
    - For example, a Simple Timer Interrupt or WatchDog Timer (WDT) Interrupt(when timer times out)

# External Interrupts



## Microcontrollers

| Arduino Boards                    | Digital Pins Usable For Interrupts                                               |
|-----------------------------------|----------------------------------------------------------------------------------|
| Uno, Nano, Mini, other 328-based  | 2, 3                                                                             |
| Uno WiFi Rev.2, Nano Every        | all digital pins                                                                 |
| Mega, Mega2560, MegaADK           | 2, 3, 18, 19, 20, 21                                                             |
| Micro, Leonardo, other 32u4-based | 0, 1, 2, 3, 7                                                                    |
| Zero                              | all digital pins, except 4                                                       |
| MKR Family boards                 | 0, 1, 4, 5, 6, 7, 8, 9, A1, A2                                                   |
| Nano 33 IoT                       | 2, 3, 9, 10, 11, 13, 15, A5, A7                                                  |
| Nano 33 BLE, Nano 33 BLE Sense    | all pins                                                                         |
| Due                               | all digital pins                                                                 |
| 101                               | all digital pins (Only pins 2, 5, 7, 8, 10, 11, 12, 13 work with <b>CHANGE</b> ) |



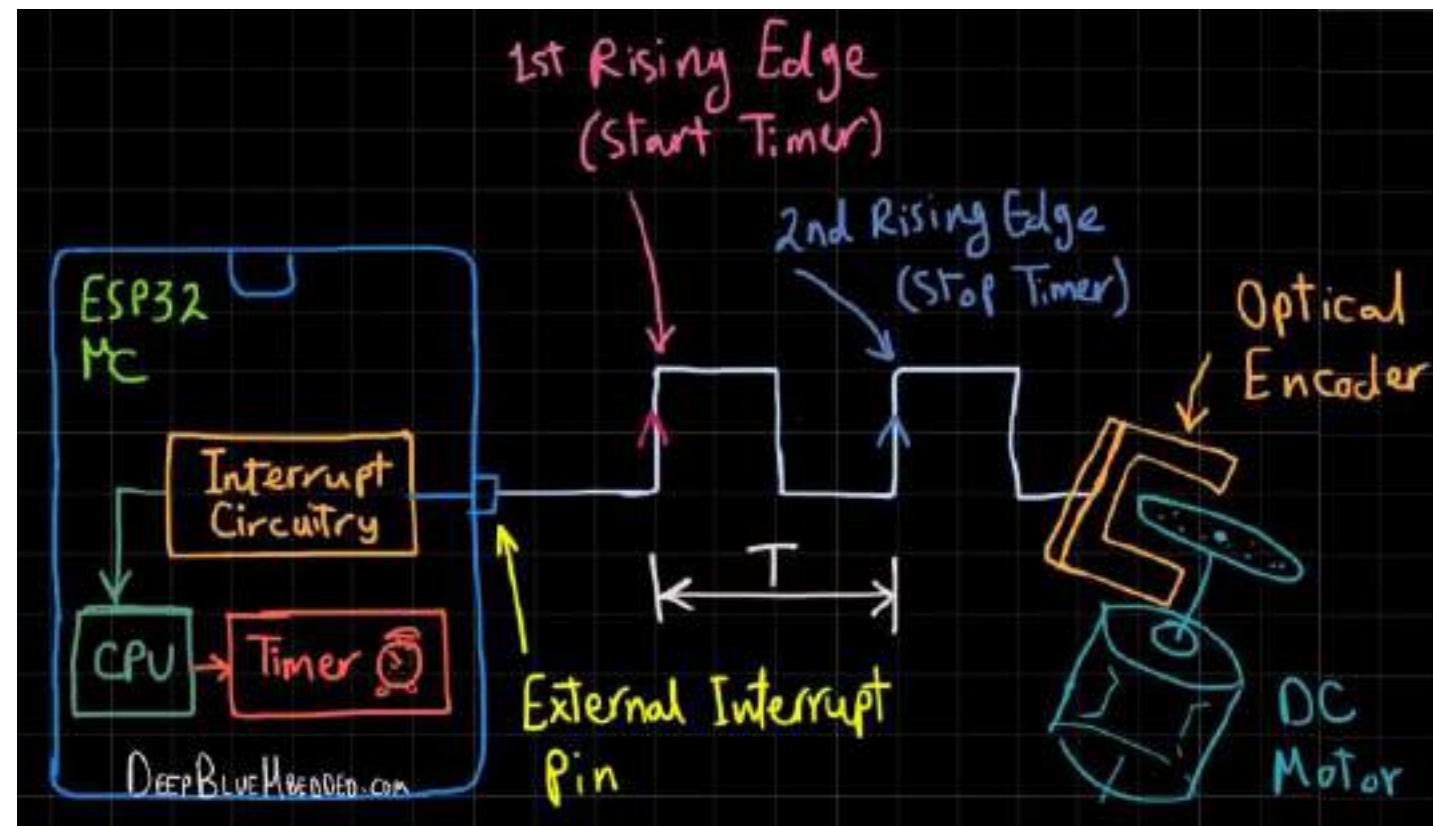
<https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/>



# External Interrupts

## Example: To measure frequency of the motor rotation

- As you can see in the diagram, the motor rotation causes the optical encoder to generate a square wave signal. By reading (measuring) the frequency of this signal, we can deduce the motor's speed (RPM).
- The digital signal is being measured in this application example by using an external interrupt pin + a Timer module.
- On the first rising edge, an interrupt occurs, so the CPU suspends the main program execution and starts a timer, then it resumes back the main program.
- On the next rising edge, an interrupt is fired, the CPU suspends the main program and stops the timer.
- Now, the timer count can tell us the total time ( $T$ ) or period of the signal. To get the frequency we'll do ( $F = 1/T$ ) and we're done.



<https://deepbluembedded.com/esp32-external-interrupts-pins-arduino-examples/>  
<https://lastminuteengineers.com/handling-esp32-gpio-interrupts-tutorial/>

# External Interrupt Programming



```
digitalPinToInterrupt(GPIO)
attachInterrupt(GPIO, ISR, mode)
```

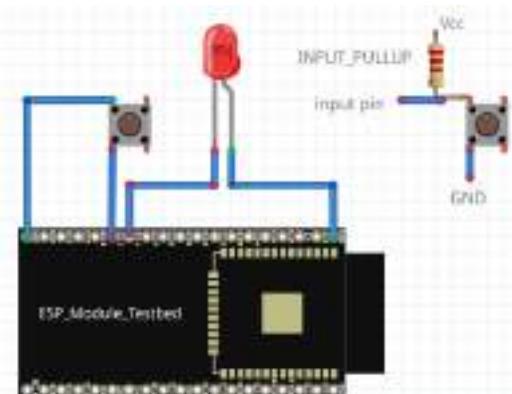
- GPIO: the number of a pin number where the interrupt signal generating device will be attached
- ISR – the name of a function of interrupt service routine,
- Mode - defines when interrupt signal is triggered. There are five basic mode values:
  - LOW - interrupt is triggered when the pin value is LOW
  - HIGH - interrupt is triggered when the pin value is HIGH,
  - CHANGE - interrupt is triggered when the pin value is changed,
  - RISING - interrupt is triggered when the pin value is changed from LOW to HIGH.
  - FALLING – interrupt is triggered when the pin value is changed from HIGH to LOW
- detachInterrupt (GPIO) :

It is an **option** when there is no longer to monitor GPIO

# Example 1

```
byte ledPin = 14;
byte interruptPin = 12;          /* pin that is attached to interrupt */
volatile byte state = LOW;      /* hold the state of LED when toggling */
void setup() {
    pinMode(ledPin, OUTPUT);
    /* set the interrupt pin as input pullup */
    pinMode(interruptPin, INPUT_PULLUP);
    /* attach interrupt to the pin function blink will be invoked when interrupt
       occurs interrupt occurs whenever the pin change value */
    attachInterrupt(digitalPinToInterrupt(interruptPin), blink, CHANGE);
}

void loop() {
}
/* interrupt function toggle the LED */
void blink()
{
    state = !state;
    digitalWrite(ledPin, state);
}
```



# Example 2: Button Input (1/4)



## Interrupt Handler for Button Input

```
int button_interrupt = 0;      // Interrupt 0 is on pin 2 !!
int toggle_on = false;        // Button click switches state

void setup() {
    Serial.begin(9600);
    attachInterrupt(button_interrupt, handle_click, RISING); // Register handler
}

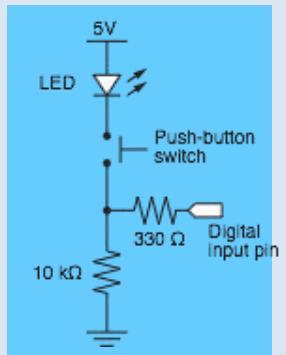
void loop() {
    if ( toggle_on ) {
        Serial.println("on");
    } else {
        Serial.println("off");
    }
}

void handle_click() {

    static unsigned long last_interrupt_time = 0;          // Zero only at start
    unsigned long interrupt_time = millis();                // Read the clock

    if ( interrupt_time - last_interrupt_time > 200 ) { // Ignore when < 200 msec
        toggle_on = !toggle_on;
    }

    last_interrupt_time = interrupt_time;
}
```



# Example 2: Button Input (2/4)



## Interrupt Handler for Button Input

```
int button_interrupt = 0;      // Interrupt 0 is on pin 2 !!
int toggle_on = false;        // Button click switches state

void setup() {
    Serial.begin(9600);
    attachInterrupt( button_interrupt, handle_click, RISING); // Register handler
}

void loop() {
    if ( toggle_on )
        Serial.println("on");
    else {
        Serial.println("off");
    }
}

void handle_click() {

    static unsigned long last_interrupt_time = 0;          // Zero only at start
    unsigned long interrupt_time = millis();               // Read the clock

    if ( interrupt_time - last_interrupt_time > 200 ) { // Ignore when < 200 msec
        toggle_on = !toggle_on;
    }

    last_interrupt_time = interrupt_time;
}
```

button\_interrupt is the ID or number of the interrupt. It must be 0 or 1

Interrupt handler must be registered when program starts

A RISING interrupt occurs when the pin changes from LOW to HIGH

The interrupt handler, handle\_click, is a user-written function that is called when an interrupt is detected

# Example 2: Button Input (3/4)



## Interrupt Handler for Button Input

```
int button_interrupt = 0;      // Interrupt 0 is on pin 2 !!
int toggle_on = false;        // Button click switches state

void setup() {
    Serial.begin(9600);
    attachInterrupt(button_interrupt, handle_click, RISING); // Register handler
}

void loop() {
    if (toggle_on) {
        Serial.println("on");
    } else {
        Serial.println("off");
    }
}

void handle_click() {

    static unsigned long last_interrupt_time = 0;          // Zero only at start
    unsigned long interrupt_time = millis();               // Read the clock

    if (interrupt_time - last_interrupt_time > 200) {   // Ignore when < 200 msec
        toggle_on = !toggle_on;
    }

    last_interrupt_time = interrupt_time;
}
```

toggle\_on is a global variable that remembers the “state”. It is either true or false (1 or 0).

The loop() function only checks the state of toggle\_on. The value of toggle\_on is set in the interrupt handler, handle\_click.

The value of toggle\_on is flipped only when a *true* interrupt even occurs. De-bouncing is described in the next slide.

# Example 2: Button Input (4/4)



## Interrupt Handler for Button Input

```
int button_interrupt = 0;      // Interrupt 0 is on pin 2 !!
int toggle_on = false;        // Button click switches state

void setup() {
  Serial.begin(9600);
  attachInterrupt( button_interrupt, handle_click, RISING); // Register handler
}

void loop() {
  if ( toggle_on ) {
    Serial.println("on");
  } else {
    Serial.println("off");
  }
}

void handle_click() {
  static unsigned long last_interrupt_time = 0;
  unsigned long interrupt_time = millis();

  if ( interrupt_time - last_interrupt_time > 200 ) { // Ignore when < 200 msec
    toggle_on = !toggle_on;
  }

  last_interrupt_time = interrupt_time;
}
```

Value of a *static* variable is always retained

Use *long*: the time value in milliseconds  
can become large

Clock time when current interrupt occurs

Ignore events that occur in less than 200 msec from  
each other. These are likely to be mechanical bounces.

Save current time as the new “last” time



## Example 3: 2 ISRs

```
void setup() {  
    pinMode(32, OUTPUT);  
    pinMode(33, OUTPUT);  
    pinMode(35, INPUT);  
    pinMode(34, INPUT);  
    attachInterrupt(35,pin35_ISR,CHANGE);  
    attachInterrupt(34,pin34_ISR,CHANGE);  
}  
void loop() {  
    digitalWrite(33,1);delay(100);  
    digitalWrite(33,0);delay(100);  
}  
void pin35_ISR()  
{  
    digitalWrite(32, HIGH);  
}  
void pin34_ISR()  
{  
    digitalWrite(32, LOW);  
}
```



## Example 4: IRAM\_ATTR (1)

```
struct Button {
    const uint8_t PIN;
    uint32_t numberKeyPresses;
    bool pressed;
};

Button button1 = {18, 0, false};

void IRAM_ATTR isr() {
    button1.numberKeyPresses += 1;
    button1.pressed = true;
}

void setup() {
    Serial.begin(115200);
    pinMode(button1.PIN, INPUT_PULLUP);
    attachInterrupt(button1.PIN, isr, FALLING);
}

void loop() {
    if (button1.pressed) {
        Serial.printf("Button 1 has been pressed %u times\n", button1.numberKeyPresses);
        button1.pressed = false;
    }

    //Detach Interrupt after 1 Minute
    static uint32_t lastMillis = 0;
    if (millis() - lastMillis > 60000)
    { lastMillis = millis();
        detachInterrupt(button1.PIN);
        Serial.println("Interrupt Detached!");
    }
}
```



# Example 5: IRAM\_ATTR (2)

## Timer Interrupts

```
volatile int interrupts;
int totalInterrupts;

hw_timer_t * timer = NULL;
portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;

void setup() {
    Serial.begin(115200);

    // Configure Prescaler to 80, as our timer runs @ 80Mhz
    // Giving an output of 80,000,000 / 80 = 1,000,000 ticks / second
    timer = timerBegin(0, 80, true);
    timerAttachInterrupt(timer, &onTime, true);
    // Fire Interrupt every 1m ticks, so 1s
    timerAlarmWrite(timer, 1000000, true);
    timerAlarmEnable(timer);
}

void IRAM_ATTR onTime() {
    portENTER_CRITICAL_ISR(&timerMux);
    interrupts++;
    portEXIT_CRITICAL_ISR(&timerMux);
}

void loop() {
    if (interrupts > 0) {
        portENTER_CRITICAL(&timerMux);
        interrupts--;
        portEXIT_CRITICAL(&timerMux);
        totalInterrupts++;
        Serial.print("totalInterrupts");
        Serial.println(totalInterrupts);
    }
}
```



# Example 7: Software Interrupt

## ESP32: Deep Sleep → TimeWakeup

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** TimerWakeUp | Arduino 1.8.13
- File Menu:** File Edit Search Tools Help
- Sketchbook:** Examples
- Code Area:** The code is for a Timer Wakeup example. It includes a loop that prints "DeepSleep" and then enters deep sleep via `esp_deep_sleep_start()`. A `Serial` port is used for debugging.
- Serial Monitor:** Shows the output: "DeepSleep" followed by a series of compressed data packets.
- Bottom Status Bar:** Shows the connection status as "Connected via WiFi" and the battery level at 100%.

**Context Menu (Mouse Right Clicked on the code area):**

- Build-in Examples: 01.Basics, 02.Digital, 03.Analog, 04.Communication, 05.Control, 06.Sensors, 07.Display, 08.Strings, 09.USB, 10.StarterKit\_BasicKit, 11.ArduinoISP
- Examples for any board: Adafruit Circuit Playground, Bridge, Ethernet, Pinata, LiquidCrystal, SD, Stepper, Tempio, RETIRED
- Examples for ESP32 Dev Module: ArduinoOTA, BluetoothSerial, DNSServer, EEPROM, ESP32, ESP32 Async UDP, ESP32 Async IoT Arduino, ESP32 BLE Arduino
- External WakeUp: DeepSleep, ESPNow, FreeRTOS, GND, HallSensor, I2S, ResetReason, IR, RTC, Touch
- TimerWakeup: TimerWakeup, TouchWakeup

**Text Box (Right Side):**

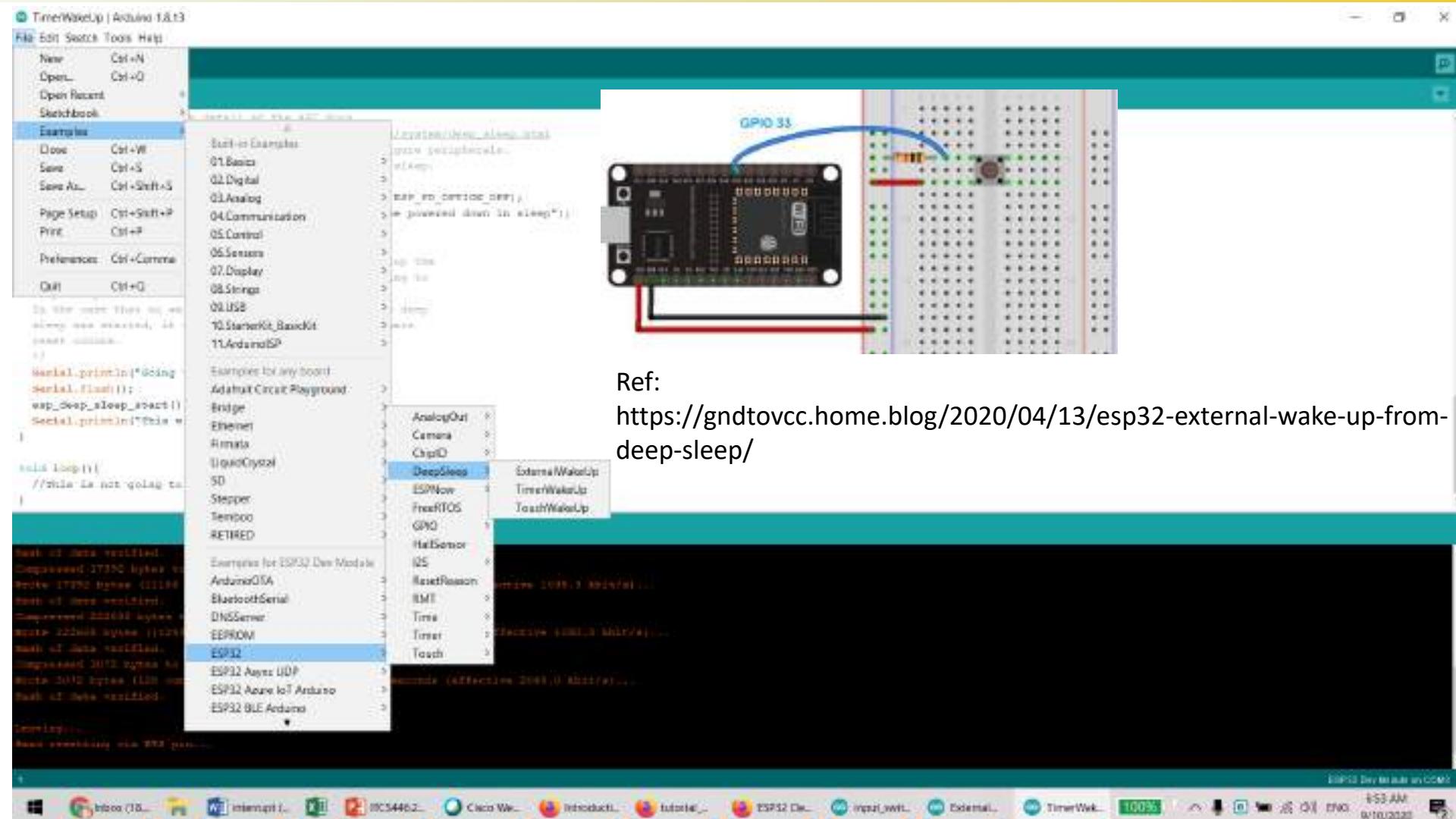
ESP32 provides 5 Modes for Deep Sleep:  
Active, Modem-sleep, Light-sleep, Deep-sleep, Hibernation

| Power mode                      | Active                    | Modem-sleep | Light-sleep | Deep-sleep                   | Hibernation |
|---------------------------------|---------------------------|-------------|-------------|------------------------------|-------------|
| Sleep pattern                   | Association sleep pattern |             |             | ULP sensor-monitored pattern | -           |
| CPU                             | ON                        | ON          | PAUSE       | OFF                          | OFF         |
| Wi-Fi/BT baseband and radio     | ON                        | OFF         | OFF         | OFF                          | OFF         |
| RTC, memory and RTC peripherals | ON                        | ON          | ON          | ON                           | OFF         |
| ULP co-processor                | ON                        | ON          | ON          | ON/OFF                       | OFF         |



# Example 8: Software Interrupt

## ESP32: Deep Sleep → External Wakeup



Ref:

<https://gndtovcc.home.blog/2020/04/13/esp32-external-wake-up-from-deep-sleep/>



[Mar11] Michael Margolis, **Arduino Cookbook**, O'Reilly, 2011.

[OK13] Robert Oshana, Mark Kraeling, **Software Engineering for Embedded Systems**, e-ISBN: 978-0-12-415941-9, Newnes, 2013.

[Whi11] Elecia White, **Making Embedded Systems**, O'Reilly, 2011.

[Kol17] Neil Kolban, Kolban's Book on ESP32, May 2017.

[SS1.4] Rui Santos and Sara Santos, Learn ESP32 with Arduino IDE v1.4.

## Arduino Tutorial

- ✓ <http://www.ladyada.net/learn/arduino/lesson5.html>

## Using interrupts

- ✓ <http://www.uchobby.com/index.php/2007/11/24/arduino-interrupts/>
- ✓ <https://lastminuteengineers.com/handling-esp32-gpio-interrupts-tutorial/>
- ✓ <http://www.arduino.cc/en/Reference/AttachInterrupt>
- ✓ <https://randomnerdtutorials.com/esp32-deep-sleep-arduino-ide-wake-up-sources/>



# **Finite State Machine**

## **FSM**

Asst. Prof. Dr. Thitinan Tantidham

- Logic control is an essential part to develop an intelligence device
- Logic control can be developed by
  - Truth table
    - You only need to know the corresponding input/output relation
    - As there is **no memory**, only simple operations can be achieved
  - Finite State Machine (FSM) or State Machine
    - Decision is based on what it has done i.e. the system has memory (history). Therefore the performance can be more complex.
    - A sequential system contains state stored in memory elements internal to the system.
    - Its behavior depends both on the set of inputs supplied and on the contents of the internal memory, or state of the system.
    - Thus, the sequential system cannot be described with a truth table.

# ESP32: Arduino IDE Programming



- Arduino ESP32 core is built over FreeRTOS. If you look into the `cores/esp32` folder, you will see a file called “`main.cpp`”.
  - Here you can see our Arduino program is just a task of FreeRTOS.
  - When the task is invoked it will invoke “`setup()`” function and then invoke the “`loop()`” in an infinite loop.
- Suppose that we implement an application that have 2 jobs:
  - + job A to scan the input from keyboard.
  - + job B to send a message and wait for response.
  - If job B takes long time to finish, it will block job A.
  - So job A may miss some keyboard pressing event. It is not a good design

```
void loop(){
    jobA_scan_the_keyboard();
    jobB_send_message();
    while(1){
        if( jobB_get_response() ){
            break;
        }
    }
}
```

<http://www.iotsharing.com/2017/05/how-to-apply-finite-state-machine-to-arduino-esp32-avoid-blocking.html>

# Finite State Machines (FSM)



- The purpose of this method is to divide a machine into **a collection of states**.
- At a specific time, **only one state is active** and in each state the machine just executes a specific action.
- After finishing that action, it may transit to another state to execute another action.
- For example:
  - we have a lighting system.
  - It can have 3 states: ON, OFF, IDLE.
  - Its normal state is IDLE and waiting for trigger to change state ON or OFF and then back to IDLE.

<https://arduining.com/2015/09/18/traffic-light-states-machine-with-arduino/>

<http://www.iotsharing.com/2017/05/how-to-apply-finite-state-machine-to-arduino-esp32-avoid-blocking.html>

## Switch/Case

- FSM can be implemented by **switch/case** structure.
- Here jobB is divided into 3 states: **SEND\_MESSAGE**, **WAIT\_FOR\_RESPONSE** and **IDLE**.
- jobB executes a specific action according to its state in every cycle and do not block jobA anymore.
- Each job has a chance to execute itself in every cycle, look like multitasking.

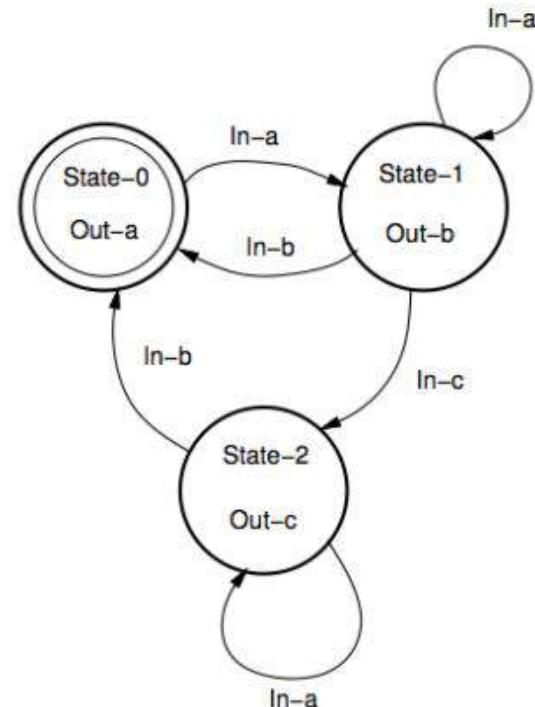
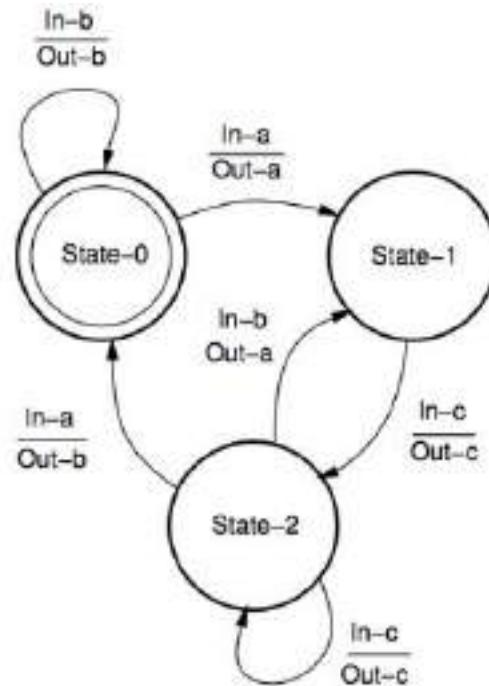
```
void loop(){
    jobA_scan_the_keyboard();
    switch(jobB_state)
    { case SEND_MESSAGE:
        jobB_send_message();
        jobB_state = WAIT_FOR_RESPONSE;
        break;
    case WAIT_FOR_RESPONSE:
        if(jobB_get_response()){
            jobB_state = IDLE;
            break;
        }
    case IDLE:
        if(need_send_message){
            jobB_state = SEND_MESSAGE;
            break;
        }
    case default:
        break;
    }
}
```

<http://www.iotsharing.com/2017/05/how-to-apply-finite-state-machine-to-arduino-esp32-avoid-blocking.html>

## How do we control a plant?

- Automation of the discrete operations (on-off) is largely a matter of a series of carefully timed on-off steps.
- Discrete Signals
  - Control parameters: true or false
  - Actuators: on or off
- Interlocks:
  - Output = function(input)
  - Boolean algebra
- Sequence Networks (Diagrams)
  - Dynamic systems
    - Output =  $f(\text{input}, \text{state})$
    - Next\_state =  $g(\text{input}, \text{state})$
  - FSM, Petri Nets, Grafcet/SFC (Sequential Function Chart), Grafchart

## Diagrams (1/2)



Mealy:

- Output events (actions) are associated with input events

Moore:

- State transitions in response to input events
- Output events (actions) are associated with states.

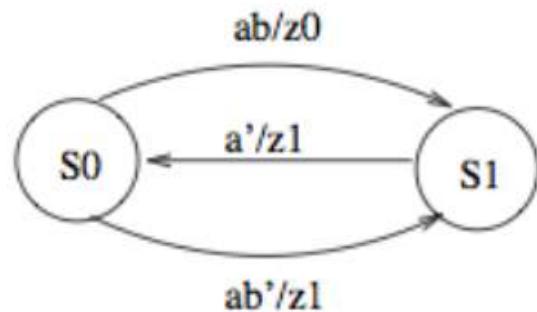
Ref:

<http://www.archive.control.lth.se/media/Education/EngineeringProgram/FRTN20/2016/LectureSlides-02-2016.pdf>

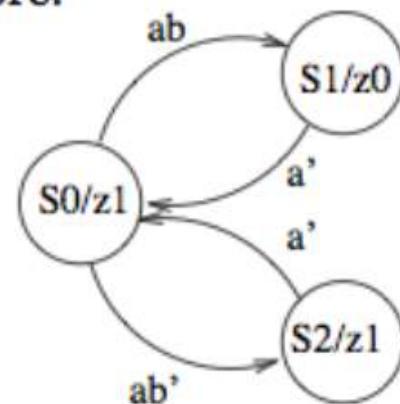


## Diagrams (2/2)

Mealy:



Moore:



Mealy:

- Output events (actions) are associated with input events

Moore:

- State transitions in response to input events
- Output events (actions) are associated with states.

Ref:

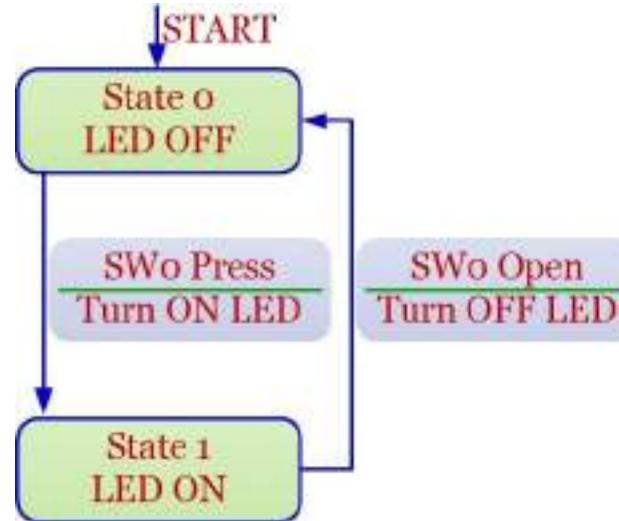
<http://www.archive.control.lth.se/media/Education/EngineeringProgram/FRTN20/2016/LectureSlides-02-2016.pdf>

# FSM Example 0



## Switch Off/On LED

```
#define LED0 15
#define SW0 32
byte State;
bool StateSW0;
void setup() {
    pinMode(LED0,OUTPUT);
    digitalWrite(LED0,LOW);
    pinMode(SW0,INPUT_PULLUP);
    Serial.begin(115200);
    Serial.println("...START...");
    State = 0;
}
void loop() {
    StateSW0 = digitalRead(SW0);
    switch (State)
    { case 0: //LED OFF
        if(StateSW0 == LOW)
        {   digitalWrite(LED0,HIGH);
            State = 1;
        }
        break;
    case 1: //LED ON
        if(StateSW0 == HIGH)
        {
            digitalWrite(LED0,LOW);
            State = 0;
        }
        break;
    }
```



Ref:

<https://phatiphatt.wordpress.com/esp32-finite-state-machine/>

<https://arduinoplusplus.wordpress.com/2019/07/06/finite-state-machine-programming-basics-part-1/>



# Finite State Machines

Introduction Video

# Finite State Machine



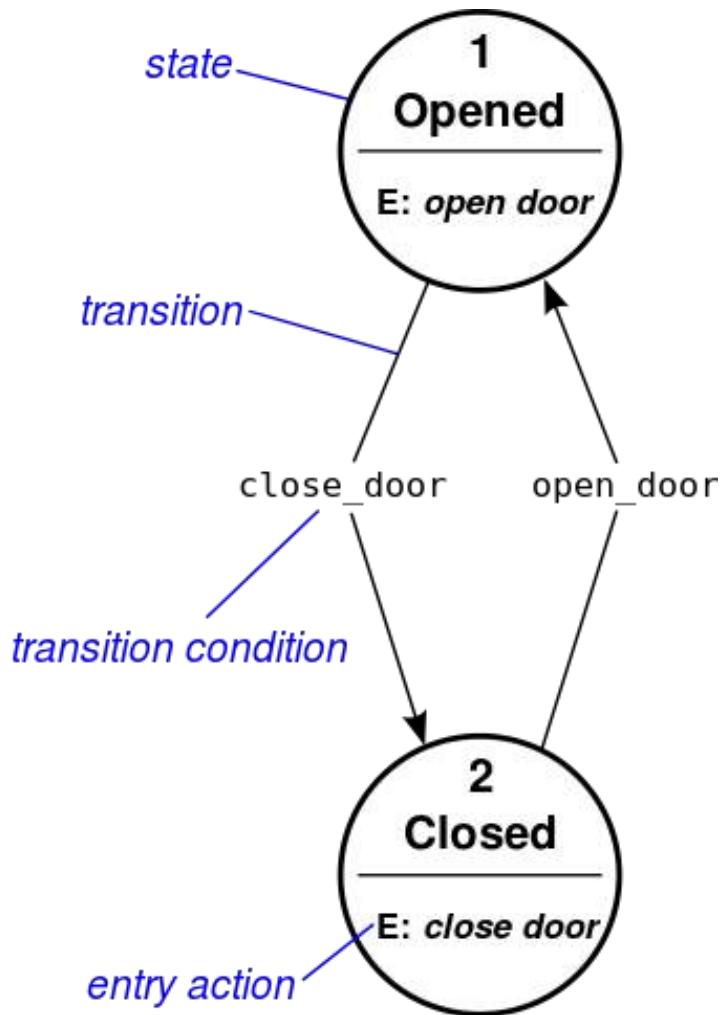


## Open/Close Door

Example of a door

- State
- Transition
- Transition condition
- Entry action

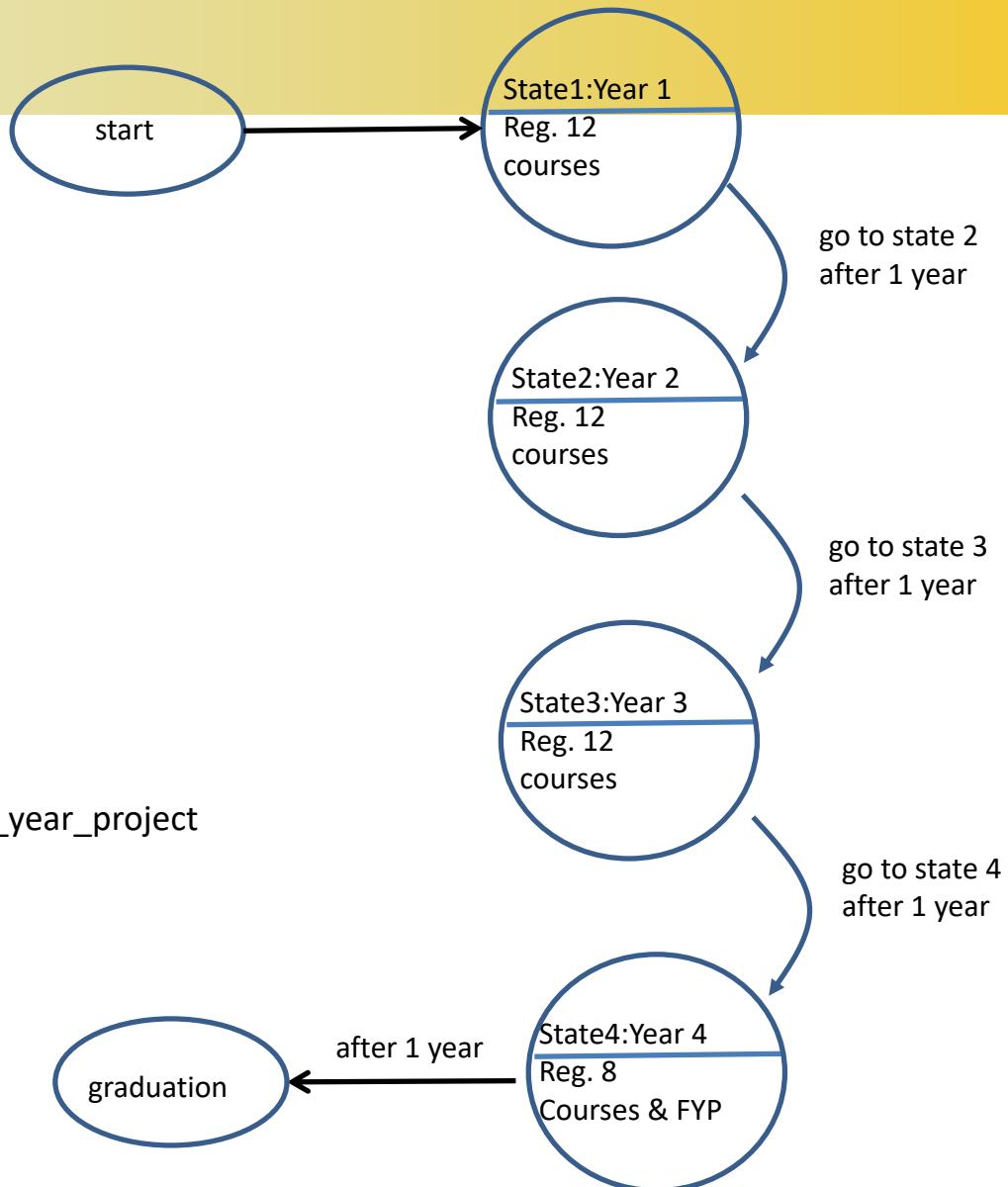
[http://en.wikipedia.org/wiki/State\\_diagram](http://en.wikipedia.org/wiki/State_diagram)



## Simple Study Life

State of a student at CUHK

- Start: go to state 1
- State 1=Year 1:
  - entry action: register 12 courses
  - Transition: go to state 2 after 1 year
- State 2=Year 2:
  - entry action: register 12 courses
  - Transition: go to state 3 after 1 year
- State 3=Year 3:
  - entry action: register 12 courses
  - Transition: go to state 4 after 1 year
- State 4=Year 4:
  - entry action: register 8 courses and FYP Final\_year\_project
  - Transition: go to stop after 1 year
- Stop: Graduation





## Life with Study Hard Condition

State of a student at CUHK

- Start: go to state1
- State 1=Year 1:
  - entry action: register 12 courses
  - Transition: if (study hard) promote to state2 (year2) else go back to state 1
- State 2=Year 2:
  - entry action: register 12 courses
  - Transition: if (study hard) promote to state3 (year3) else go back to state 2
- State 3=Year 3:
  - entry action: register 12 courses
  - Transition: if (study hard) promote to state4 (year4) else go back to state 3
- State 4=Year 4:
  - entry action: register 12 courses
  - Transition: if (study hard) promote to stop(graduation) else back go to state 4
- *Stop: Graduation*



We will have **three** examples here:

a. Simple finite state machine (no sensor).

e.g.: The dancing robot

b. An finite state machine uses 2 sensors.

e.g.: The robot that follows the magnetic strip

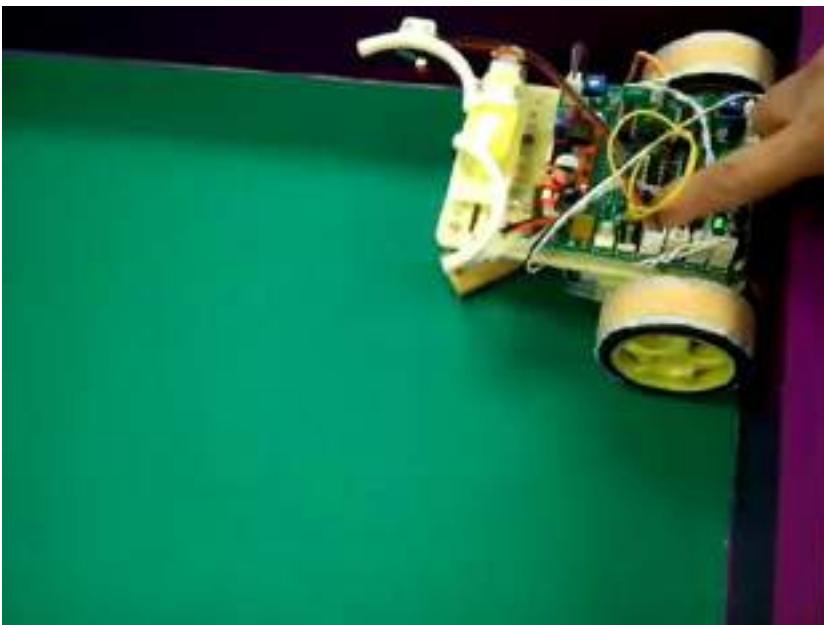
c. An finite state machine uses 3 sensors.

e.g.: The robot that follows the magnetic strip, and stops when it detects a magnet in front of the robot.



## No transition condition

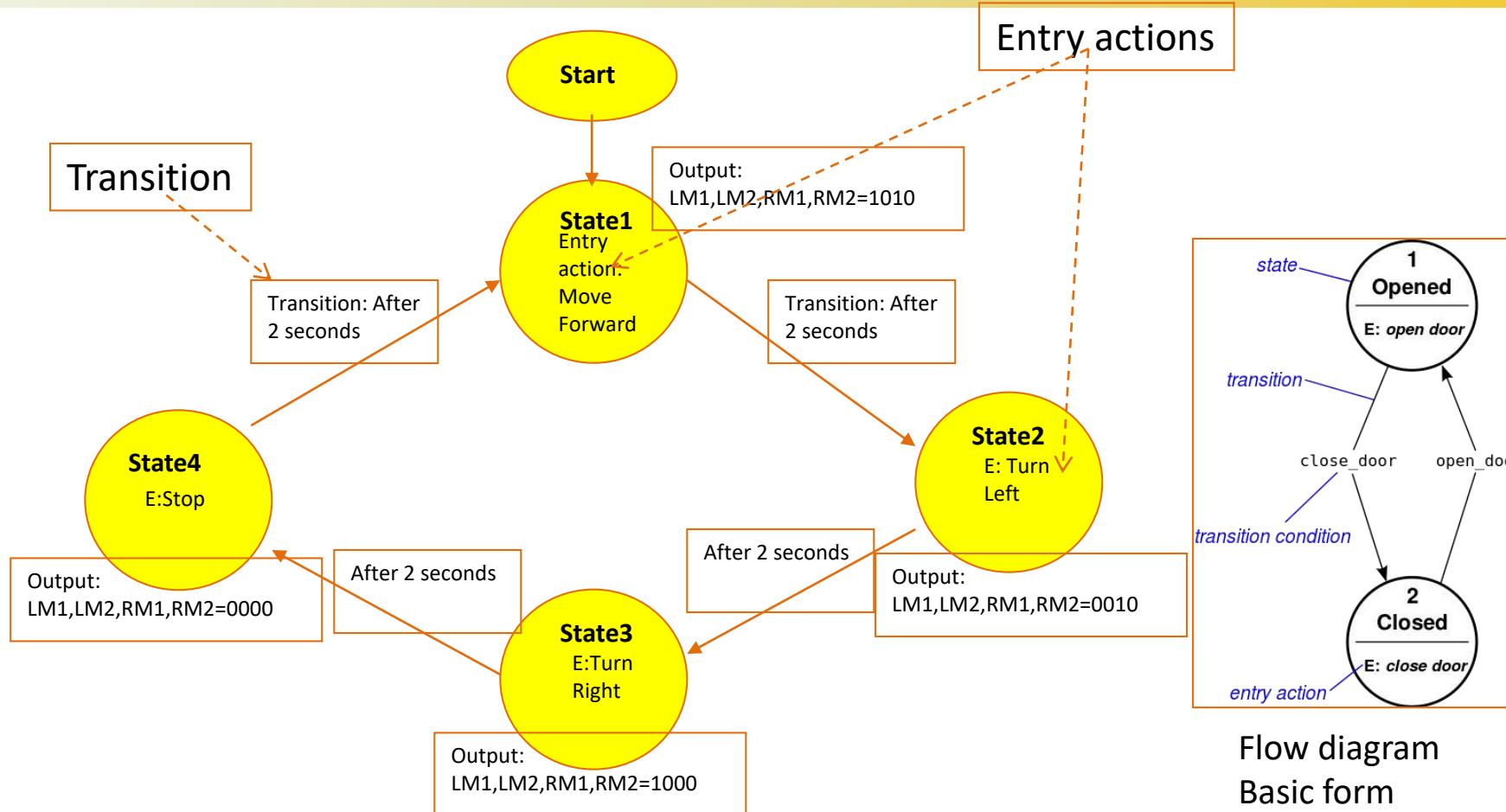
- The robot that dances with a pattern
  - Forward 2 seconds, turn left 2 seconds and turn right 2 seconds, stop and repeat the pattern again
- Video demo: <http://youtu.be/iyakbVyoafI>
- Using timing as a condition to change to another state



# FSM Example 3a: Simple FSM (2/3)



## No sensor input (no transition condition)



# FSM Example 3a: Simple FSM (3/3)



## Implementation

Part of the sample source code is shown below:

```
switch(state)
{
    case STATE1:
        LM1=1;LM2=0;RM1=1;RM2=0;SPEED=200; //entry action
        DELAY_TIME=2000; // transition :delay 2 seconds
        motors(LM1,LM2,RM1,RM2,SPEED,SPEED,DELAY_TIME);
        state=STATE2; // next state will be state2
        break; //end of the current state
    case STATE2:
        LM1=0;LM2=0;RM1=1;RM2=0;SPEED=200;DELAY_TIME=2000; // delay 2 seconds
        motors(LM1,LM2,RM1,RM2,SPEED,SPEED,DELAY_TIME);
        state=STATE3; //next state will be state3
        break; //end of the current state
    .
    .
    .
}
```

Set motor to run forward with speed=200

Use of DELAY:

DELAY\_TIME=2000

motors(LM1,LM2,RM1,RM2,SPEED,  
SPEED,DELAY\_TIME);

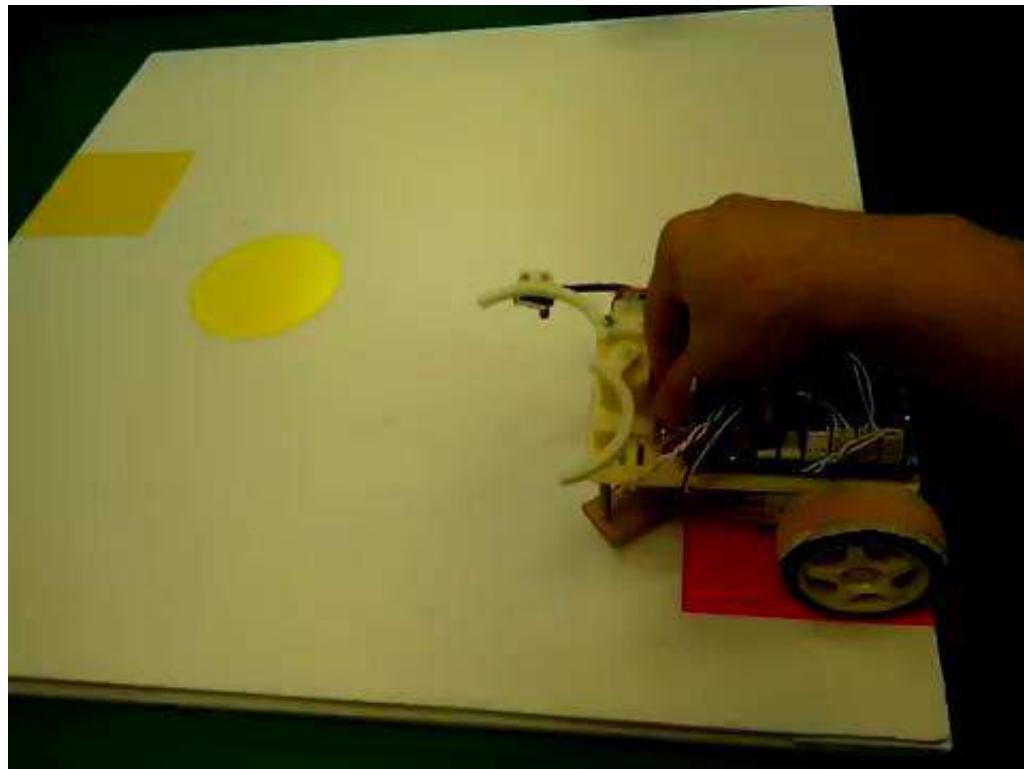
You may need to write  
the function motors( ) for your project

Exercise: explain the meaning of state 2

# FSM Example 3b: FSM with 2 Sensors (1/2)



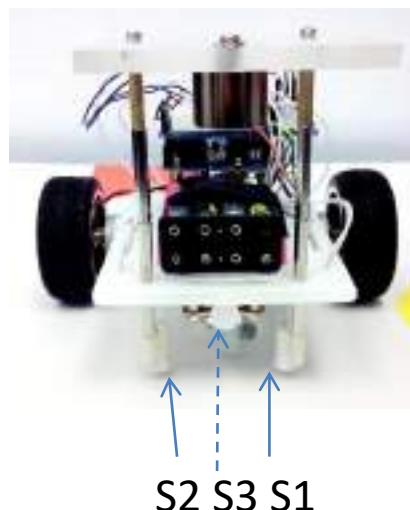
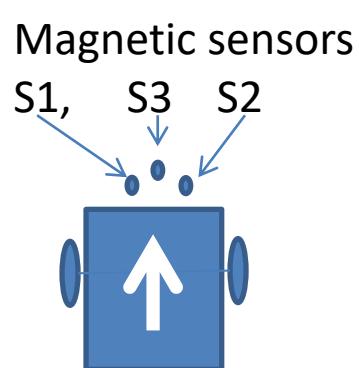
- The robot can follow the magnetic strip
- Video Demo: [http://youtu.be/NWHjWrq\\_VoY](http://youtu.be/NWHjWrq_VoY)



# FSM Example 3b: FSM with 2 Sensors (2/2)



- Sensors: S2, S1 are facing the ground to detect the magnetic strip
- S3 is facing the front, used to detect the target object
  - S3=1 if no object is detected
  - S3=0 if an object is detected

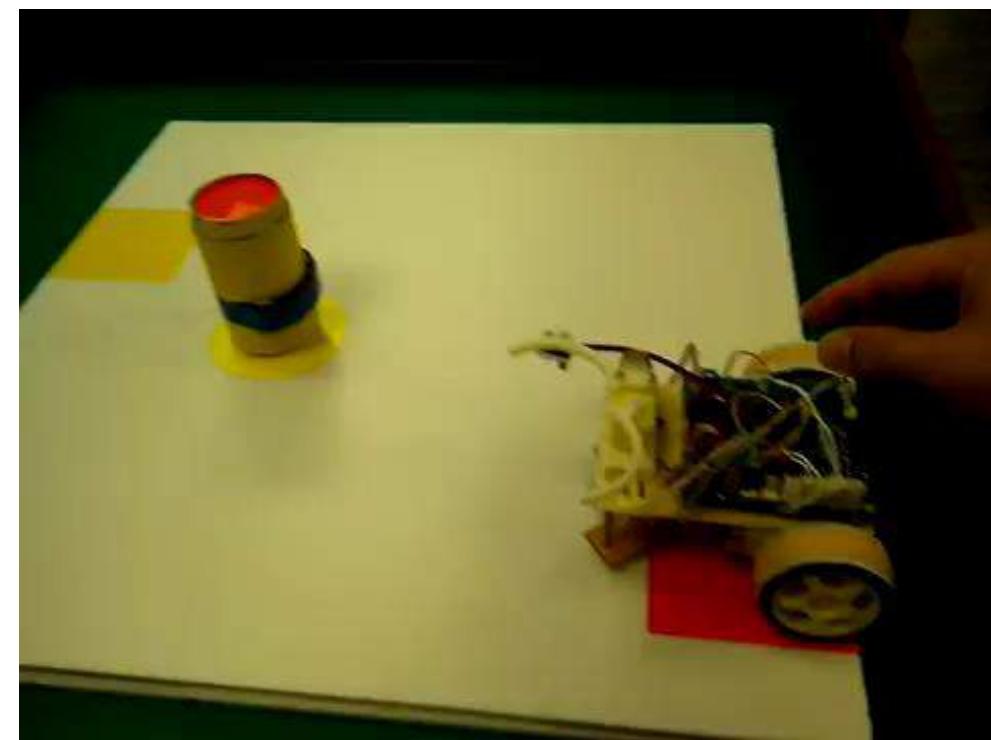
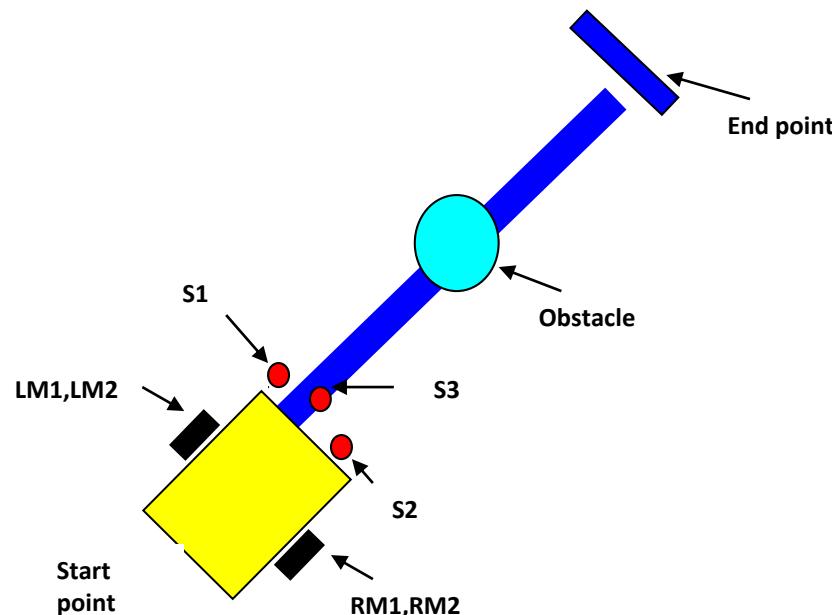


# FSM Example 3c: FSM with 3 Sensors



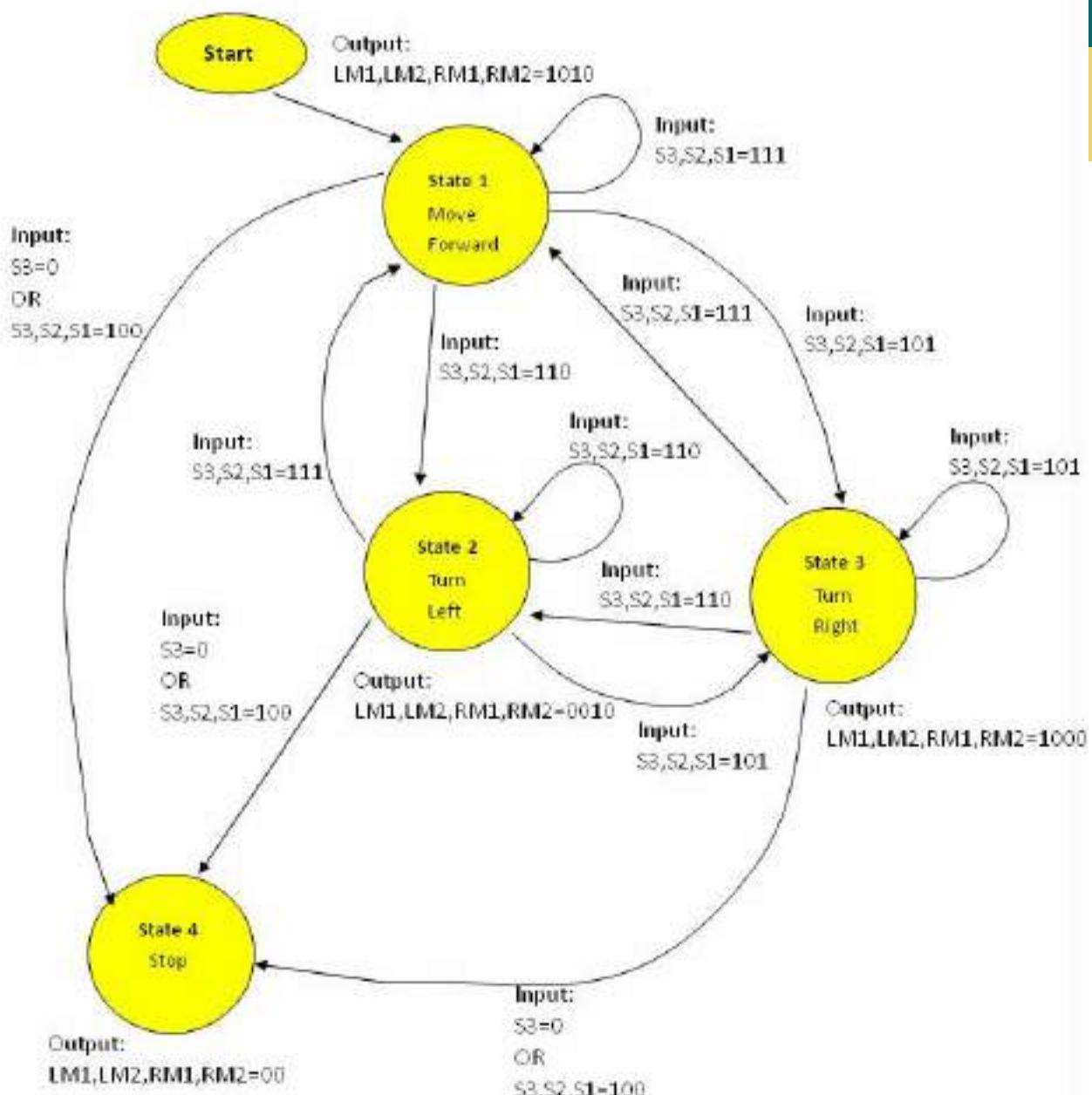
- Video Demo : <http://youtu.be/JEQkuax7IKE>

- The robot finds the CAN using the magnetic strip placed under the testing board and stops

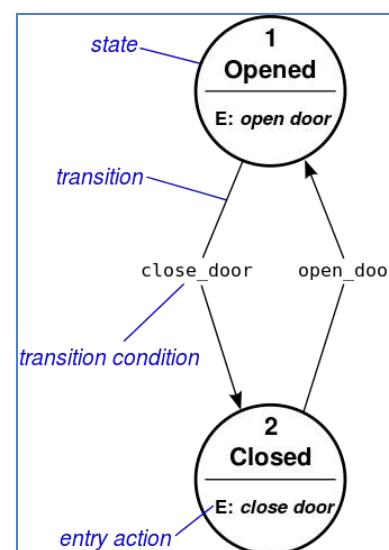




The state diagram of the Smart-car logic can be represented by the state diagram below.



Flow diagram  
Basic form





1/2

The sample source code (program\_segment3) is shown below:

```
switch(state)
{
    case STATE1: // forward for 1 second
        LM1=1;LM2=0;RM1=1;RM2=0;
        SPEED=200;DELAY_TIME=10;
        motors(LM1,LM2,RM1,RM2,SPEED,DELAY_TIME);
        //
        if ( S3()==1 && S2()==1 && S1()==0 ) state=STATE2;
        else if(S3()==1 && S2()==0 && S1()==1) state=STATE3;
        else if((S3==0) || (S3()==1 && S2()==0 && S1()==0)) state=STATE4;
        break;

    case STATE2: //robot turns left
        LM1=0;LM2=0;RM1=1;RM2=0;SPEED=200;DELAY_TIME=10;
        motors(LM1,LM2,RM1,RM2,SPEED,DELAY_TIME);
        //
        if ( S3()==1 && S2()==1 && S1()==1 ) state=STATE1; //back to state 1
        else if(S3()==1 && S2()==0 && S1()==1) state=STATE3;
        else if((S3==0) || (S3()==1 && S2()==0 && S1()==0)) state=STATE4;    break;
}
```

If S3=0, a CAN is detected, next state is state4

Move forward for 1 second

Robot deviated to the right, goto state 2

Robot deviated to the left goto state 3

2/2

```
case STATE3: //robot turns right
    // To be filled by students as an exercise
    .
    .
    .
    .

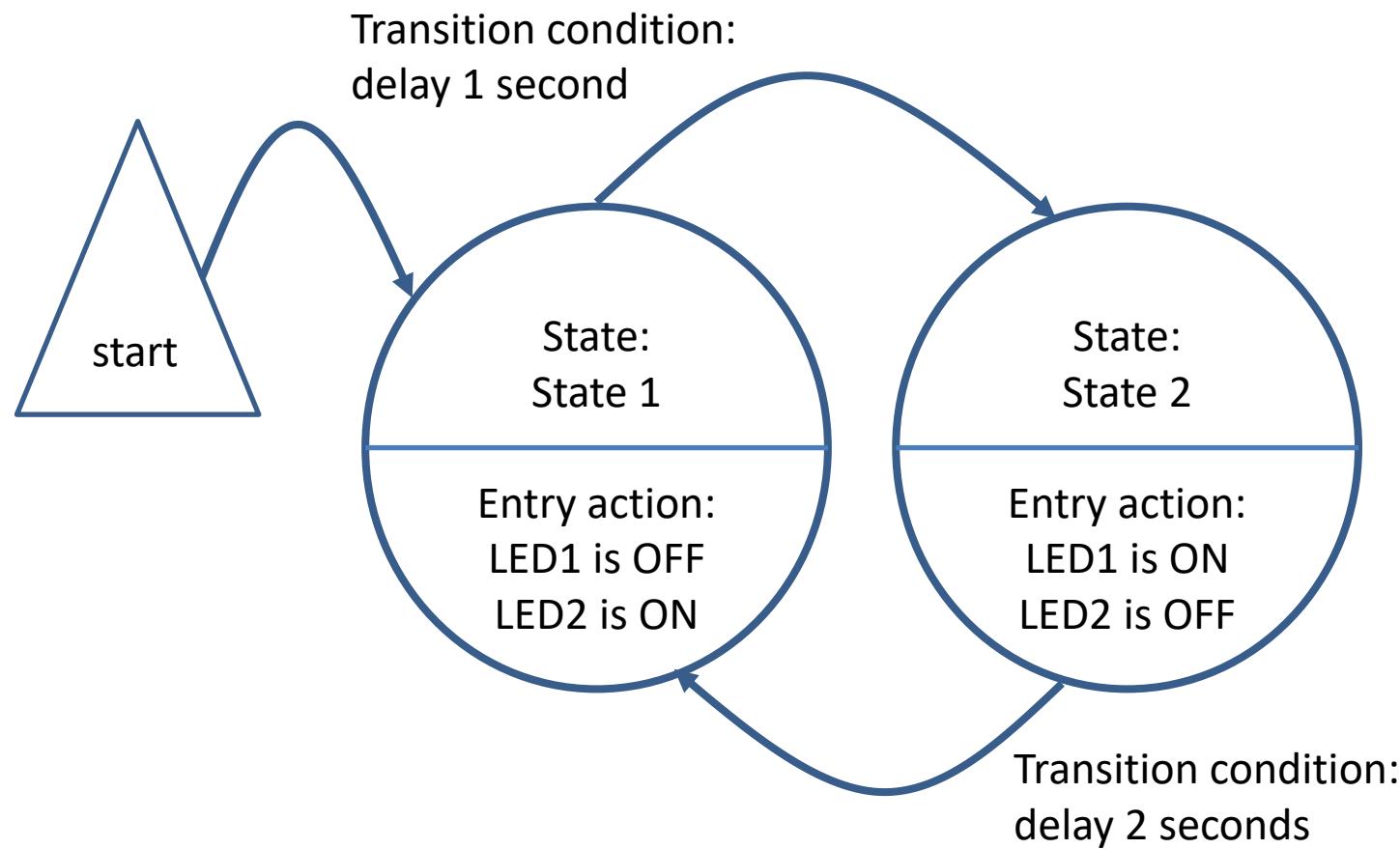
case STATE4: //stop
    // To be filled by students as an exercise
    .
    .

default: //none of above states
    state=STATE4;
    LM1=0;LM2=0;RM1=0;RM2=0;
    SPEED=200;DELAY_TIME=10;
    motors(LM1,LM2,RM1,RM2,SPEED,DELAY_TIME);
break;
}
```



1/2

- Using delay time as transition condition



# FSM Example 4: LEDs: FSM without Input/Sensor



2/2

```
#define STATE1 1
#define STATE2 2
#define STATE_END 100
unsigned char state=1;
//init. to state1

void setup() {
    pinMode (LED1, OUTPUT);
    pinMode (LED2, OUTPUT);
}

void loop() {
    switch(state) {
        case STATE1:
            digitalWrite(LED1, HIGH);
            digitalWrite(LED2, LOW);
            delay(1000);
            state=STATE2;
            break;
        case STATE2:
            digitalWrite(LED1, LOW);
            digitalWrite(LED2, HIGH);
            delay(2000);
            state=STATE1;
            break;
        case STATE_END:
            // turn off the LEDs, this state is not used here
            digitalWrite(LED1, HIGH);
            digitalWrite(LED2, HIGH);
            break;
        default:
            state=STATE_END;
            break;
    }
}
```

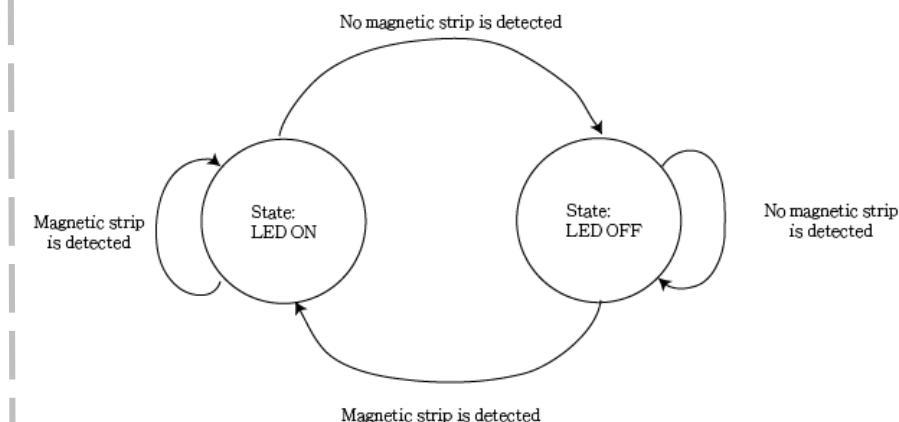
1/2

- When a magnetic strip is detected, the LED is ON
- When there is no magnetic strip, the LED is OFF

State Transition Table

| Input                         | Current State | Next State |
|-------------------------------|---------------|------------|
| Magnetic strip is detected    | ON            | ON         |
| Magnetic strip is detected    | OFF           | ON         |
| No magnetic strip is detected | ON            | OFF        |
| No magnetic strip is detected | OFF           | OFF        |

State Diagram



# FSM Example 4: LEDs: FSM with Input/Sensor



2/2

```
#define STATE1 1
#define STATE2 2
#define STATE_END 100
int magnetic = 7;
int ledPin_S1 = 5;
int ledPin_S2 = 6;
unsigned char state=1;
void setup() {
    pinMode (magnetic, INPUT);
    pinMode (ledPin_S1, OUTPUT);
    pinMode (ledPin_S2, OUTPUT);
}
void loop() {
    switch(state) {
        case STATE1:
            digitalWrite(ledPin_S1, HIGH);
            digitalWrite(ledPin_S2, LOW);
            if (digitalRead(magnetic) == LOW)
                state=STATE2; break;
        case STATE2:
            digitalWrite(ledPin_S1, LOW);
            digitalWrite(ledPin_S2, HIGH);
            if (digitalRead(magnetic) == HIGH)
                state=STATE1; break;
        case STATE_END:
            digitalWrite(ledPin_S1, HIGH);
            digitalWrite(ledPin_S2, HIGH);
            break;
        default:
            state=STATE_END;
            break;  }}
```



## Without FSM

```
void loop()
{
    /* if client was disconnected then try to reconnect again */

    if (!client.connected())
    {
        mqttconnect();
    }
    /* this function will listen for incoming subscribed topic-process-invoke receivedCallback */
}

client.loop();

/* we measure temperature every 3 secs we count until 3 secs reached to avoid blocking
program if using delay()*/
long now = millis();
if (now - lastMsg > 3000) {
    lastMsg = now;
    /* read DHT11/DHT22 sensor and convert to string */
    temperature = dht.readTemperature();
    if (!isnan(temperature))
    { snprintf (msg, 20, "%lf", temperature);
      /* publish the message */
      client.publish(TEMP_TOPIC, msg);
    }
}
```

<http://www.iotsharing.com/2017/05/how-to-apply-finite-state-machine-to-arduino-esp32-avoid-blocking.html>

# FSM Example 5: MQTT & DHT11



## With FSM

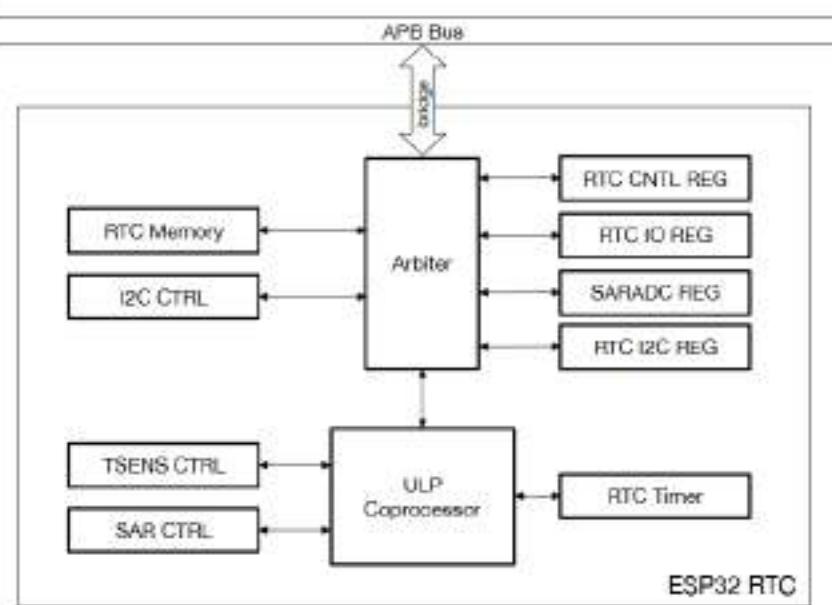
```
/* these are available states of DHT */
typedef enum {
    /* wait until 3 seconds */
    WAIT_FOR_TIMEOUT = 1,
    /* read DHT sensor for temperature */
    START_MEASURE,
    /* publish MQTT topic temperature */
    PUBLISH_MEASURE
} DHT_State;

/* create a variable to hold current state of DHT */
DHT_State state = WAIT_FOR_TIMEOUT;

void loop()
{
    /* if client was disconnected then try to reconnect again */
    if (!client.connected()) {
        mqttconnect();
    }
    /* this function will listen for incomming subscribed topic-process-invoke receivedCallback */
    client.loop();
    /* we measure temperature every 3 secs this code is to avoid blocking program and easy to read
     * maintain when using Finite State Machine */
    switch(state){
        case WAIT_FOR_TIMEOUT:
            long now = millis();
            if (now - lastMsg > 3000) {
                state = START_MEASURE;
            }
            break;
        case START_MEASURE:
            temperature = dht.readTemperature();
            if (!isnan(temperature)) {
                sprintf (msg, 20, "%lf", temperature);
                state = PUBLISH_MEASURE;
            }
            break;
        case PUBLISH_MEASURE:
            /* publish the message */
            client.publish(TEMP_TOPIC, msg);
            lastMsg = now;
            state = WAIT_FOR_TIMEOUT;
            break;
    }
}
```

<http://www.iotsharing.com/2017/05/how-to-apply-finite-state-machine-to-arduino-esp32-avoid-blocking.html>

## FSM with ULP Co-Processor



- The ULP co-processor is an ultra-low-power processor that remains powered on during the Deep-sleep mode of the main SoC.
- Hence, the developer can store in the RTC memory a program for the ULP co-processor to access peripheral devices, internal sensors and RTC registers during deep sleep.
- This is useful for designing applications where the CPU needs to be woken up by an external event, or timer, or a combination of these, while maintaining minimal power consumption.

- The ULP co-processor is a programmable FSM (Finite State Machine) that can work during deep sleep.
- Like general-purpose CPUs, ULP co-processor also has some instructions which can be useful for a relatively complex logic, and **also some special commands for RTC controllers/peripherals**.
- RTC GPIOs can be used to wake up the ESP32 from deep sleep when the ULP co-processor is running. (RTC\_GPIO0, RTC\_GPIOs 3-17)

References:

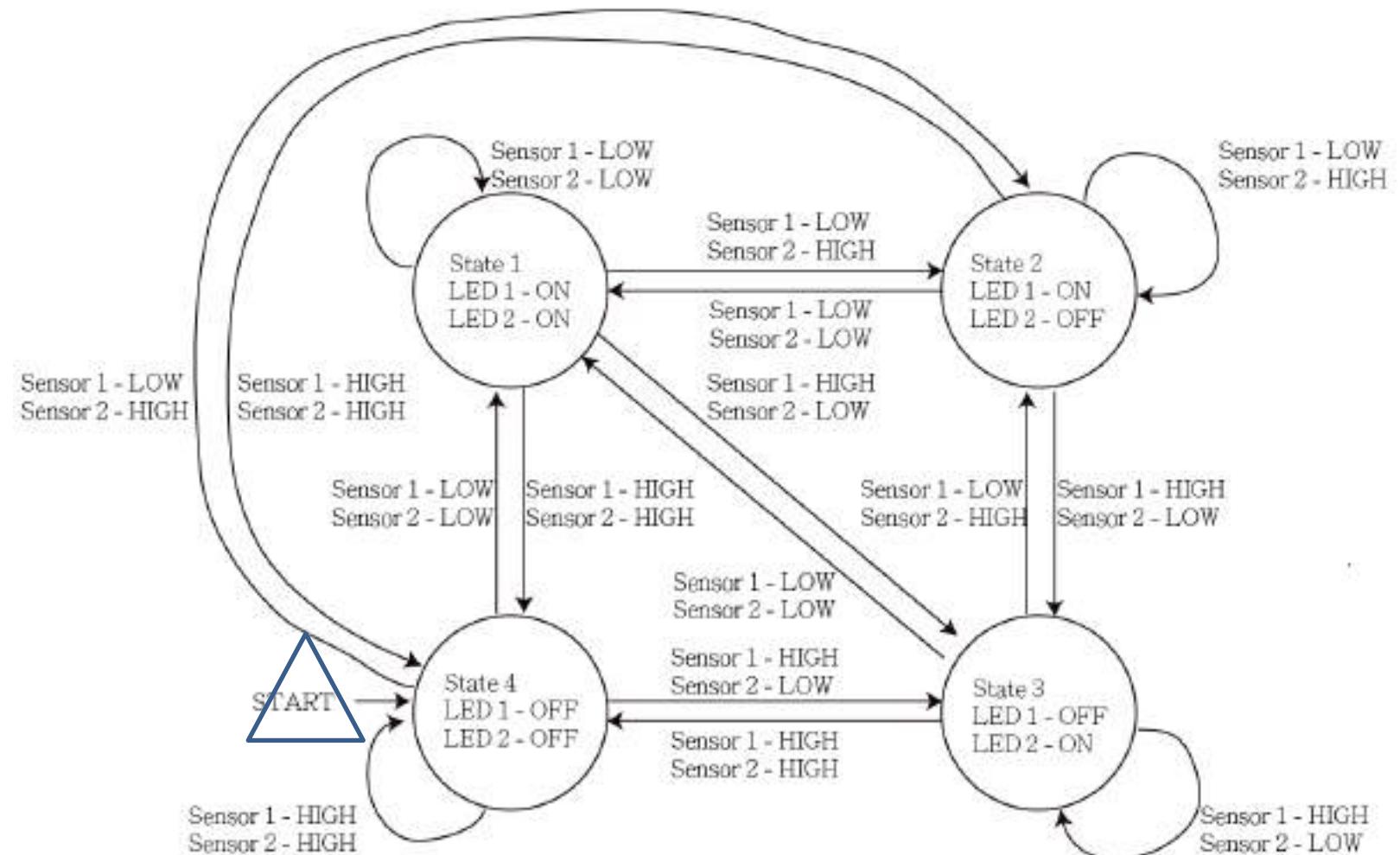
<https://circuits4you.com/2018/12/31/esp32-devkit-esp32-wroom-gpio-pinout/>

[http://www.ee.ic.ac.uk/pcheung/teaching/DE1\\_EE/Labs/esp32\\_technical\\_reference\\_manual\\_en.pdf](http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/Labs/esp32_technical_reference_manual_en.pdf)



- **Write state transition table and program** that controls **two LEDs** through two input signals, the specification is shown in flow diagram in the next slide
  - Use inputs
    - pinA for sensor1, and pinB for sensor2.
    - The values of input signals (sensor 1 and sensor 2) can be either GND (LOW) or 5V (HIGH)
  - Use outputs
    - pinC for LED1 and pinD for LED2
  - Print out the status of two input sensors on console.
- Submit a zip file in Grxx.FSMExercise-StudentID-StudentID on mycourses with
  - pdf report contains FSM diagram, fritzing schematic and source code
  - video file

## FSM Diagram





## Hints

```
//hint
#define STATE1 1
#define STATE2 2
#define STATE3 3
#define STATE4 4
#define STATE_END 100

int sensor1 = 0;
int sensor2 = 1;
int led1 = 5;
int led2 = 6;
unsigned char state=4;

void setup() {
    pinMode (sensor1, INPUT_PULLUP);
    pinMode (sensor2, INPUT_PULLUP);
    pinMode (led1, OUTPUT);
    pinMode (led2, OUTPUT);
}

void loop() {
    switch(state) {
        case STATE1:
            digitalWrite(led1, HIGH);
            digitalWrite(led2, HIGH);
            if ((digitalRead(sensor1) == LOW) &&
                (digitalRead(sensor2) == HIGH)) state=STATE2;
            else if ((digitalRead(sensor1) == HIGH) &&
                      (digitalRead(sensor2) == LOW)) state=STATE3;
            else if ((digitalRead(sensor1) == HIGH) &&
                      (digitalRead(sensor2) == HIGH)) state=STATE4;
            break;
    }
}
```

# References



## Arduino:

- <http://playground.arduino.cc/Code/FiniteStateMachine>
- Slides from Prof. Kin Hong Wong

## FSM:

[http://users.ece.utexas.edu/~valvano/Volume1/E-Book/C10\\_FiniteStateMachines.htm?fbclid=IwAR1UPmrNbXNd8zUhABG7FMLFWm8CdNr2-lukWrfW-MkszEkA5gGNMh7Y1s](http://users.ece.utexas.edu/~valvano/Volume1/E-Book/C10_FiniteStateMachines.htm?fbclid=IwAR1UPmrNbXNd8zUhABG7FMLFWm8CdNr2-lukWrfW-MkszEkA5gGNMh7Y1s)

## FSM Libraries:

Automaton: <https://github.com/tinkerspy/Automaton>

## Function-FSM:

<https://jrvеаlе.com/2019/06/03/library-function-fsm-c/>

github.com/JRVeale/function-fsm\

<https://github.com/jeroendoggen/Arduino-finite-state-machine-library>

## Cases:

<https://medium.com/coinmonks/multitasking-on-the-arduino-with-a-finite-state-machine-and-why-youll-need-it-for-sigfox-d52dafc55d8e>

## Books:

[Mar12] Michael Margolis, **Arduino Cookbook**, O'Reilly, 2012.

[Box13] John Boxall, Arduino Workshop: A Hands-On Introduction with 65 Projects,  
No Starch Press, San Francisco, 2013.

[Mon14] Programming Arduino: Next Steps: Going Further with Sketches, McGrawHill 2014.



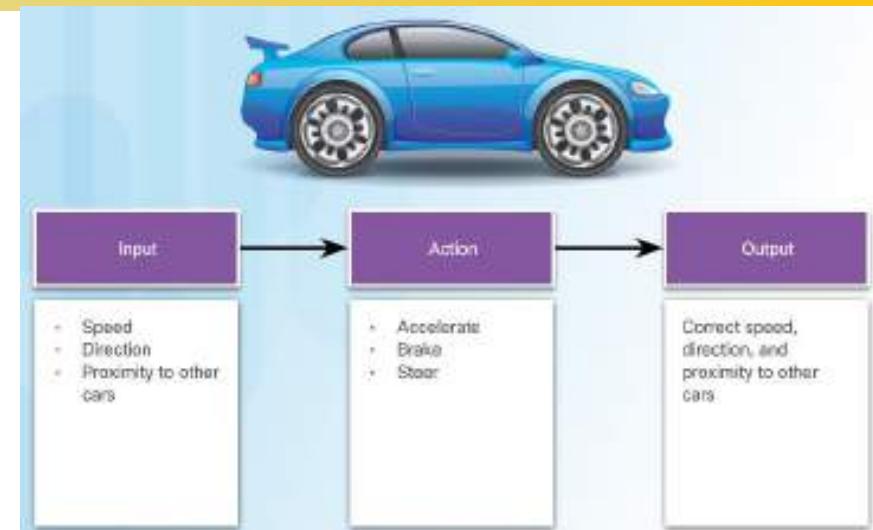
# PID

## Proportional Integral Derivative

# Review: Processes in Controlled Systems



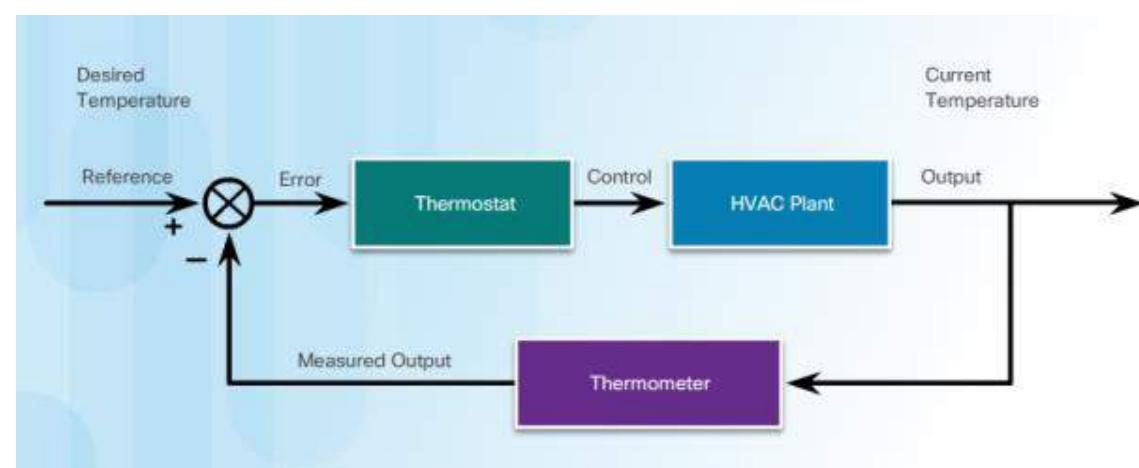
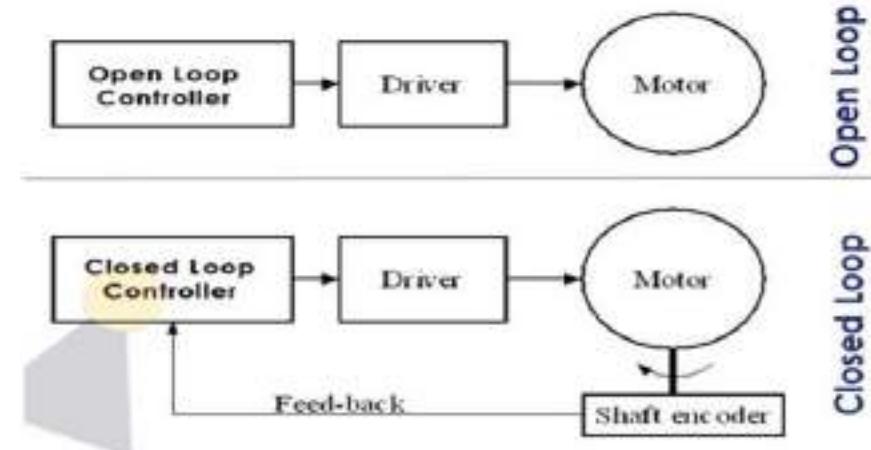
- Processes
  - ✓ A process is **a series of steps or actions** taken to achieve a desired result by the consumer of the process.
- Feedback
  - ✓ Feedback is when **the output of a process affects the input**.
  - ✓ Feedback is often referred to as a feedback loop.
  - ✓ Feedback loops can be positive or negative.
- Control Systems
  - ✓ Includes a controller that **uses inputs and outputs to manage and regulate the behavior of the system in an attempt to achieve a desired state**.
  - ✓ The controlled portion of the system is often called the **plant**.
  - ✓ Choosing the adjustments to apply to a plant to achieve a desired output is called **control theory**.
  - ✓ Control theory is applied to many systems, including driving a car.



# Review: Open-Loop/Closed-Loop Control Systems



- Open-Loop Control Systems
  - ✓ Open-loop control systems **do not use feedback**.
  - ✓ The plant performs a predetermined action without any verification of the desired results.
  - ✓ Open-loop control systems are often used for simple processes.
- Closed-Loop Control Systems
  - ✓ A closed-loop control system uses feedback to determine whether the collected output is the desired output.
  - ✓ The result is then fed back into a controller to adjust the plant for the next iteration of output, and the process repeats.



# Review: Closed-Loop Controllers



- Types of Closed-Loop Controllers
  - ✓ Proportional controllers (P): based on the difference between the measured output and the desired output.
    - Most systems have many interdependent pieces contributing to and affecting the output.
  - ✓ Integral controllers (PI): use historical data to measure how long the system has deviated from the desired output.
  - ✓ Proportional, Integral and Derivative controllers (PID): include data about how quickly the system is approaching the desired output.
    - PID controller is an efficient way to implement feedback control.
    - The Arduino and Raspberry Pi devices can be used to implement PID controllers.



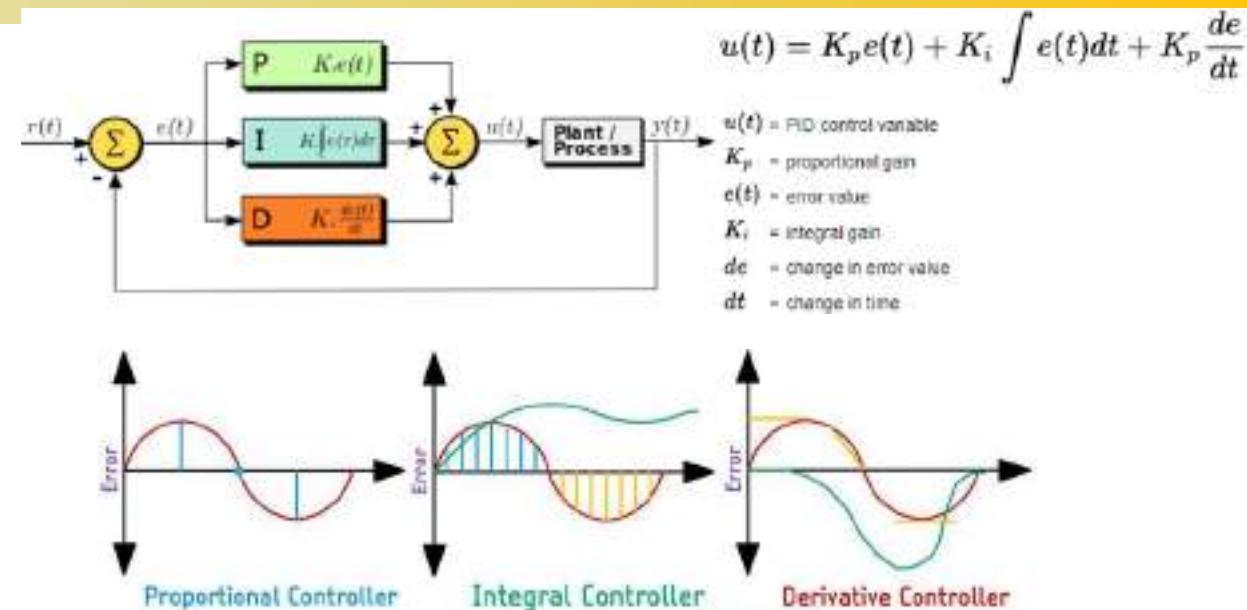
# PID (Proportional Integral Derivative)



- The **proportional** term:
  - It generates a response that is proportional to the error.
  - For example: the error is the angle of inclination of the robot.
- The **integral** term:
  - It generates a response based on the accumulated error.
  - This is essentially the sum of all the errors multiplied by the sampling period.
  - This is a response based on the behavior of the system in past.
- The **derivative** term:
  - It is proportional to the derivative of the error.
  - This is the difference between the current error and the previous error divided by the sampling period.
  - This acts as a predictive term that responds to how the robot might behave in the next sampling loop.
- Multiplying each of these terms by their corresponding constants (i.e, K<sub>p</sub>, K<sub>i</sub> and K<sub>d</sub>) and summing the result, we generate the output which is then sent as command to drive the motor.

For example:

```
motorPower = Kp*(error) + Ki*(errorSum)*sampleTime - Kd*(currentAngle-prevAngle)/sampleTime;
```



Ref:

<https://circuitdigest.com/microcontroller-projects/arduino-based-encoder-motor-using-pid-controller>

# PID Controller





## ESP32

### Project Examples

- ✓ Rancilio Silvia Coffee Machine  
<https://www.instructables.com/PID-Controlled-Thermostat-Using-ESP32-Applied-to-a/>
- ✓ PID based Encoder Motor Controller:  
<https://www.youtube.com/watch?v=Ds2E0NC4JhA>  
<https://josef-adamcik.cz/electronics/self-balancing-stories-ep2-firmware.html>
- ✓ Arduino Self Balancing Robot  
<https://www.instructables.com/Arduno-Self-Balancing-Robot-1/#step6>
- ✓ <https://githubmemory.com/repo/simplefoc/Arduino-FOC-balancer>

### Libraries

- ✓ QuickPID  
<https://www.arduino.cc/reference/en/libraries/quickpid/>  
<https://github.com/Dlloyddev/QuickPID/wiki/AutoTune>
- ✓ ArcPID (PID.h)  
<https://github.com/ettoreleandrotognoli/ArcPID>  
<https://platformio.org/lib/show/11921/ArcPID>
- ✓ SimpleFOClibrary (PIDController.h)  
[https://docs.simplefoc.com/pi\\_controller](https://docs.simplefoc.com/pi_controller)



- [VB17] Ovidiu Vermesan and Joel Bacquet, “**Cognitive Hyperconnected Digital Transformation: Internet of Things Intelligence Evolution**”, River Publishers, 2017.
- [BHC+18] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, “**The Industrial Internet of Things (IIoT): An Analysis Framework**”, Computers in Industry 101, October 2018.
- [BS19] Rajkumar Buyya and Satish Narayana Srirama, “**Fog and Edge Computing: Principles and Paradigms**”, Wiley Series on Parallel and Distributed Computing, 2019.
- [RS19] Ammar Rayes and Samer Salam, “**Internet of Things from Hype to Reality: The Road to Digitization**”, 2<sup>nd</sup> Edition, Springer, 2019.
- [LEA20] Perry Lea, “**IoT and Edge Computing for Architects: Implementing edge and IoT systems from sensors to clouds with communication systems, analytics, and security**”, 2<sup>nd</sup> Edition, Packt Publishing, 2020.
- [GG20] Patrick Grossetete and Stephen Goodman, “**Wireless Technologies and Use Cases in Industrial IoT**”, BKKIOT-1775, Cisco Live, January 2020.
- Lecture Course: Wireless and Mobile Networking <https://www.cse.wustl.edu/~jain/cse574-20/index.html>



# ITCS447

## Lecture 5

### Sensors-Actuators

Asst. Prof. Dr. Thitinan Tantidham



มหาวิทยาลัยมหิดล  
Mahidol University

ห้ามมิให้นักศึกษาทำข้า ดัดแปลง หรือใช้ประโยชน์จากการอันมีลิขสิทธิ์ปราการอยู่ในระบบการศึกษาอิเล็กทรอนิกส์ (E-Learning) ของมหาวิทยาลัย ไม่ว่าจะทั้งหมดหรือบางส่วน โดยไม่ได้รับอนุญาตจากมหาวิทยาลัย นอกจากนี้จากการศึกษาส่วนบุคคล ทั้งนี้ การทำข้า ดัดแปลง หรือเผยแพร่ต่อสาธารณะนั้นงานอันมีลิขสิทธิ์ จะมีโทษปรับตั้งแต่ 20,000 บาท ถึง 200,000 บาท และหากเป็นการกระทำเพื่อการค้า จะมีโทษจำคุกตั้งแต่ 6 เดือน ถึง 10 ปี หรือปรับตั้งแต่ 100,000 บาท ถึง 800,000 บาท หรือทั้งจำทั้งปรับ

Unless allowed by Mahidol University, the User shall not copy, modify, or exploit in part or in whole of the copyrighted materials on the Platform, other than for your own individual study. The copy, modify or communication to public of the copyrighted materials shall be inflicted with a fine from 20,000 Baht up to 200,000 Baht. If the offence is committed with the commercial purpose, the offender shall be inflicted with imprisonment for a term from 6 months up to 10 years or a fine from 100,000 Baht up to 800,000 Baht or both imprisonment and fine.

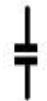


- Introduction
- Digital/Analog Data vs Signal and Devices
- Sensors
- Actuators
- ADC
- DAC
- PWM
- Motors
- Special Issues: Wearable Sensors & Smart Watch



## Basic Electronic Components [Arduino Cookbook]

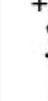
**Capacitor - Unpolarized**



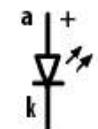
**Capacitor - Polarized**



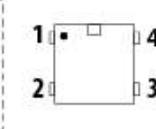
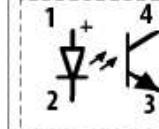
**Diode**



**LED**



**Optocoupler**



**Photoresistor**



**Pot (Potentiometer)**



**Relay**



**Resistor**



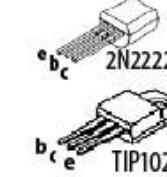
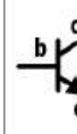
**Stepper Motor**



**Switch**

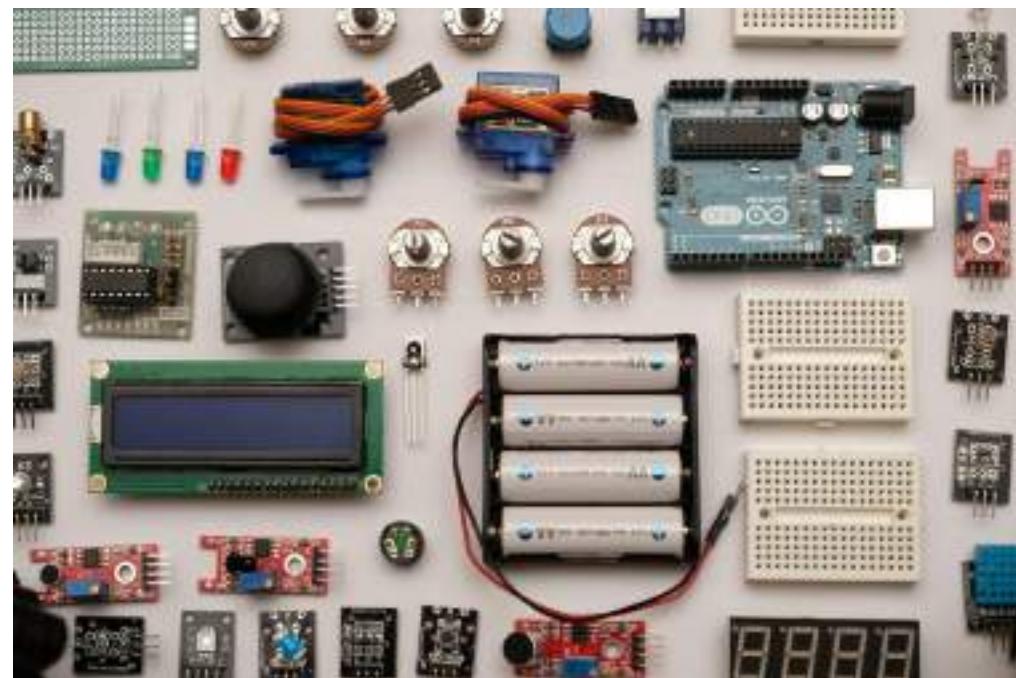


**Transistor**





## IoT Kit Components



[https://makeradvisor.com/tools/?utm\\_source=rnt&utm\\_medium=post&utm\\_campaign=post](https://makeradvisor.com/tools/?utm_source=rnt&utm_medium=post&utm_campaign=post)



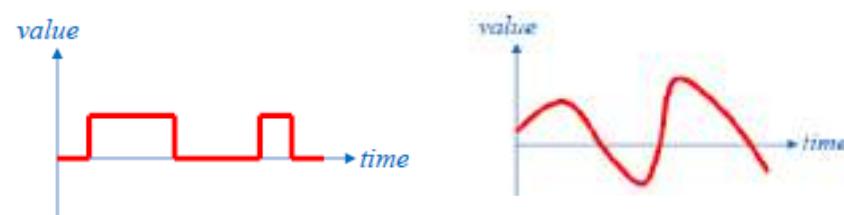
# **Digital/Analog Data vs Signal**

# **Digital/Analog Input/Output Devices**

# Digital/Analog Data vs Digital/Analog Signal



- Digital Data
  - Discrete values
  - Number of items e.g. people walking through in/out a gate (increasing/decreasing counter)
  - Switch position (input)
    - e.g. push button (on/off), dip switch (2x levels)
  - LED (output) – on/off
- Analog Data
  - Continuous values
  - e.g. human voice, temperature reading
- Digital Signals
  - Have a limited number of valid values
- Analog Signals
  - Have an infinite number of values in a range



# Transducers and Digital/Analog Input/Output Devices



- Transducer is a device that converts one form of energy or signal to another e.g.

Microphone:

from sound wave to electrical signal

Speaker:

from electrical signal to sound wave

- Digital/Analog Input/Output Devices



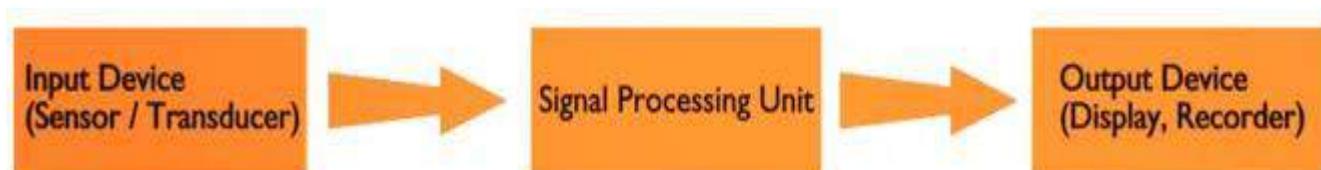
|        | Digital                                                                                         | Analog                                                                                                      |
|--------|-------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| Input  | <ul style="list-style-type: none"><li>• Push switch</li><li>• Proximity sensor</li></ul>        | <ul style="list-style-type: none"><li>• Light sensor</li><li>• Microphone</li><li>• Potentiometer</li></ul> |
| Output | <ul style="list-style-type: none"><li>• LED (on/off)</li><li>• Buzzer</li><li>• Relay</li></ul> | <ul style="list-style-type: none"><li>• LED (dimmable)</li><li>• Motor</li><li>• Servo</li></ul>            |



# Sensors

## Introduction

- Allow the microcontroller to receive information about the environment (input)
  - ✓ How bright is it? (analog)
  - ✓ How loud is it? (analog)
  - ✓ How humid is it? (analog)
  - ✓ Is the button being pressed? (digital)
- Perform operations (SPU) based on the state of the environment (output)
  - ✓ Turn on a light if it's dark out
  - ✓ Voice-controlled operation





## Sensing the Environment

- Microcontrollers sense voltage
  - ✓ `digitalRead(pin)` returns state of a digital pin
  - ✓ `analogRead(pin)` returns the analog voltage on a pin
- Sensor logic must convert an environmental effect into voltage

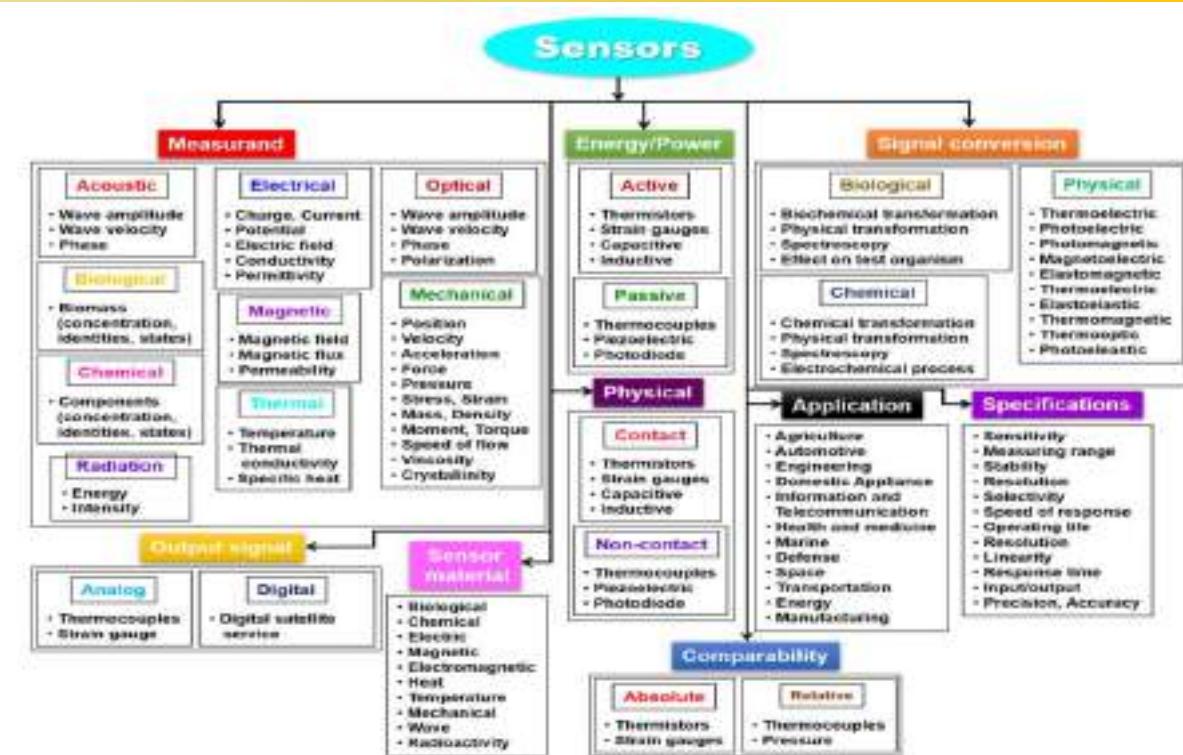
A sensor is a device that responds to any change in physical phenomena or environmental variables like heat, pressure, humidity, movement etc. This change affects the physical, chemical or electromagnetic properties of the sensors which is further processed to a more usable and readable form. Sensor is the heart of a measurement system. It is the first element that comes in contact with environmental variables to generate an output.

<https://www.electrical4u.com/sensor-types-of-sensor/>



## Types of Sensors

- Acoustics, sound, vibration
- Automotive, transportation
- Chemical
- Electric current, electric potential, magnetic, radio
- Flow, fluid velocity
- Ionizing radiation, subatomic particles
- Navigation instruments
- Position, angle, displacement, distance, speed, acceleration
- Optical, light, imaging, photon
- Pressure
- force, density, level
- Thermal, heat, temperature
- Proximity, presence





## Sensor Types for Different Applications [KKC+16]

| Service Category                                                                                                                                                                                                                                                                                                                       | Research and Development Topics                                                                                                                                                                                                                                                                                                                                                                                                         | Sensors                                                                                                                                                                                                                                                                                                   |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"><li>• Energy Management [1]</li><li>• Environmental Monitoring [3]</li><li>• Healthcare [4]</li><li>• Home Automation [6]</li><li>• Home Security [7]</li><li>• Location Recognition [8]</li><li>• Smart LED Lighting [9]</li><li>• Smart Plug and Outlet [10]</li><li>• Smart Switch [11]</li></ul> | <ul style="list-style-type: none"><li>Smart Home Energy Management System</li><li>ZigBee-based Intelligent Self-Adjusting Sensor</li><li>Fall Detection Technology</li><li>Advanced Universal Remote Controller</li><li>Surveillance Robot</li><li>Resident Location-Recognition Algorithm</li><li>Intelligent Household LED Lighting System</li><li>Intelligent Power Gateway and Management Device</li><li>LED Smart Switch</li></ul> | <ul style="list-style-type: none"><li>Light, Humidity, Temperature, PIR</li><li>Temperature, Humidity, Light, Motion, Carbon Monoxide</li><li>Image, Video</li><li>Gas, PIR, Magnetic, Camera</li><li>IR, Camera</li><li>PIR</li><li>CDS, Motion</li><li>Motion, CDS</li><li>Motion, CDS, Touch</li></ul> |

Learn more:

<https://www.electronicshub.org/different-types-sensors/>

<https://instrumentationtools.com/sensors-and-transducers-classification/>

## Categories

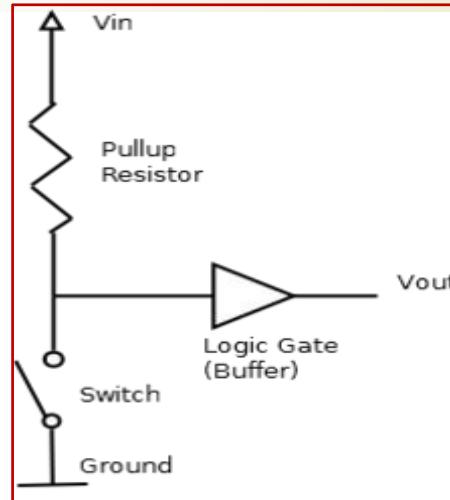
- **Digital Sensors**
  - Buttons and switches
  - On/off devices
  - I2C devices
  - SPI devices
  - RS-232 devices
  - Other Sensors
- **Analog Sensors**
  - Resistance-based sensors
  - Voltage-based sensors
  - Current-based sensors
- **Signal Conditioning**
  - Voltage divider
  - Amplifiers
  - Filters
  - Analog-to-digital conversion
  - Supply voltage
  - Multiplexer



## Desirable Features

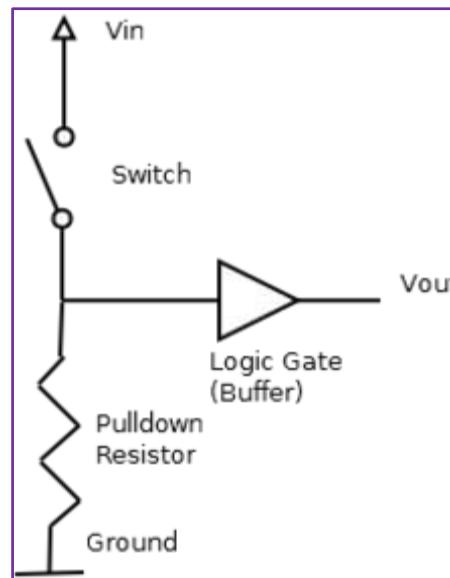
- **Accuracy:** Accuracy should be high. How close output to the true value is the accuracy of the device.
- **Precision:** There should not be any variations in the sensed output over a period of time precision of the sensor should be high.
- **Operating Range:** Sensor should have wide range of operation and should be accurate and precise over this entire range.
- **Speed of Response:** Should be capable of responding to the changes in the sensed variable in minimum time.
- **Calibration:** Sensor should be easy to calibrate time and trouble required to calibrate should be minimum. It should not require frequent recalibration.
- **Reliability:** It should have high reliability. Frequent failure should not happen.
- **Cost and Ease of operation:** Cost should be as low as possible, installation, operation and maintenance should be easy and should not required skilled or highly trained persons

## Buttons and Switches



A **pull-up resistor** weakly “pulls” the voltage of the wire it is connected to towards its voltage source level when the other components on the line are inactive.

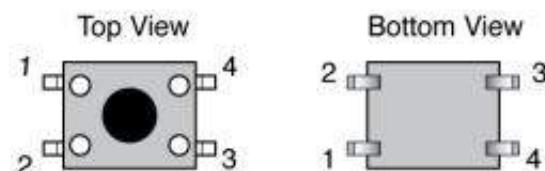
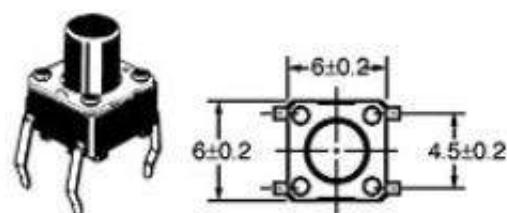
- When all other connections on the line are inactive, they are high impedance and act like they are disconnected.
- **Arduino will read HIGH until circuit is activated and it will switch to LOW.**



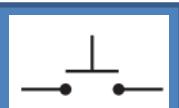
A **pull-down resistor** is connected to ground, and holds the logic signal near zero volts when no other active device is connected.

- **Arduino will read LOW until circuit is activated and it will switch to HIGH.**
- “Make a pin high when the button is pressed, and low when it is not pressed”

## Push Button Switch



Normally Open(NO)



Normally Close(NC)



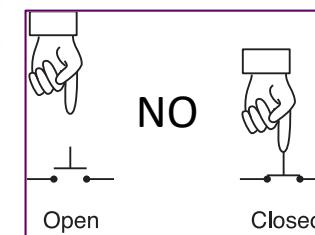
| Measured Resistance ( $\Omega$ ) |                  |              |
|----------------------------------|------------------|--------------|
| Connect Pins                     | When not pressed | When pressed |
| 1 and 2                          | $\infty$         | 0            |
| 1 and 3                          | $\infty$         | 0            |
| 1 and 4                          | 0                | 0            |
| 2 and 3                          | 0                | 0            |

Side 1  
Two connected pins



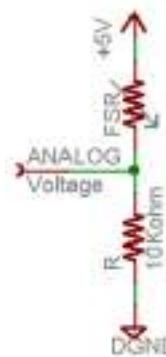
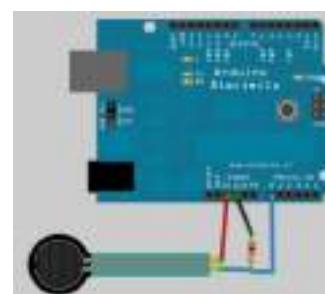
Pushbutton  
Connects two sides  
when pressed

Side 2  
Two connected pins

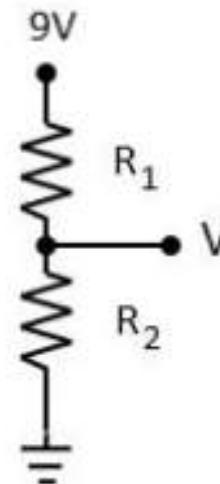
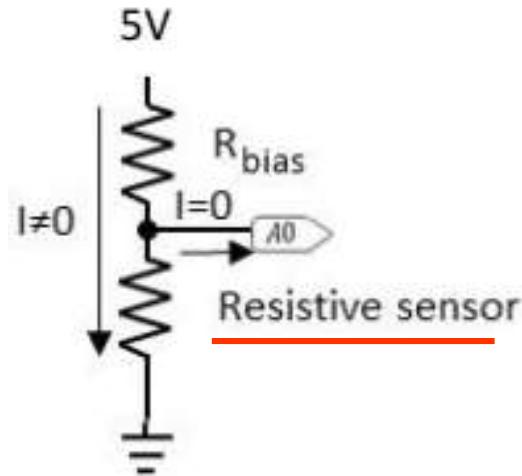


## Resistance-based or Resistive Sensors

- Many sensors change resistance
- For examples:
  - Potentiometer
  - Photoresistor or LDR,
  - thermistor,
  - flex resistor,
  - forcing sensing resistor,
  - fluid (water) level,
  - Gas sensor,
  - Joy stick, etc.
- Connect sensor in a voltage divider



## Resistive Sensors vs Voltage Divider Circuit



$$V_{R_1} = \frac{R_1}{R_1+R_2} V_{source}$$

$$V_{R_2} = V = \frac{R_2}{R_1+R_2} V_{source}$$

$$V_{source} = 9V$$

## Voltage-based Sensors

- To produce directly a voltage as its output pin
- For examples:
  - Temperature sensor: LM35
  - Thermocouples: MLX90614
  - Thermopile
  - Hall sensor: A1324
  - Acceleration sensor: ADXL335, SM-24 (geophone)
  - microphone

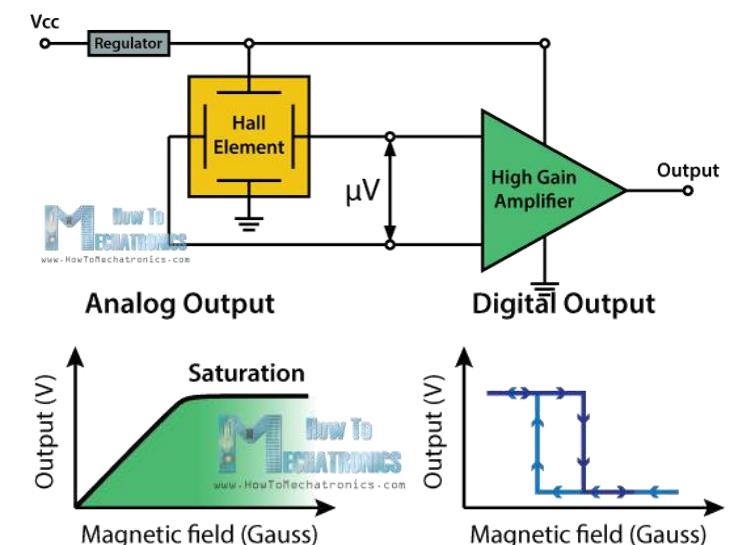


## Current-based Sensors

- To produce directly a current as its output pin
- The current is obtained by measuring the voltage on the resistor and applying formula proportional to the voltage.
- They can be used to determine the power consumption or detect whether the device is turned on or short circuits.
- Other non-invasive measurement methods involve hall effect sensors for DC and AC and inductive coils for AC.

For examples:

- Phototransistors: BPX38 (SFH3310)
- Solar cells
- Amperometric sensor
- Direct Current sensor (DC sensor)
- Proximity sensors
- MEMS Compasses
- Hall Effect Switches
- Wheel Speed Sensors (RPM)



Ref:

Vladislav V. Kharton, "Solid State Electrochemistry I: Fundamentals, Materials and their Applications", John Wiley & Sons, 2009.

Hall Effect: <https://www.youtube.com/watch?v=QMsv9PadpY>, <https://www.youtube.com/watch?v=wPAAA3qeOYi>



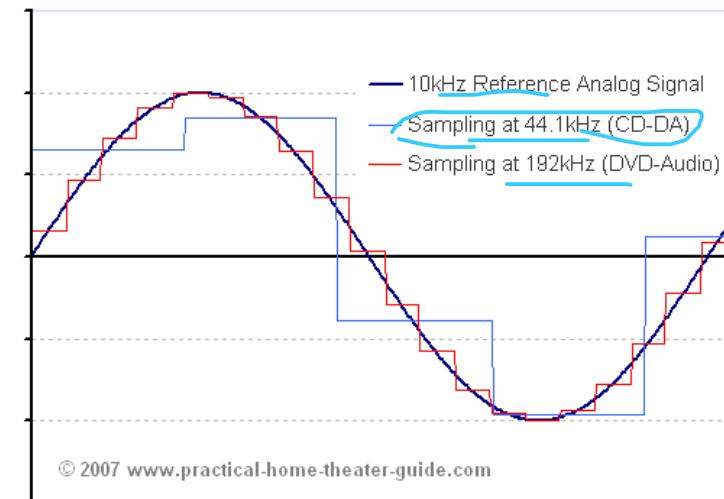
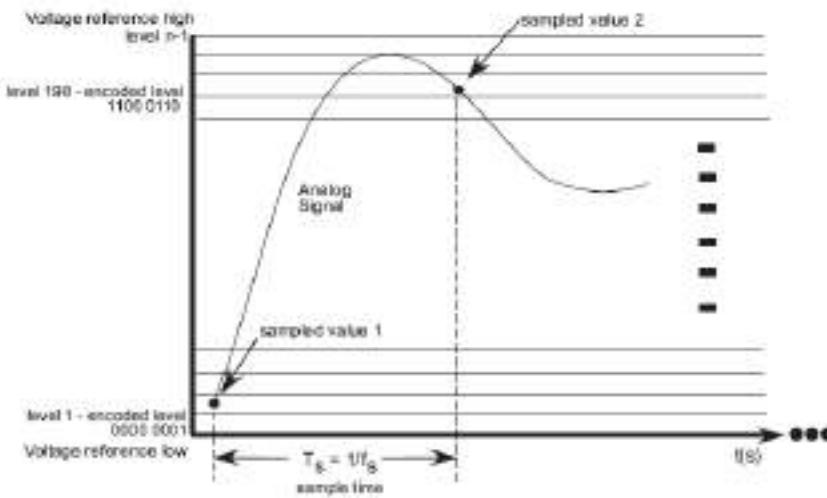
# Analog to Digital Conversion (ADC)

# Analog to Digital Conversion (ADC)



## Overview

- To be processed by microcontrollers, all information must be converted to a binary representation possessing a finite number of distinct values that combine logical low and high.
- Think about music stored on a CD---an analog signal captured on digital media
  - Sampling: sample rate e.g. Nyquist theorem at least 8KHz for human voice signal
  - Quantization
  - Encoding

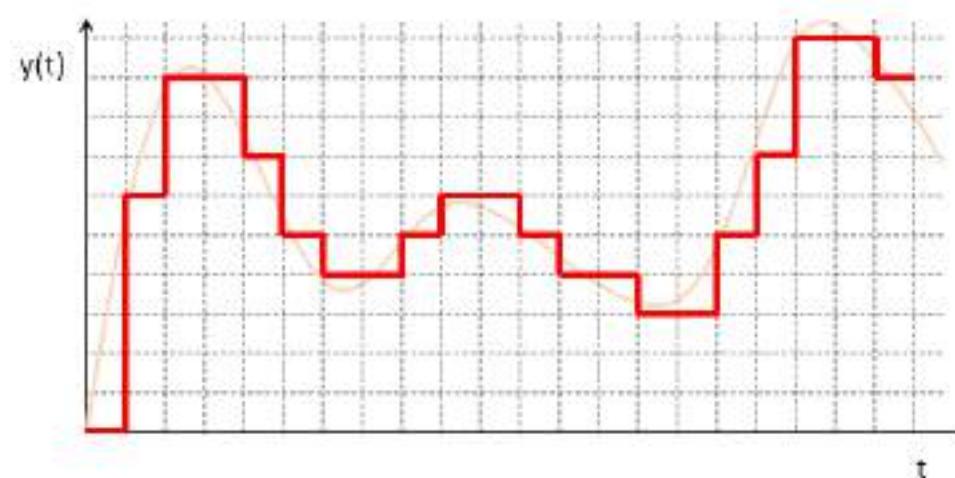




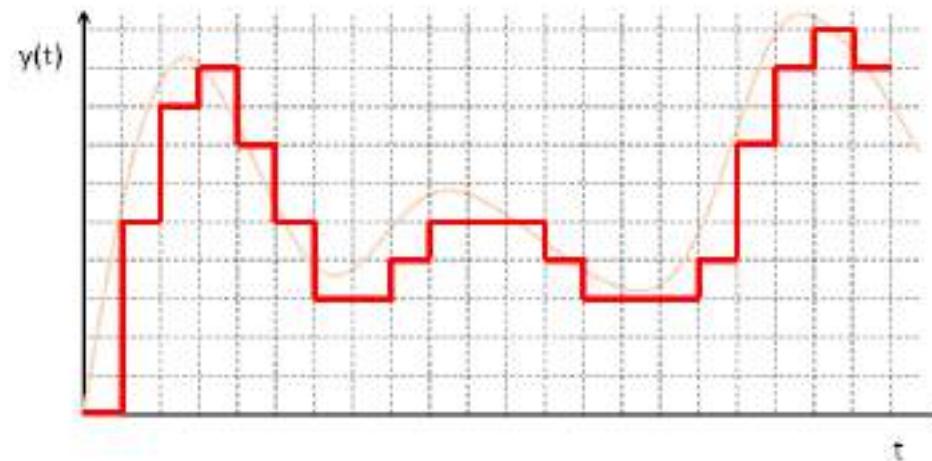
# Analog to Digital Conversion (ADC)

## Rounded vs Truncated Quantization

Rounded Quantization



Truncated Quantization



# Analog to Digital Conversion (ADC)



## Resolution

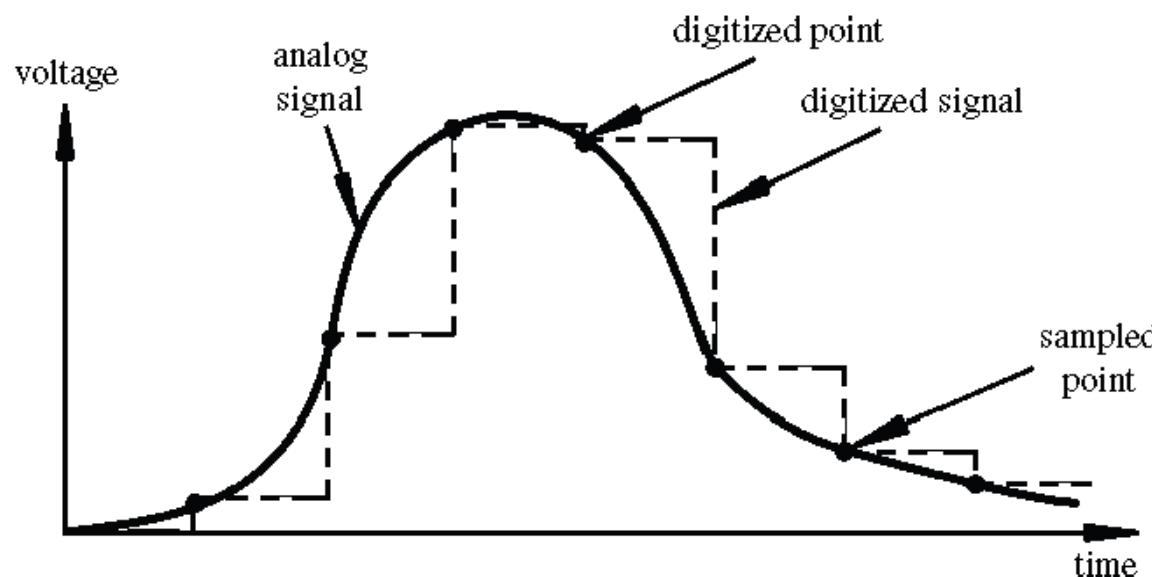


Image credit: Tod Kurt

- **Resolution:** the number of different voltage levels (i.e., *states*) used to discretize an input signal
  - ✓ Arduino:
    - The Arduino uses 1024 states (10 bits)
    - Smallest measurable voltage change is  $5V/1024$  or  $4.8\text{ mV}$
    - Maximum sample rate is 10,000 times a second
  - ✓ ESP32:
    - 8, 10, 12, 15 bits
    - For example: 12 bits: a smallest measurable voltage change is  $3.3V/4096$  or  $0.8\text{ mV}$

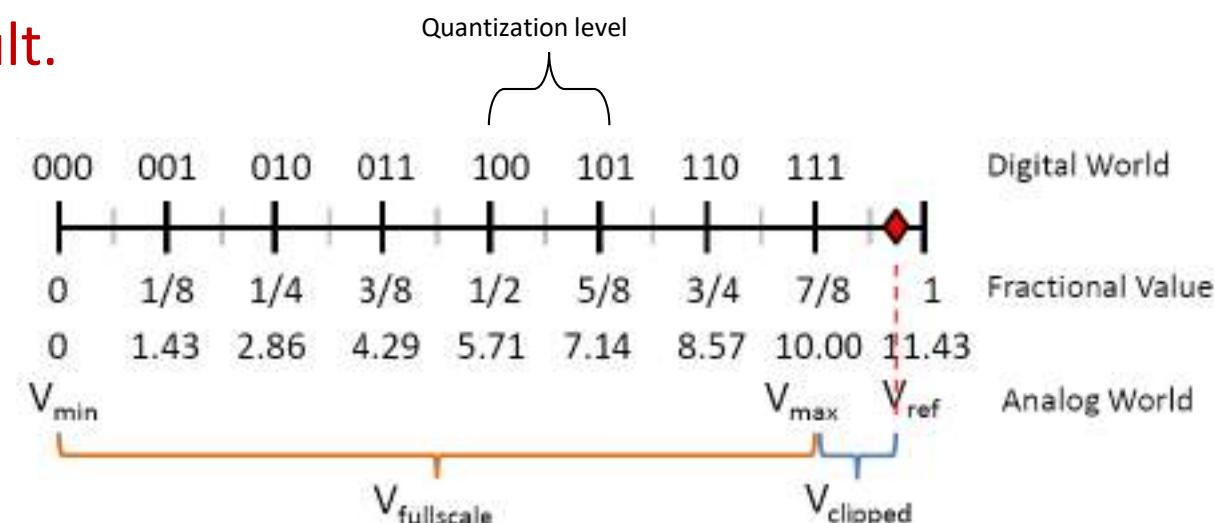


## Digital Representation

- Digital codes have no inherent units, so scaling must be defined for code to have real-world meaning.
- For a unipolar signal, it seems obvious to associate:
  - $V_{min}$  with binary zero
  - $V_{max}$  with binary  $2^N - 1$  and its fraction value of  $1/2^N$
  - If  $V_{min} < 0$  or  $V_{max} > 2^N - 1$ ,  
then amplitude clipping may result.

$$V_{fs} = V_{max} = V_{ref} \left( \frac{2^N - 1}{2^N} \right)$$

$$Q = \frac{V_{fullscale}}{2^N - 1} = \frac{V_{ref}}{2^N}$$



# Analog to Digital Conversion (ADC)



## Integer Code Examples

### Unipolar Voltages

- Can be coded to unsigned integers
- For example: 0 – 5 Volts coded to 3 bit unsigned integer.

| Voltage | Digital Value | Decimal Equivalent |
|---------|---------------|--------------------|
| 0       | 000           | 0                  |
| 0.71    | 001           | 1                  |
| 1.43    | 010           | 2                  |
| 2.14    | 011           | 3                  |
| 2.86    | 100           | 4                  |
| 3.57    | 101           | 5                  |
| 4.29    | 110           | 6                  |
| 5       | 111           | 7                  |

### Bipolar Voltage

- Two's complement
- MSB is treated with a weight of  $-2^{N-1}$

| Voltage(1) | Voltage(2) | Digital Value | Decimal Equivalent |
|------------|------------|---------------|--------------------|
| +3.75      | +5         | 011           | 3                  |
| +2.50      | +3.33      | 010           | 2                  |
| +1.25      | +1.67      | 001           | 1                  |
| 0          | 0          | 000           | 0                  |
| -1.25      | -1.67      | 111           | -1                 |
| -2.50      | -3.33      | 110           | -2                 |
| -3.75      | -5         | 101           | -3                 |
|            | --         | 100           | -4                 |

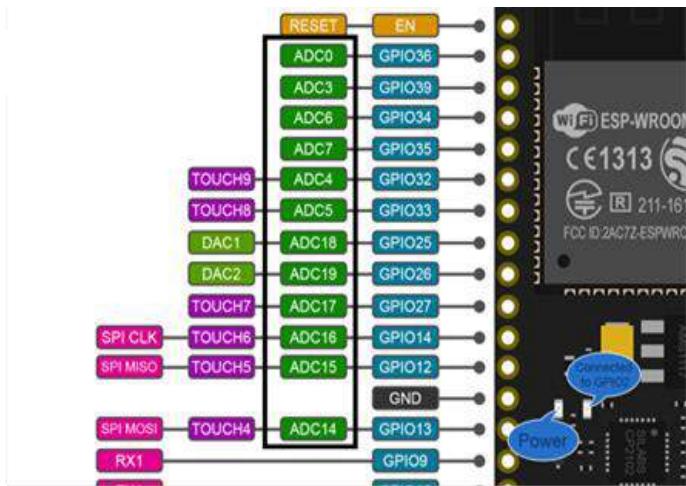
| Voltage | Digital Value | Decimal Equivalent |
|---------|---------------|--------------------|
| -5      | 000           | 0                  |
| -3.57   | 001           | 1                  |
| -2.14   | 010           | 2                  |
| -0.71   | 011           | 3                  |
| +0.71   | 100           | 4                  |
| +2.14   | 101           | 5                  |
| +3.57   | 110           | 6                  |
| +5      | 111           | 7                  |

# Analog to Digital Conversion (ADC)

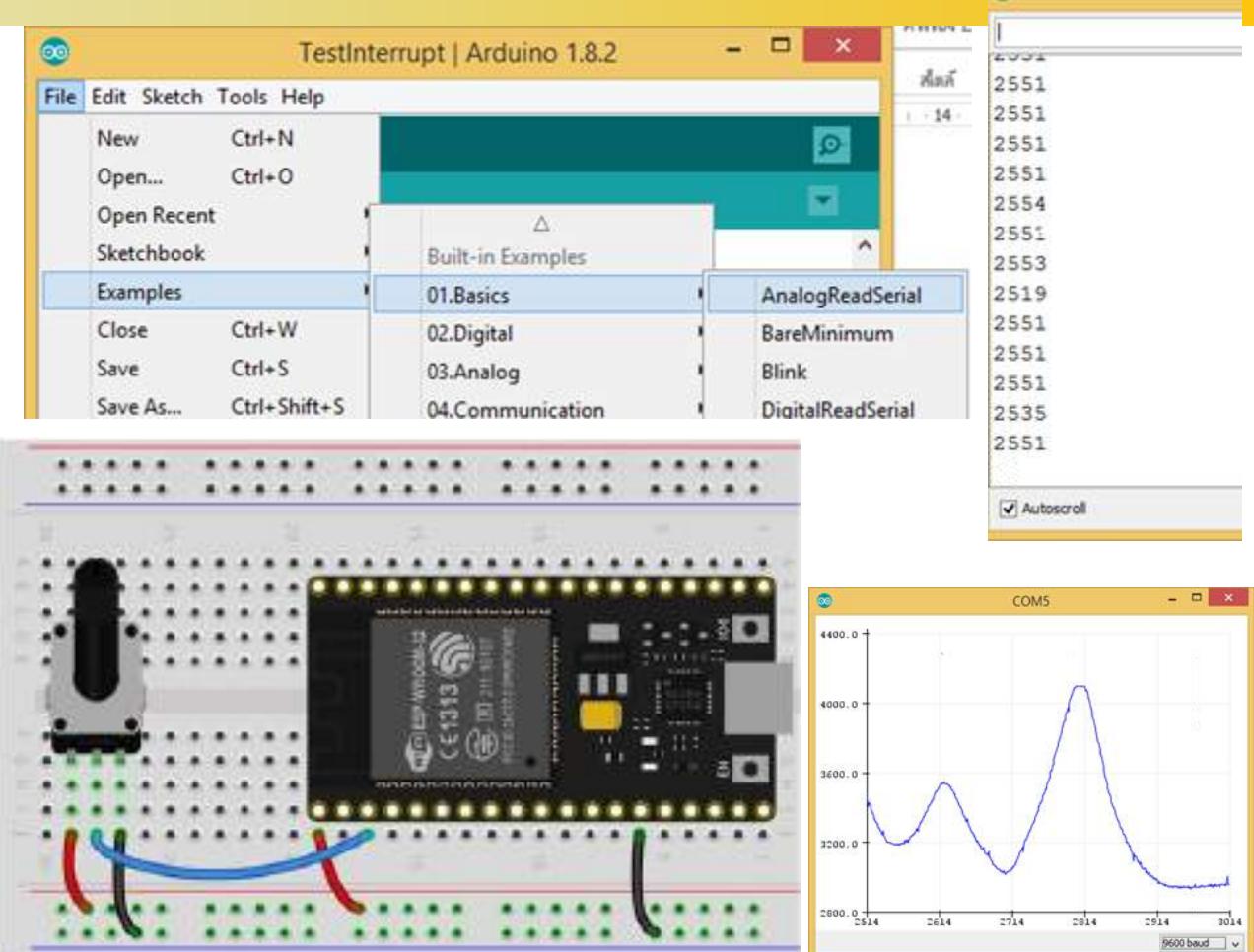


## Example: POT

```
int analogRead(int ch);
```



```
void setup() {  
    Serial.begin(9600);  
    //analogReadResolution(10); //10,12  
}  
void loop() {  
    int sensorValue = analogRead(A0);  
    Serial.println(sensorValue);  
    //Serial.println (sensorValue*3.3/4095); //if res=12  
    delay(1);  
}
```





# Actuators

## Categories

- Buzzer (Active/Passive)
- Switches
  - Light-emitting diodes (LED/RGB) and optocouplers (optoelectronics)  
<https://learn.sparkfun.com/tutorials/light-emitting-diodes-leds/all>
  - Large currents
- Motors
  - Stepper motors
  - Servomotors and model-servos
  - DC motors
- Analog Voltages: Relay
- Display (Details in Lab 6-7):
  - Display: 7-Segment, LCD, Dot Matrix

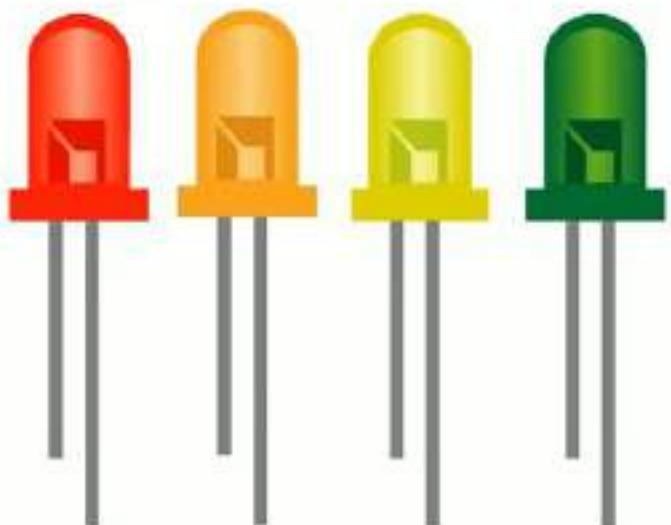
## ACTIVE AND PASSIVE BUZZERS



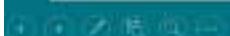
# Buzzer (2)



# What is LED?



Circuit Globe





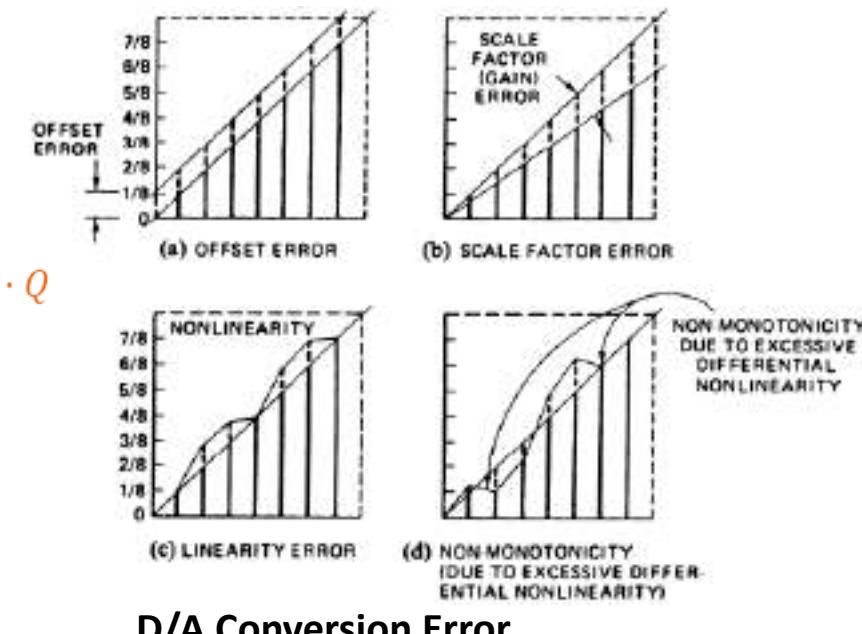
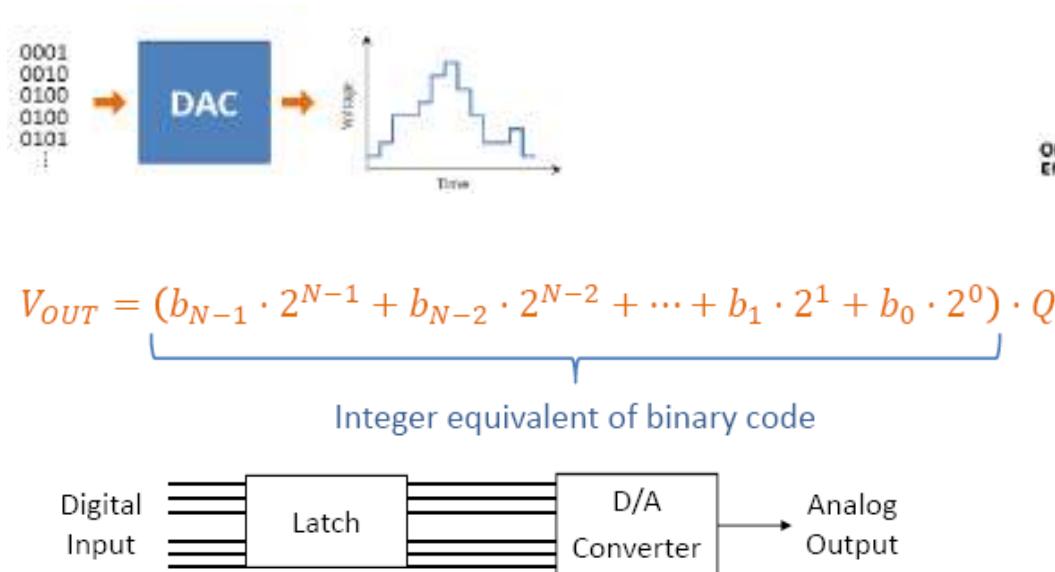
# Digital to Analog Conversion (DAC)

# Digital to Analog Conversion (DAC)



## Overview

- It is to convert digital values to analog outputs of either voltage or current.
- Digital value is stored in a register (latch), then converted.
- Duration of conversion is called *setting time*.
- Output of the DAC remains the same until the next value is sent to the register (latch) – a zero-order hold.



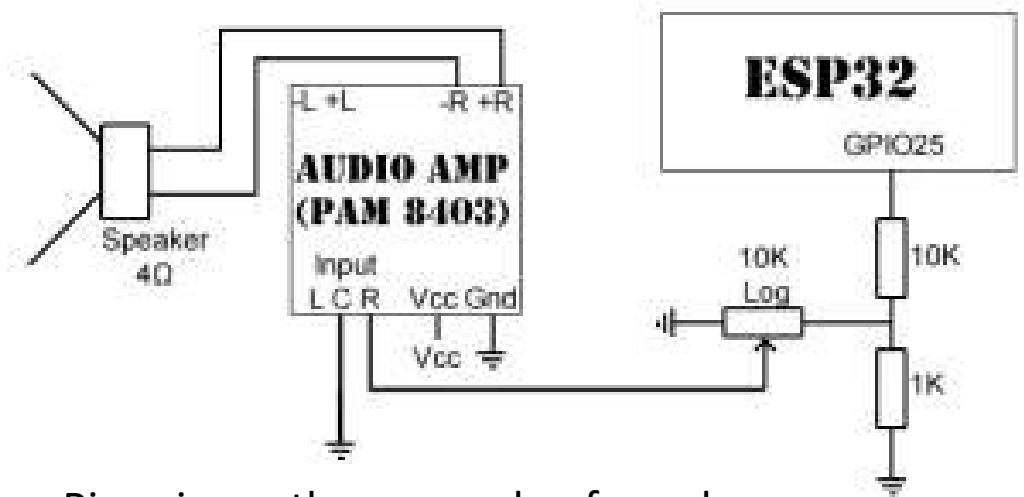
## GPIO 25 (Channel 1), GPIO (Channel 2)

- The ESP32 DAC platform allows you to output analog voltages using the **8-bit digital-to-analog converter** of the ESP32.
- Unlike the ESP32 LEDC Output, which can simulate an analog signal by **using a fast switching frequency**, the hardware DAC can output **a real analog signal with no need for additional filtering**.

```
// Test DAC
void setup()
{
}

void loop()
{
    for(int i=0;i<256;i+=32){
        dacWrite(25,i);
        delay(1500);
    }
    dacWrite(25,i);
}
```

<https://www.xtronical.com/basics/audio/dacs-on-esp32/>  
[https://esphome.io/components/output/esp32\\_dac.html](https://esphome.io/components/output/esp32_dac.html)



Piezo is another example of speaker.

## Use Cases

- DAC (output) with I2S to playing WAV music file from sdcard
- Generating a specific (and dynamic) reference voltage for an external sensor or ADC, such as the [ADS1115 Sensor](#)
- Controlling the bias of a transistor
- Driving a bar graph or large amount of LEDs using an analog-controlled LED driver like the LM3914 ([datasheet](#)); this can allow you to make tank level indicators, temperature gauges, and so on from a single output pin
- Generating 0-10 V for a dimmable light (operational amplifier required)

<http://www.iotsharing.com/2017/07/how-to-use-arduino-esp32-i2s-to-play-wav-music-from-sdcard.html>

<https://github.com/nhatuan84/esp32-i2s-sdcard-wav-player>

ADCS1115 – 16-bit ADC <https://esphome.io/components/sensor/ads1115.html>



# Pulse Width Modulation (PWM)

# Digital to Analog Conversion (DAC)

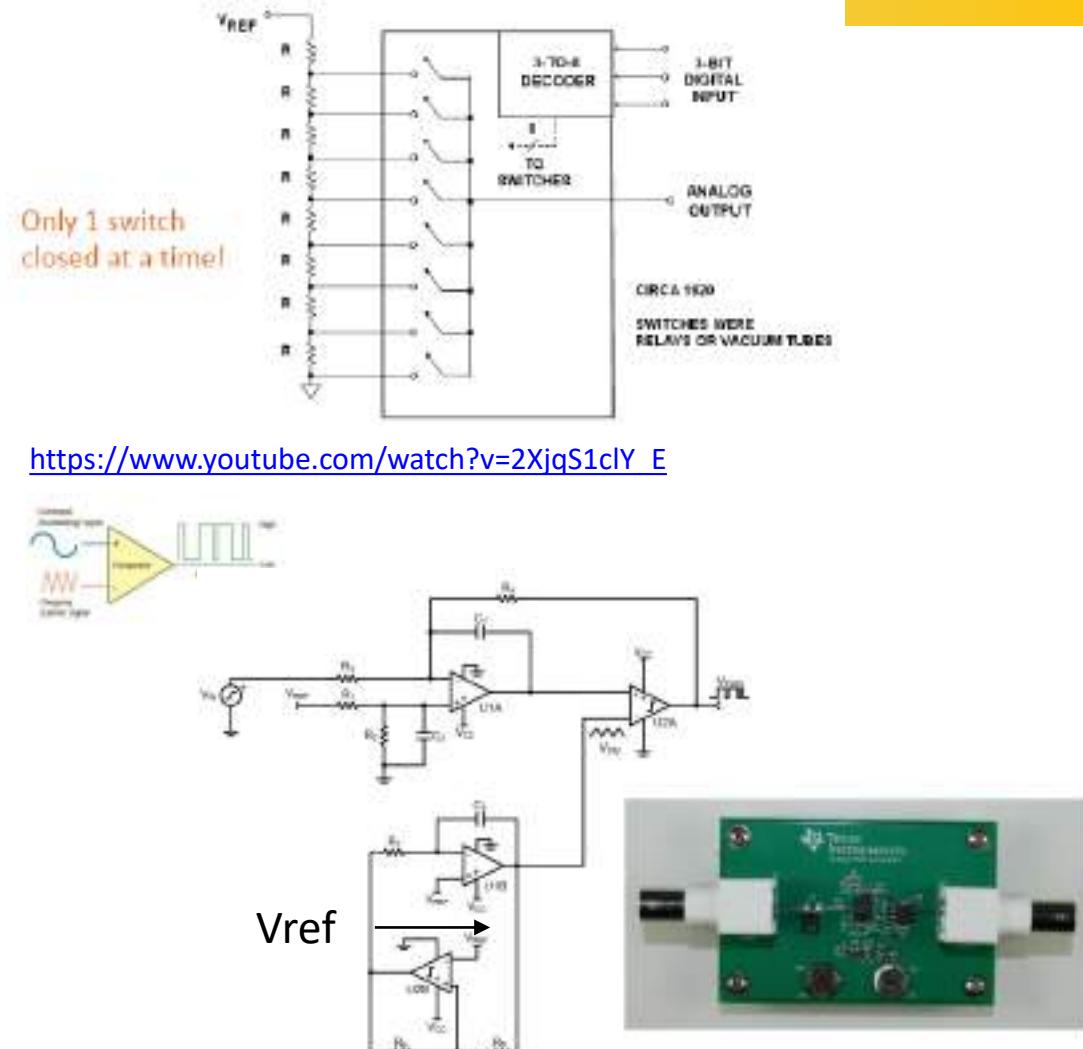


## Pulse Width Modulation (PWM)

- Resistor methods rely on voltage dividers with Vref
  - Many precision resistors necessary
  - Wasted energy dissipated as heat
- PWM (Pulse Width Modulation)
  - Rectangular pulse wave
  - Duty cycle controls average voltage
  - Very high frequency content
  - Need a low-pass filter to remove the sharp transitions at edges of the pulses.
  - About 90% efficiency

$$V_{OUT} = -\sum I_i \frac{R}{2} = -V_{REF} \left( \frac{b_{N-1}}{2} + \frac{b_{N-2}}{4} + \dots + \frac{b_0}{2^N} \right)$$

$$V_{OUT} = (b_{N-1} \cdot 2^{N-1} + b_{N-2} \cdot 2^{N-2} + \dots + b_1 \cdot 2^1 + b_0 \cdot 2^0) \cdot \frac{V_{REF}}{2^N}$$



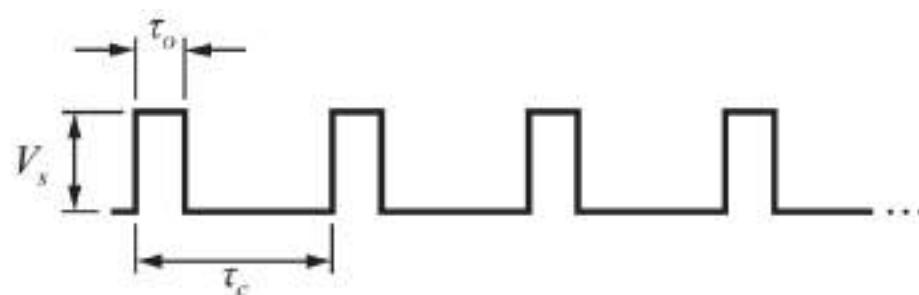
[https://www.youtube.com/watch?v=2XjqS1cIY\\_E](https://www.youtube.com/watch?v=2XjqS1cIY_E)

# Pulse Width Modulation (PWM)



Analog Output: Arduino Uno" 3,5,6,9,10,11 ESP32: All Pins except 34-39

- Arduino boards do not have arbitrary voltage output
- PWM is easy to implement and greatly extends the range of control applications with microcontrollers in general and Arduino in particular.
  - It is to modulate the pulse width.
- The frequency of pulses is fixed but the width of the pulse is variable.
- PWM is a common technique for supplying variable power to “slow” electrical devices such as LEDs and DC motors.
  - “Slow” means that the frequency of the PWM pulse train is much faster than the response time of the device



The ratio  $\tau_0/\tau_c$  is called the duty cycle.

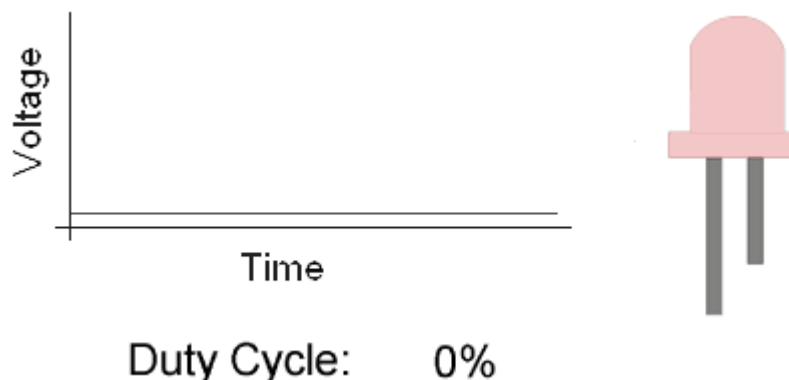
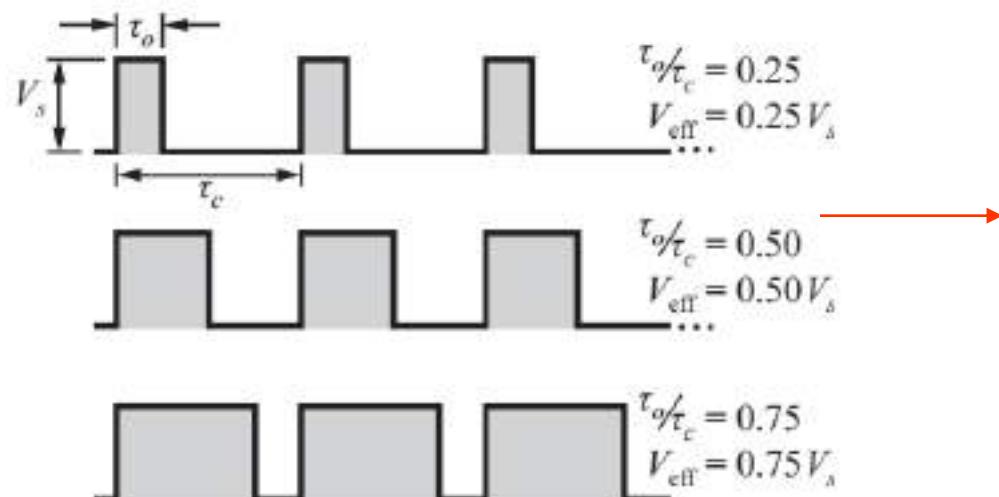
The effective voltage is  $V_{\text{eff}} = V_s \times \tau_0/\tau_c$

# Pulse Width Modulation (PWM)



## Analog Output

- Varying the duty cycle produces variable  $V_{\text{eff}}$



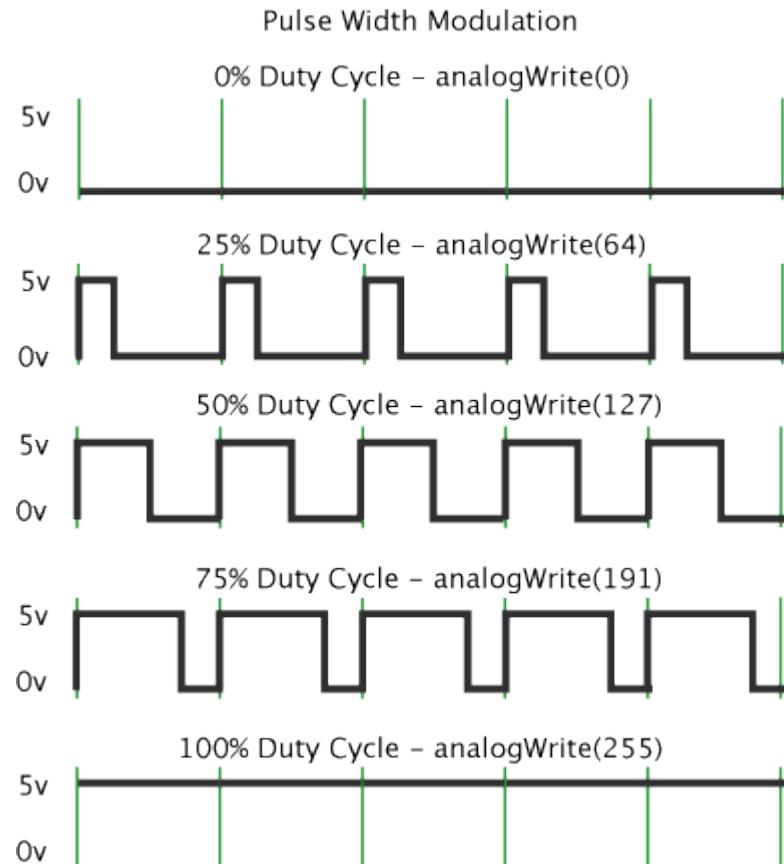
# Pulse Width Modulation (PWM)



## Analog Output Programming

Arduino \*

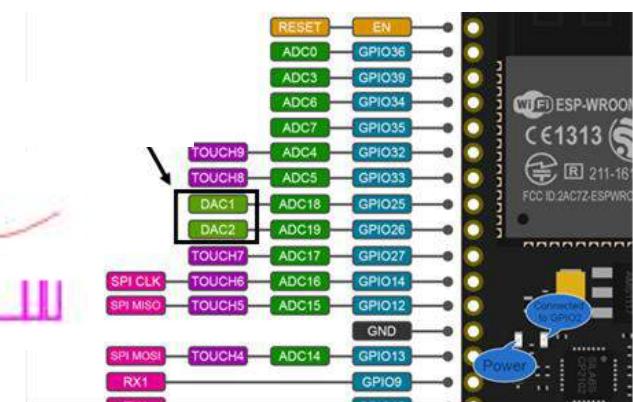
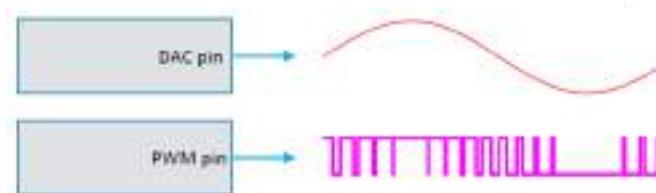
```
analogWrite(level)
```



ESP32

```
ledcAttachPin(gpio, channel);  
ledcSetup(channel, frequency, resolution);  
ledcWrite(channel, dutycycle);  
ledcRead (channel);  
ledcWriteToone (channel, frequency);  
ledcWriteNote (channel, note, octave);  
ledcReadFreq (channel);  
ledcDetachPin (gpio);
```

DAC <> PWM

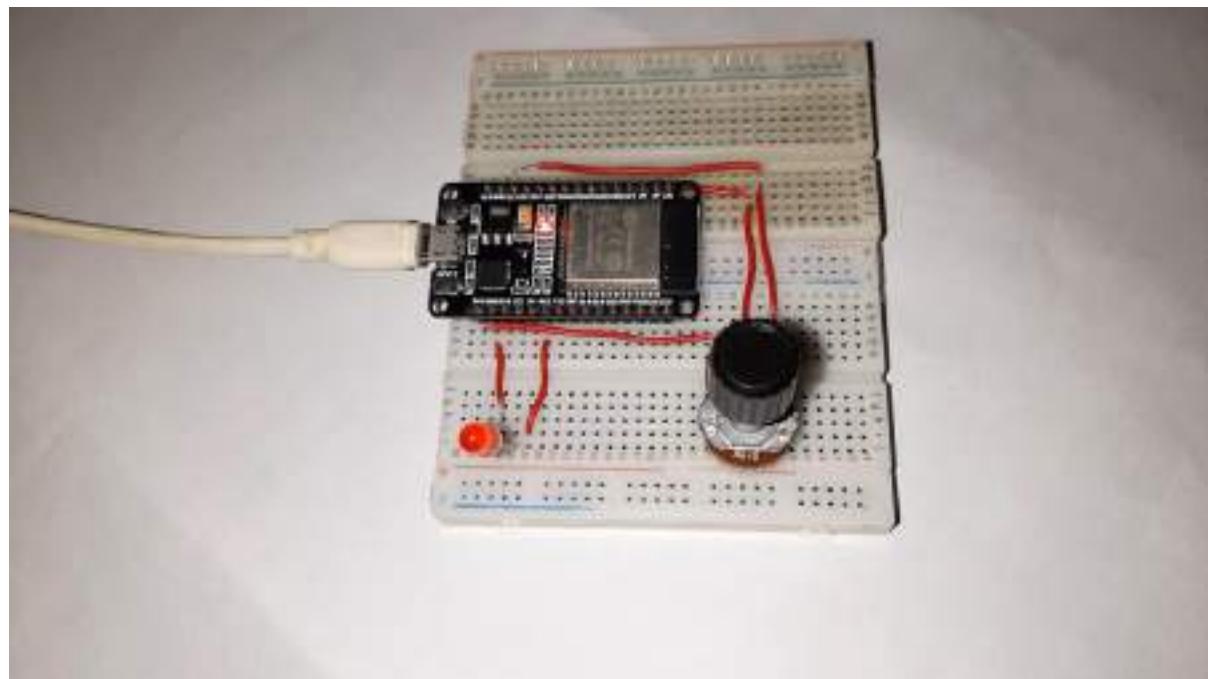


<https://www.electronicshub.org/esp32-pwm-tutorial/>

# Pulse Width Modulation (PWM)



## Example: Dimming LED



ESP32

```
analogRead(gpio)
ledcAttachPin(gpio, channel)
ledcSetup(channel, frequency, resolution)
ledcWrite(channel, dutycycle)
```

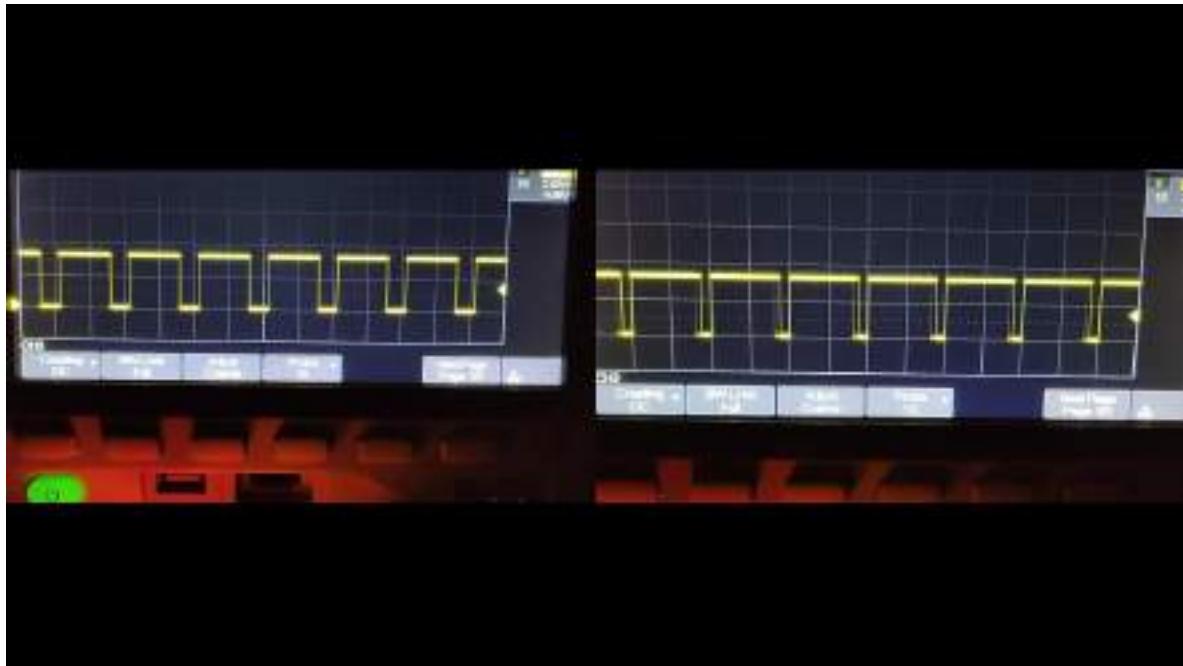
```
const int ledPin = 16; // 16 corresponds to GPIO16
uint16_t dutyCycle;
// setting PWM properties
const int freq = 15000;
const int ledChannel = 0;
const int resolution = 13;
void setup(){
    Serial.begin(9600);
    // configure LED PWM functionalitites
    ledcSetup(ledChannel, freq, resolution);
    // attach the channel to the GPIO to be controlled
    ledcAttachPin(ledPin, ledChannel);
}
void loop(){
    dutyCycle = analogRead(A0);
    Serial.print(dutyCycle);
    // changing the LED brightness with PWM
    ledcWrite(ledChannel, dutyCycle);
    delay(15);
}
```

Ref:  
<https://circuitdigest.com/microcontroller-projects/esp32-pwm-tutorial-controlling-brightness-of-led>

# Pulse Width Modulation (PWM)



## Example: Dimming LED: 8bits vs 4bits



ESP32

```
analogRead(gpio)
ledcAttachPin(gpio, channel)
ledcSetup(channel, frequency, resolution)
ledcWrite(channel, dutycycle)
```

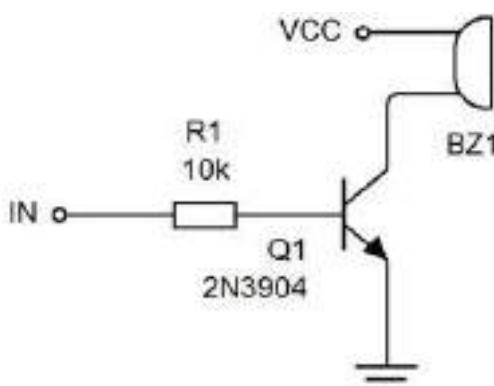
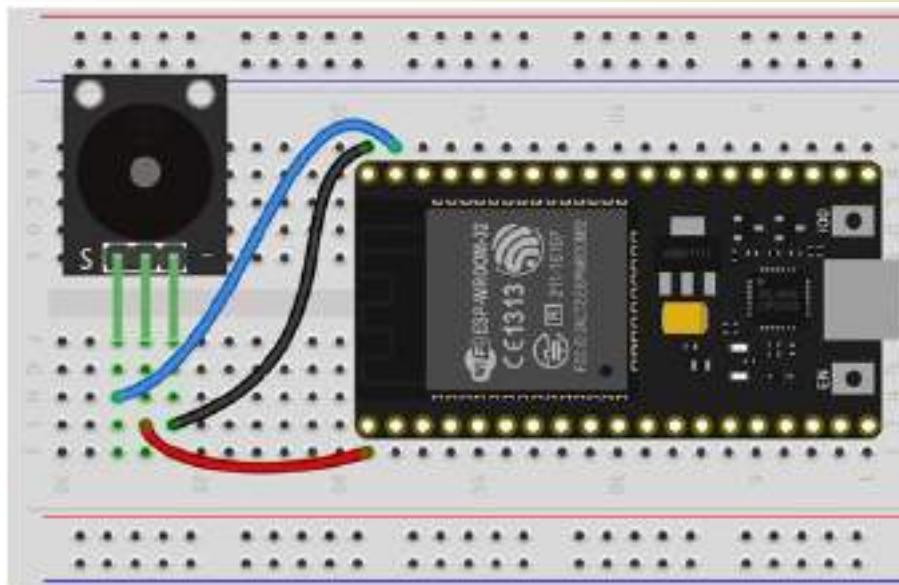
Ref:

<https://circuitdigest.com/microcontroller-projects/esp32-pwm-tutorial-controlling-brightness-of-led>

# Pulse Width Modulation (PWM)



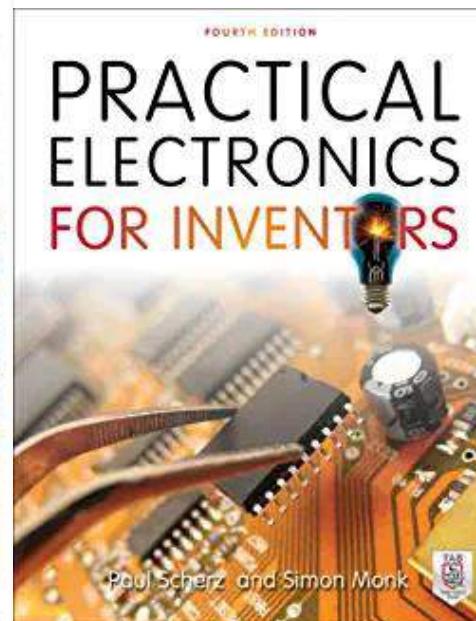
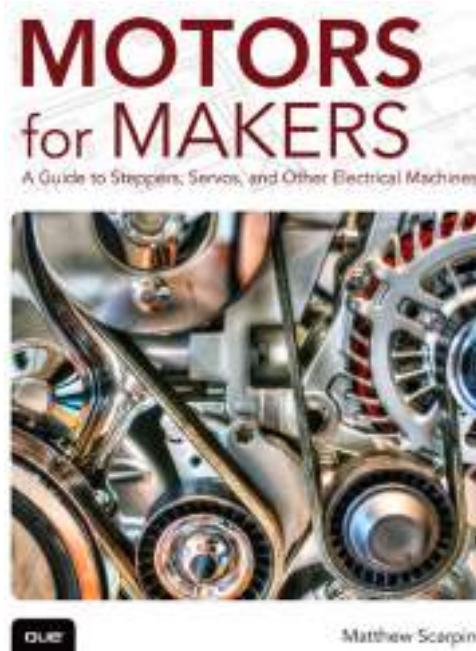
## Example: Buzzer



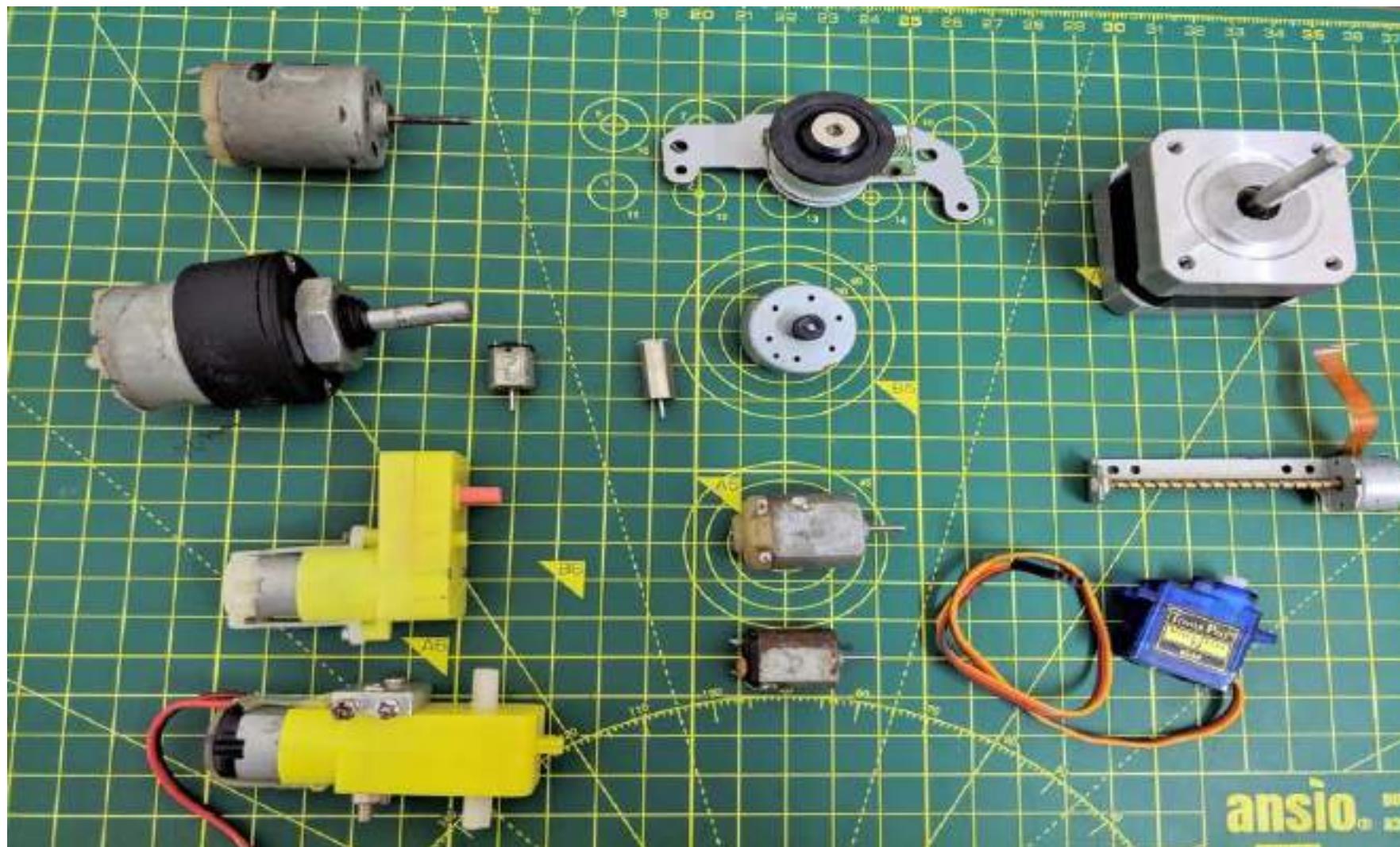
```
int buzzer_pin = 23;  
  
void setup() {  
    ledcWriteTone(0, 2000);  
}  
  
void loop() {  
    ledcAttachPin(buzzer_pin, 0);  
    delay(500);  
    ledcDetachPin(buzzer_pin);  
    delay(500);  
}
```



# Motors (Actuator)



## **Types of Motor**



## Types of Motor

AC Motor

AC Linear Motor

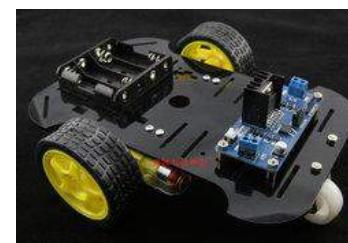


AC Induction Motor

AC Synchronous Motor

Stepper Motor

Microstepping Motor



DC Motor

Brushless DC motor



Permanent-magnet DC motor

Servo Motor

AC Servo Motor



Other Motors

DC Servo Motor



Voice Coil Motor



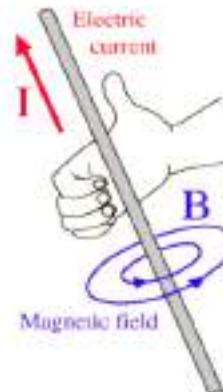
Ultrasonic

## The Foundation for Motors

- Electromagnetism has two different facets: Electricity and magnetism
  1. a moving electric charge produces magnetic fields
  2. changing magnetic fields move electric charges
- Theories developed by: Faraday, Maxwell, Lenz
  - A static distribution of charges produces an electric field
  - Charges in motion (i.e. an electrical current) produce a magnetic field.

### Right-hand rule:

Wrap your right hand around a conductor with thumb pointing in direction of current flow, your fingers curl in the direction of the field.



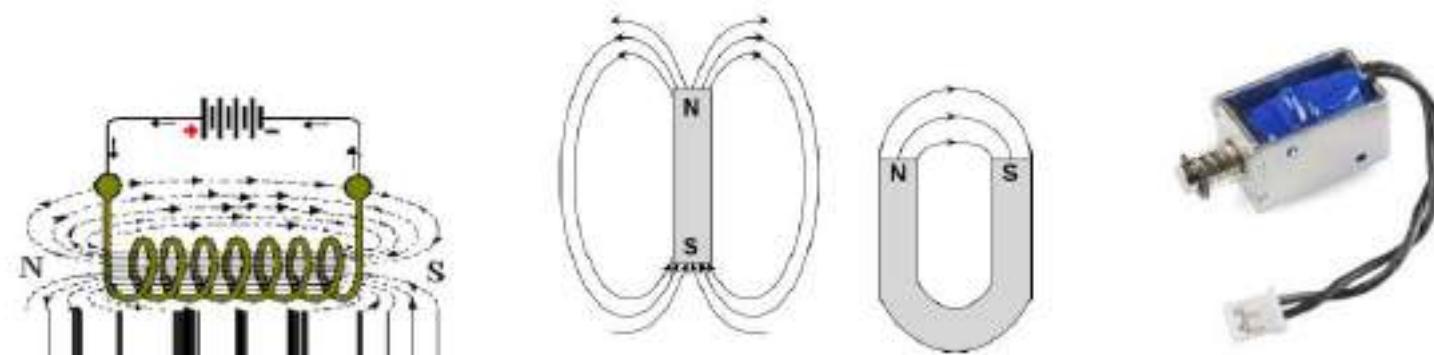
Curl your fingers in the direction of the current, your thumb points toward the north.



# Electromagnets



- Arranging wire in a coil and running a current through produces a magnetic field that looks a lot like a bar of permanent magnet:
  - This is called an electromagnet
- **Putting an iron or steel rod inside the coil makes the electromagnet stronger** – the iron is magnetic, it concentrates and amplifies the magnetic field created by the current in the coil.
- Putting a permanent magnet inside, one can move the magnet back and forth depending on current direction: called a solenoid.
- A solenoid is a simple device illustrating how the interaction between the magnetic field from a permanent magnet interacts with the magnetic field produced by an electromagnet to produce force.



Ref: [www.ee.ic.ac.uk/pcheung/teaching/DE1\\_EE](http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE)



## Faraday Demonstration



## How does it work?



## Stepper or Stepping Motor (Digital Output)

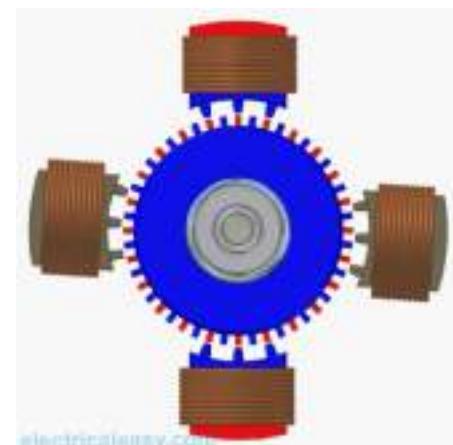
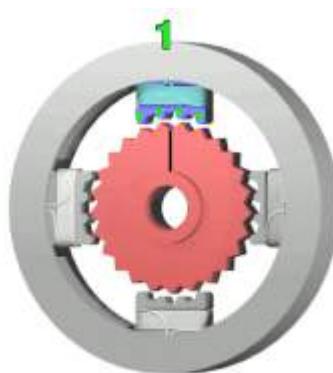
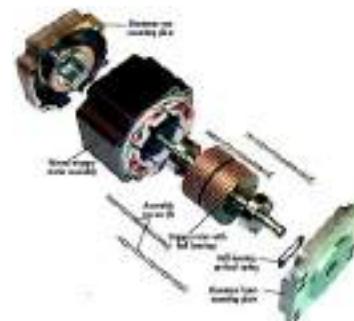
- Utilize multiple toothed electromagnets arranged around a central gear to define position.
- Full rotation of the motor is divided into small, equal steps. Stepper motor has many individually controlled electromagnets, by turning them on or off, a motor shaft rotates by one step.
- Changing switching speed or direction can precisely control turn angle, direction or full rotation speed.
- Because of exact control ability, they are used in CNC machines, 3D printers, scanners, hard drives etc.

<https://www.robotpark.com/Stepper-Motor-Working>

[https://www.youtube.com/watch?v=JgZiwYYrntM&fbclid=IwAR3A5zQojnDQtCWb77M-XLjCDbeUr03rlSd\\_TC9YZ0w4\\_xzpOF2oz6FUoQ](https://www.youtube.com/watch?v=JgZiwYYrntM&fbclid=IwAR3A5zQojnDQtCWb77M-XLjCDbeUr03rlSd_TC9YZ0w4_xzpOF2oz6FUoQ)

## Principle of Stepper Motor – Full Steps

- The rotor of a permanent magnet in stepper motor consists of permanent magnets and the stator has two pairs of windings.
- Just as the rotor aligns with one of the stator poles, the second phase is energized.
- The two phases alternate on and off and also reverse polarity.
- There are four FULL steps. One phase lags the other phase by one step. This is equivalent to one forth of an electrical cycle or  $90^\circ$ .
- Electronic circuits are used to switch supply voltages to the appropriate windings in the stator to advance each step.



[https://www.youtube.com/watch?v=JgZiwYYrntM&fbclid=IwAR3A5zQojnDQtCWb77M-XLjCDbeUr03rlSd\\_TC9YZ0w4\\_xzpOF2oz6FUoQ](https://www.youtube.com/watch?v=JgZiwYYrntM&fbclid=IwAR3A5zQojnDQtCWb77M-XLjCDbeUr03rlSd_TC9YZ0w4_xzpOF2oz6FUoQ)



## Principle of Stepper Motor – Half Steps

- By energizing two windings some of the time, we get a half-step stepper motor.
- The commutation sequence for a half-step stepper motor has eight steps instead of four.
- The main difference is that the second phase is turned on before the first phase is turned off. Thus, sometimes both phases are energized at the same time.
- During the half-steps the rotor is held in between the two full-step positions.
- A half-step motor has twice the resolution of a full step motor. It is very popular for this reason.
- Step resolution can also be increased with more poles in the stator, resulting in more steps.
- Stepper motors are normally run open loop without sensors or feedback.
- Good for applications such as printers.

[https://www.youtube.com/watch?v=JgZiwYYrntM&fbclid=IwAR3A5zQojnDQtCWb77M-XLjCDbeUr03rlSd\\_TC9YZ0w4\\_xzpOF2oz6FUaQ](https://www.youtube.com/watch?v=JgZiwYYrntM&fbclid=IwAR3A5zQojnDQtCWb77M-XLjCDbeUr03rlSd_TC9YZ0w4_xzpOF2oz6FUaQ)

## Principle of Stepper Motor – Full vs Half Steps

<https://www.youtube.com/watch?v=cYAPK7xoISE>

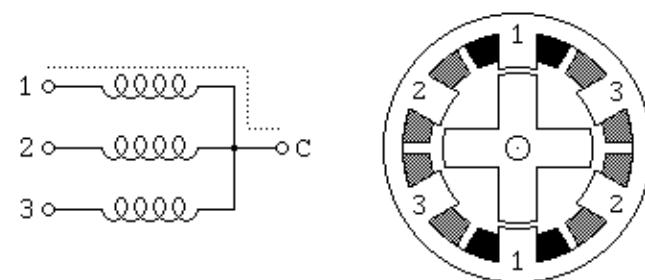
<https://www.youtube.com/watch?v=dmk6zIkj7WM>

## Stepper or Stepping Motor (Digital Output)

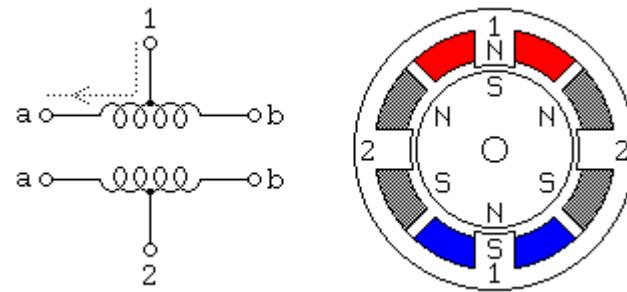


## Stepper Motor Types

- **Variable Reluctance Motors**

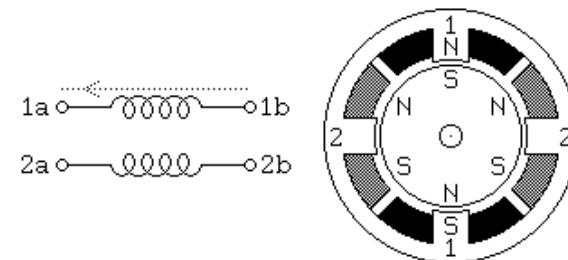


- **Unipolar Motors**



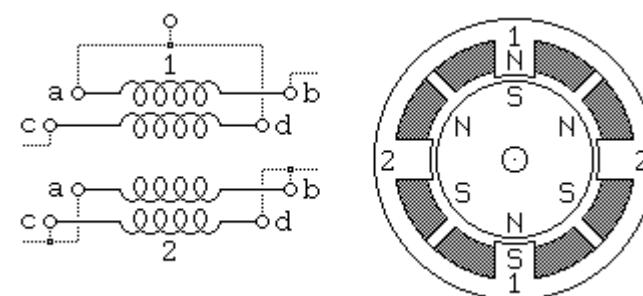
Unipolar stepper motor turns only one step at a time in either direction depending on the control

- ▶ **Bipolar Motors**



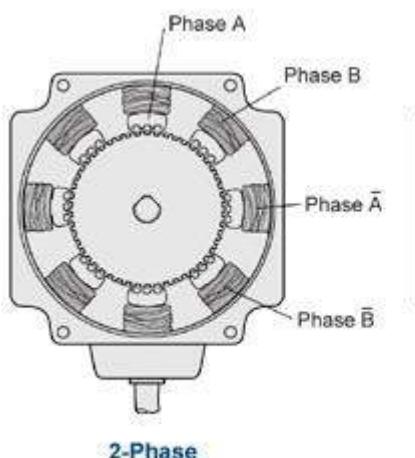
Bipolar stepper motor doesn't have a common lead like in a unipolar stepper motor. There is no natural reversal of current direction through the winding.

- ▶ **Bifilar Motors**

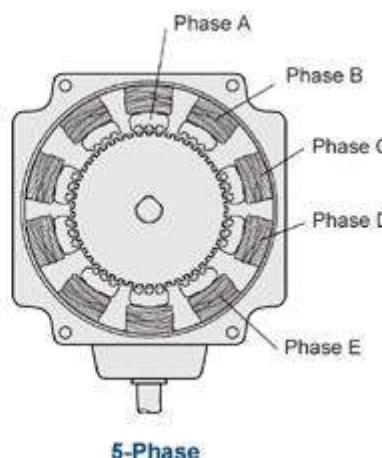




## Stepper Motor

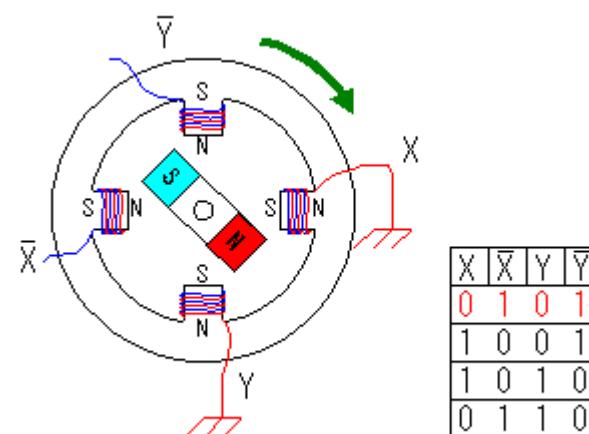


2-Phase



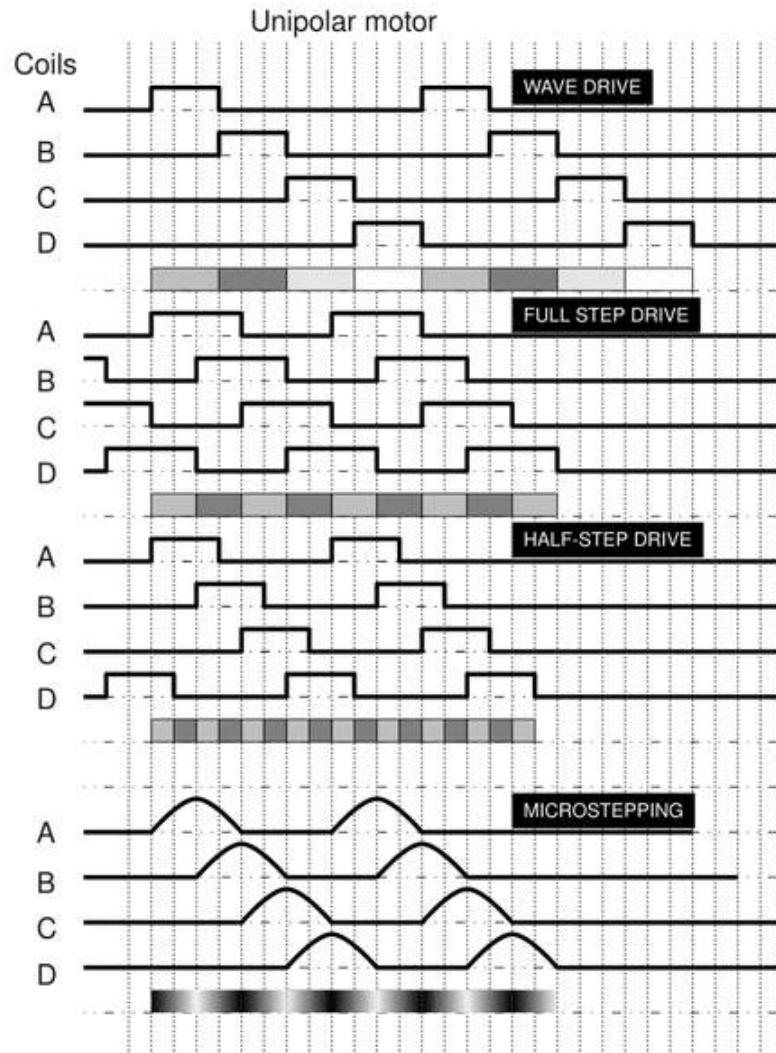
5-Phase

- When electromagnet 'A' is powered it attracts the gear's teeth and aligns them, slightly offset from the next electromagnet 'B'.
- When 'A' is switch off, and 'B' switched on, the gear rotates slightly to align with 'B', and so on around the circle, with each electromagnet around the gear energizing and de-energizing in turn to create rotation.
- Each rotation from one electromagnet to the next is called a "step", and thus the motor can be turned by precise pre-defined step angles through a full 360 Degree rotation.

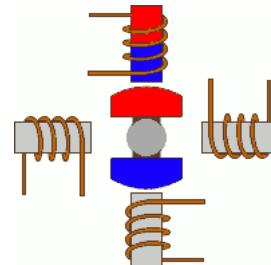




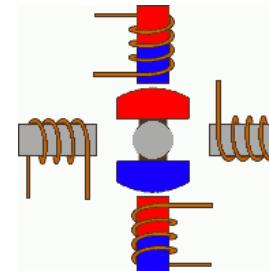
## Stepper Motor Driving Methods (1/2)



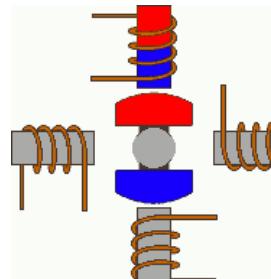
Wave Drive: single phase full step



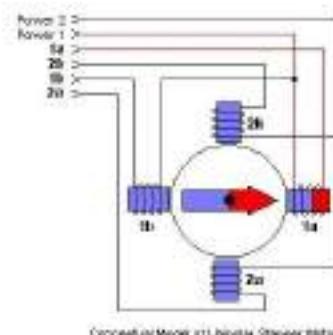
Full Step Drive : two phase full step



Half Stepping : one/two phase half step

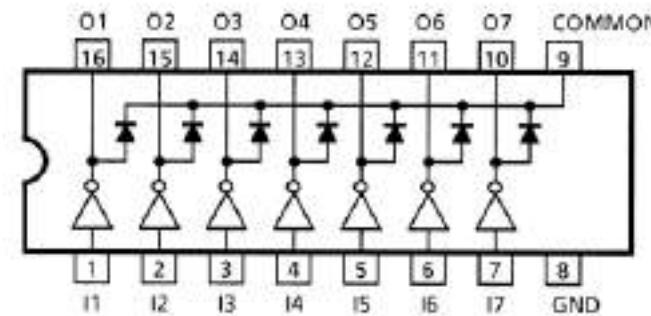
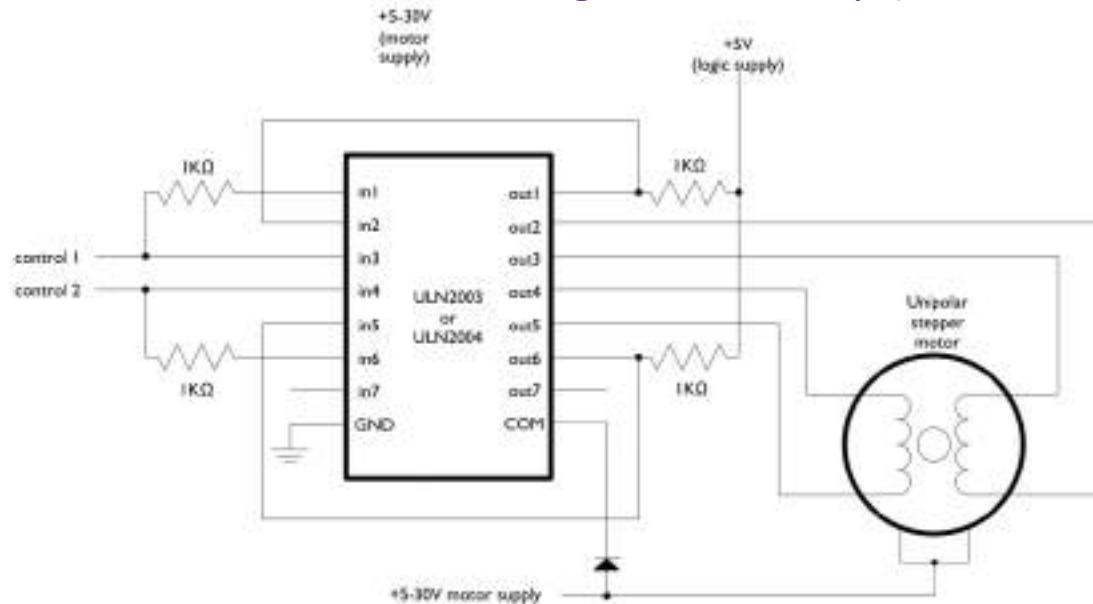


Microstepping

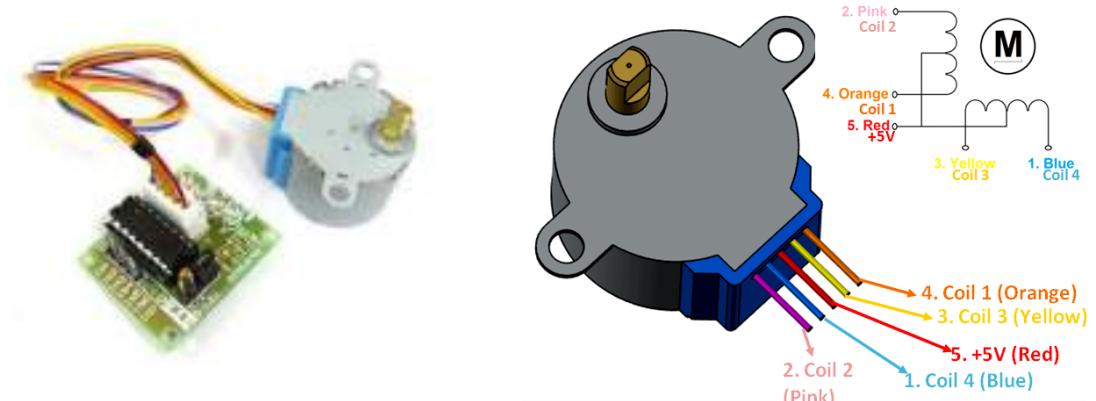


## Stepper Motor Driving Methods (2/2)

- Need a ULN2003APG Darlington driver chip (7 channel transistor arrays)



|              | Stepping P1 | Stepping P2 | Stepping P3 | Stepping P4 |
|--------------|-------------|-------------|-------------|-------------|
| Arduino Uno  | pin 7       | pin 8       | pin 12      | pin 13      |
| Arduino Nano | pin 12      | pin 13      | pin A0      | pin A1      |
| ESP32        | pin 14      | pin 12      | pin 13      | pin 15      |



- Driver IC Interface
  - Control Pins : INA,INB,INC,IND
  - Power Pins : Vcc, Gnd

## Stepper Motor Control Example (1/2)

```
#Raspberry Pi Python sample code
#From https://www.rototron.info/raspberry-pi-stepper-motor-tutorial/
from time import sleep
import RPi.GPIO as GPIO

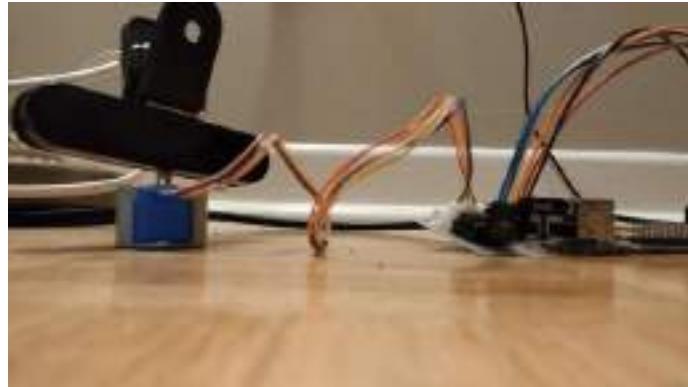
DIR = 20      #Direction GPIO Pin
STEP = 21     #Step GPIO Pin
CW = 1        #Clockwise Rotation
CCW = 0       #Counterclockwise Rotation
SPR = 48      #Steps per Revolution (360/7.5)

GPIO.setmode(GPIO.BCM)
GPIO.setup(DIR, GPIO.OUT)
GPIO.setup(STEP, GPIO.OUT)
GPIO.output(DIR, CW)

step_count = SPR
delay = .0208
for x in range(step_count):
    GPIO.output(STEP, GPIO.HIGH)
    sleep(delay)
    GPIO.output(STEP, GPIO.LOW)
    sleep(delay)

sleep(.5)
GPIO.output(DIR, CCW)

for x in range(step_count):
    GPIO.output(STEP, GPIO.HIGH)
    sleep(delay)
    GPIO.output(STEP, GPIO.LOW)
    sleep(delay)
GPIO.cleanup()
```



More Ref:

<https://medium.com/@Keithweaver /controlling-stepper-motors-using-python-with-a-raspberry-pi-b3fb482f886>

IOT-OPEN.EU Consortium, Introduction to the IoT, May 2019.

## Stepper Motor Control Example (2/2)

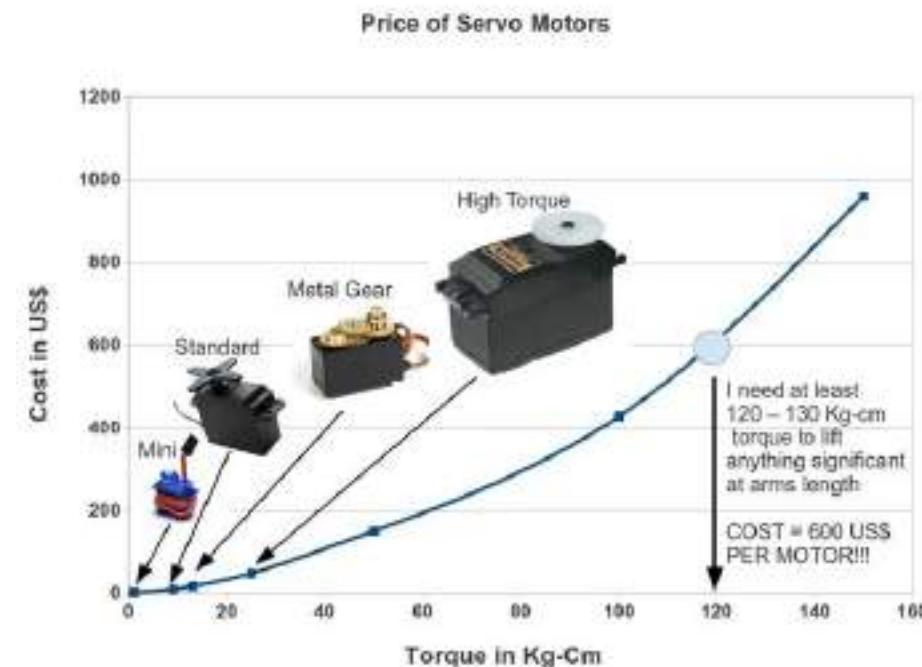
- 5V Step Motor
- Driver IC : TI ULN2003A
- Connections
  - Motor A+ connected to INA of ULN2003A, controlled by GPA3
  - Motor A- connected to INB of ULN2003A, controlled by GPA2
  - Motor B+ connected to INC of ULN2003A, controlled by GPA1
  - Motor B- connected to IND of ULN2003A, controlled by GPA0
- ```
unsigned char CCW[8]={0x08, 0x0c, 0x04, 0x06, 0x02, 0x03, 0x01, 0x09}; //counter-clockwise sequence
```
- ```
unsigned char CW[8] ={0x09, 0x01, 0x03, 0x02, 0x06, 0x04, 0x0c, 0x08}; //clockwise sequence
```





## Servo Motor (Analog Output)

- A servo-motor is a rotary actuator with a built-in feedback mechanism that responds to a control signal by moving to and holding a position, or by moving at a continuous speed.
- It typically consists of a small electric motor, driving a train of reduction gears, and controlled by a control circuit. Historically servos have been used in **radio control gadgets** and in small-scale robotics.

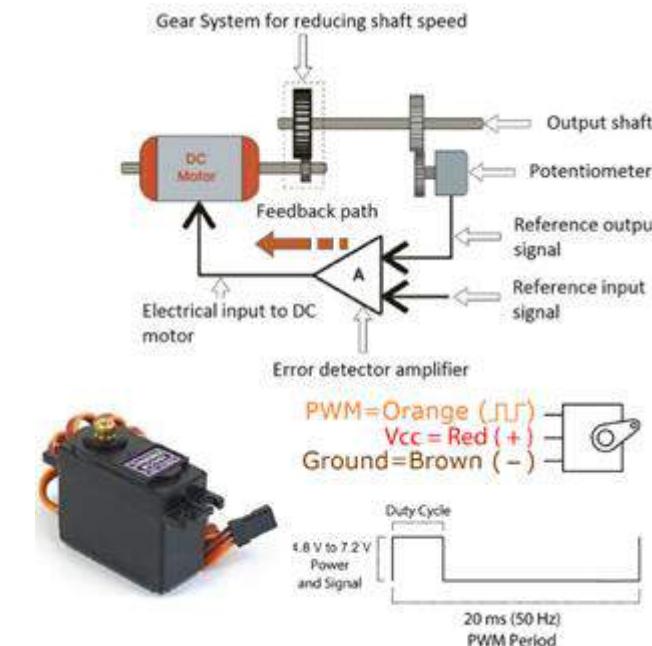


<https://www.youtube.com/watch?v=ditS0a28Sko>



## Principle of Servo Motor

- Servo motors are constructed out of basic DC motors, running in a close loop, by adding:
  - some gear reduction
  - a position sensor for the motor shaft
  - an electronic circuit that controls the motor's operation
- The basic hobby servo has a 180:1 gear ratio. The motor is typically small.
- Typically, a potentiometer (variable resistor) measures the position of the output shaft at all times so the controller can accurately place and maintain its setting.

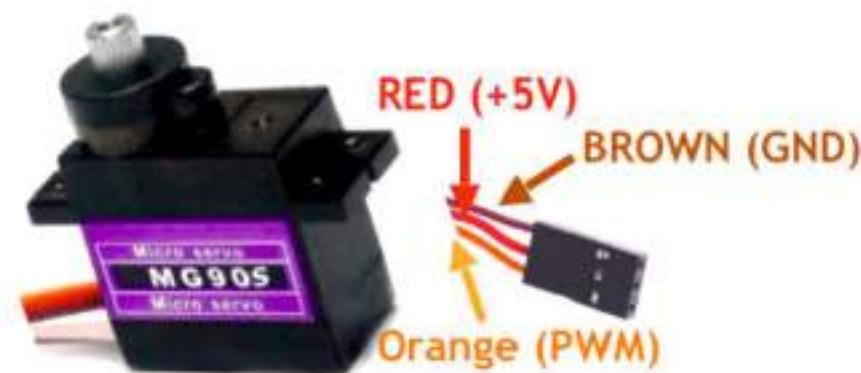
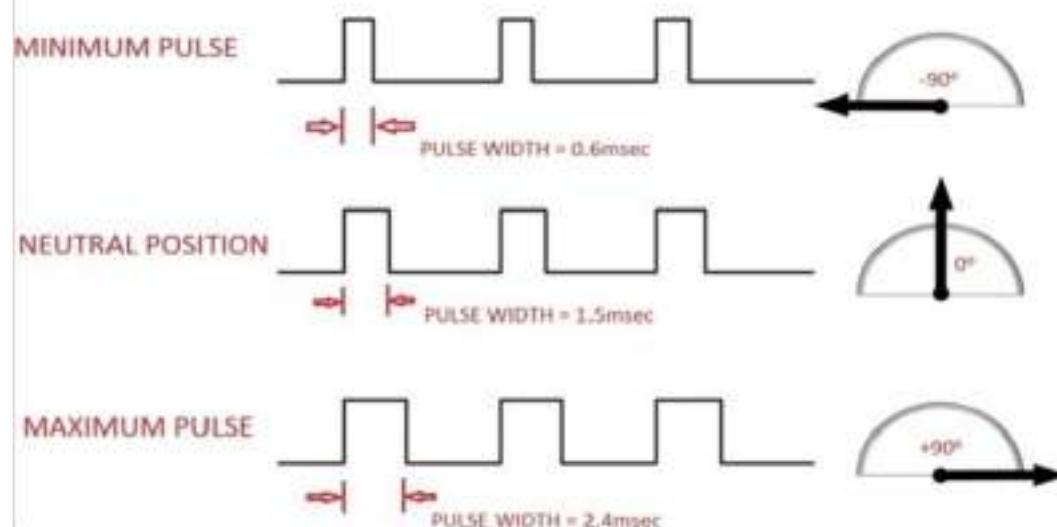




## Servo Motor Control

- An external controller (such as the ESP32) tells the servo where to go with a signal known as pulse proportional modulation (PPM) or pulse code modulation (which is often confused with pulse width modulation, PWM).
- PPM uses 1 to 2ms out of a 20ms time period to encode its information.
- A control wire communicates the desired angular movement. The angle is determined by the duration of the pulse applied to the control wire.
- The servo expects to see a pulse every 20 milliseconds (.02 seconds). The length of the pulse will determine how far the motor turns. A 1.5 millisecond pulse will make the motor turn to the 90 degree position (often called the neutral position).
- If the pulse is shorter than 1.5 ms, then the motor will turn the shaft closer to 0 degrees. If the pulse is longer than 1.5ms, the shaft turns closer to 180 degrees.

## Servo Motor PWM Signal





## Servo Motor (Analog Output)



REALPARS



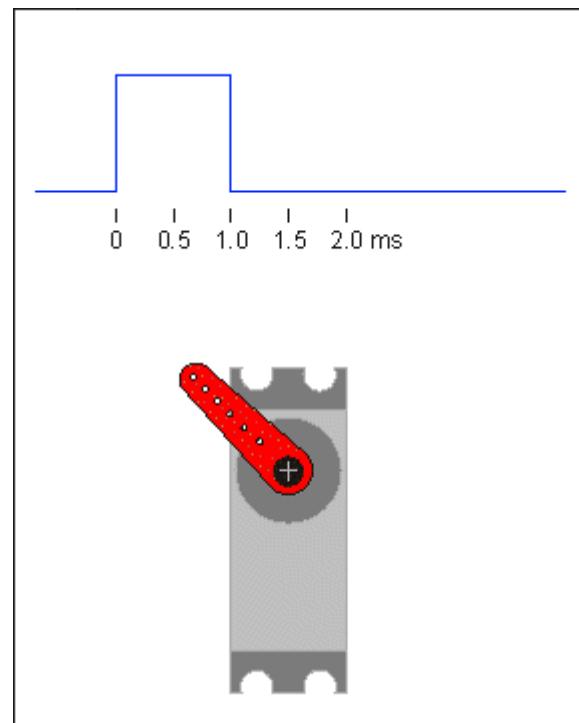
## Servo Motor Driving Method with PWM (Analog Output)

## Servo Motor (Analog Output)

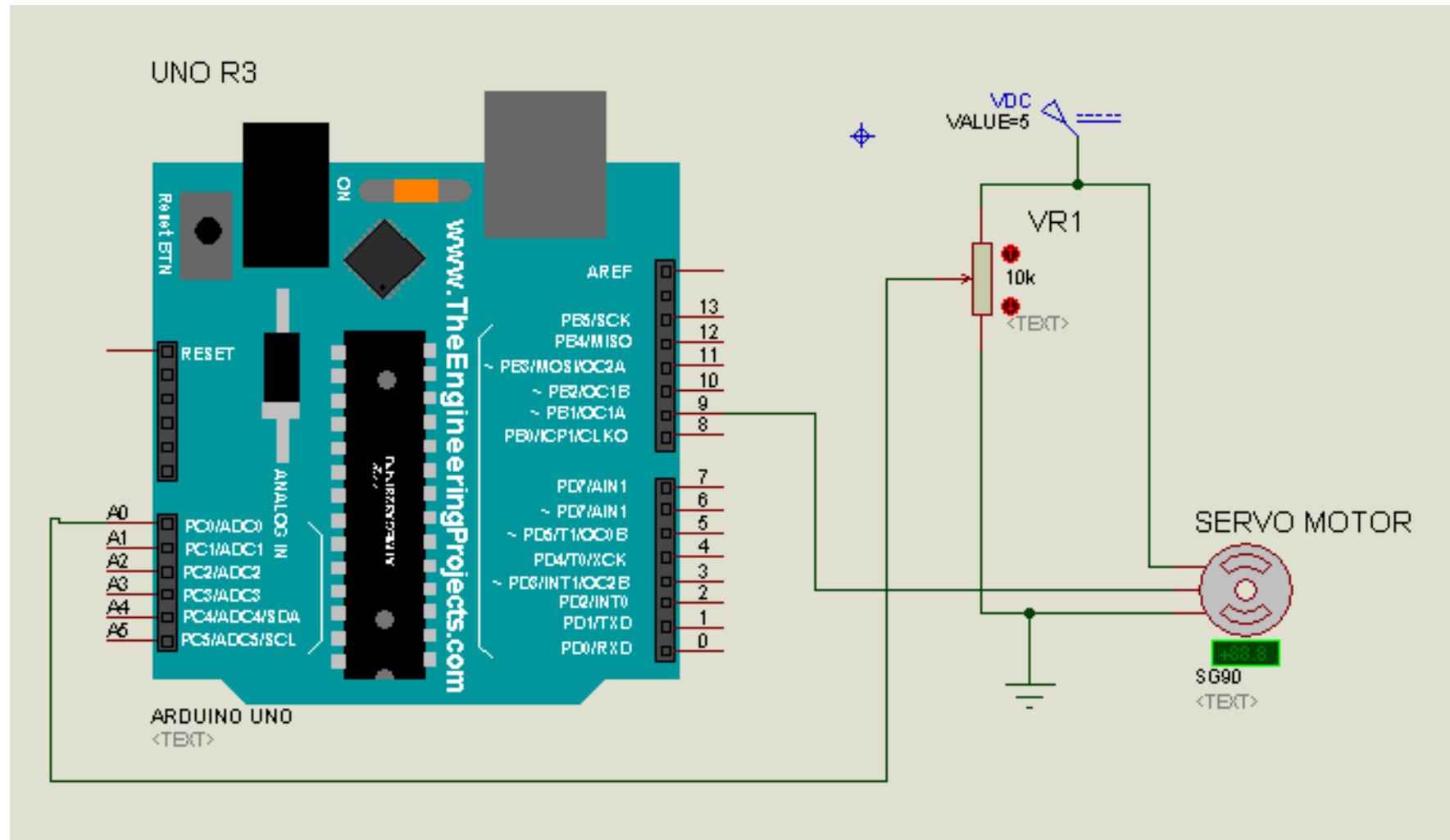


# Servo Motor

- Required analog output PWM
- Servo (SG90):  
[http://www.ee.ic.ac.uk/pcheung/teaching/DE1\\_EE/stores/sg90\\_datasheet.pdf](http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf)
  - Interface: orange = PWM, red = Vcc, brown = Gnd
- Driving Method
  - **PWM pulse duration is 20ms**
  - High width = 0.5 ~ 2.5ms to control motor rotation
- PWM controller generate 20ms pulse
- ADC read from VR =  $(V_{max} - V_{min})$  or POT to control PWM high pulse width
- ADC input variable resistance to control gripper open/close



## Servo Motor Example (1/2)



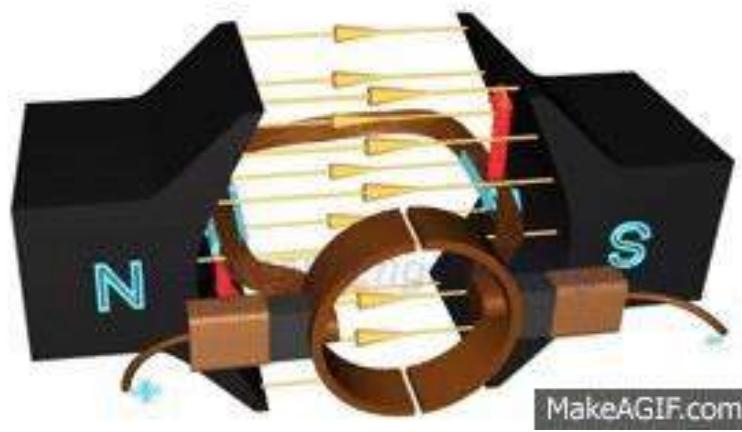


## Servo Motor Example (2/2)

```
#include <Servo.h>
int servoPin = 9;
Servo servo;
int servoAngle = 0; // servo position in degrees
int potpin = A0; // Pin to potentiometer
int val = 0;
void setup()
{ Serial.begin(9600);
  servo.attach(servoPin); //control the servo's speed
  //move the micro servo from 0 degrees to 180 degrees
  for(servoAngle = 0; servoAngle < 180; servoAngle++)
    { servo.write(servoAngle);
      delay(10);
    }
  delay(1000);
  //now move back the micro servo from 0 degrees to 180 degrees
  for(servoAngle = 180; servoAngle > 0; servoAngle--)
  { servo.write(servoAngle);
    delay(10);
  }
  //end control the servo's speed
}
void loop()
{
  //control the servo's direction and the position of the motor
  val = analogRead(potpin); // define analog read pin
  if(val <= 180 && val >= 0 )
  //now move back the micro servo from 0 degrees to 180 degrees
  {
    servo.write(val);
    Serial.println(val);
    delay(10);
  }
}
```

## DC Motor Driving Method (Digital or Analog Output to Driver)

- DC motor turns continuously in either direction depending on the polarity of the wires
- Need a L293D/DRV8833 H-Bridge driver chip to control the direction and speed of the turning according to voltage with PWM

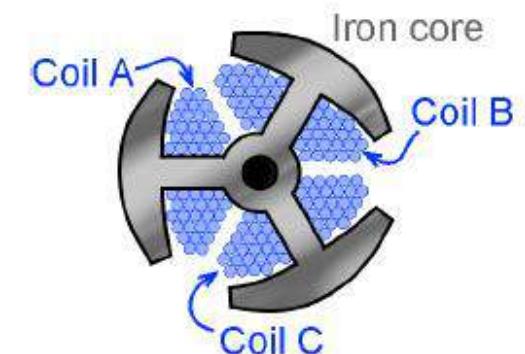
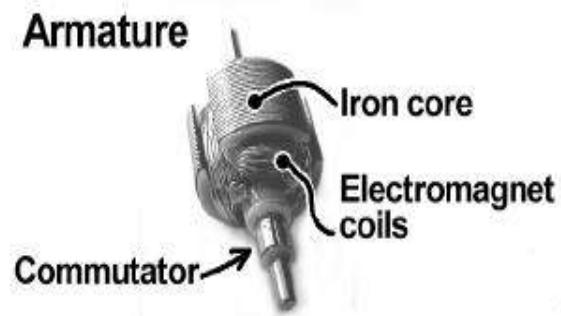


Note:

Using analog output means to apply PWM to the motor speed via the percentage of the duty cycle otherwise digital output is to on and off the motor with a single speed.

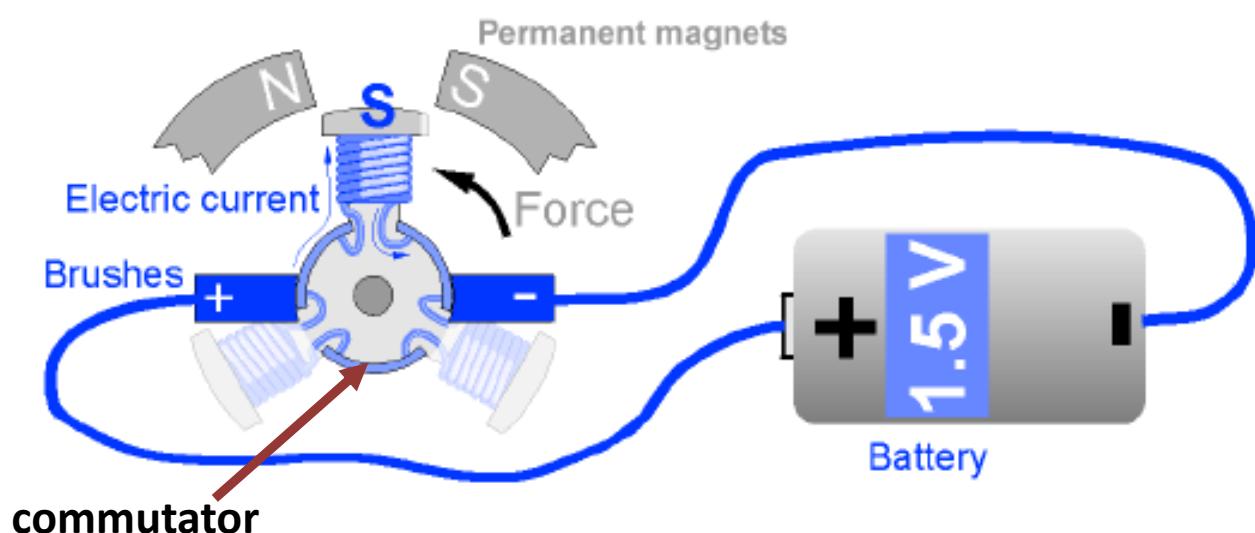
## The Principle of Simple DC Motor

- An electric motor uses electromagnets to convert electrical energy into mechanical energy.
- DC motors have **three** key components:
  - A **rotating element** (rotor) with magnets.
  - A **stationary magnet** that surrounds the rotor (stator).
  - A **commutator** that switches the electromagnets from north to south at the right place to keep the rotor spinning.
- The magnets in the rotor and stator could be **permanent magnets** or **electromagnets** (but at least one of them have to be electromagnets).
- The **rotating part of the motor, including the electromagnets**, is also called the **armature**.



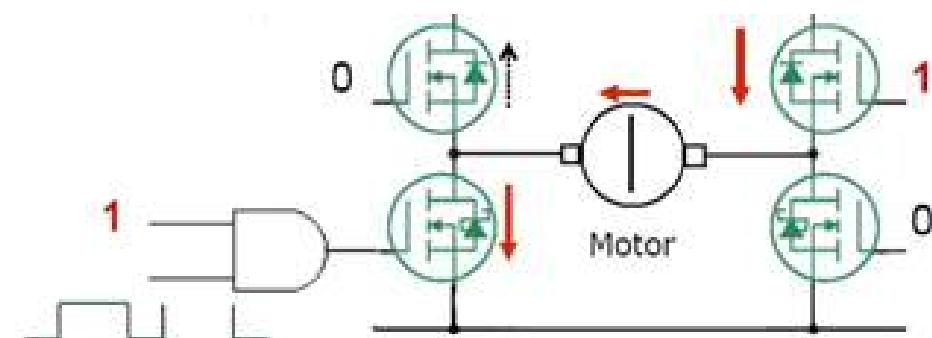
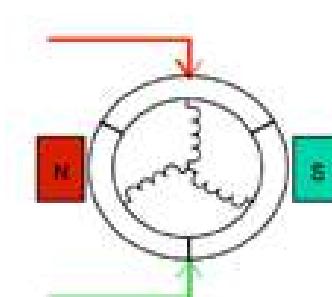
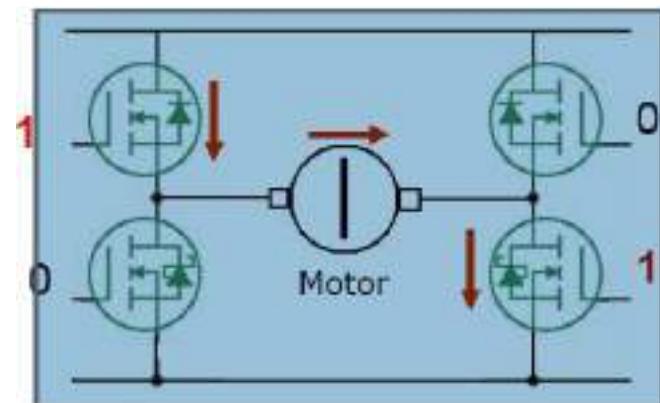
## The Principle of Brushed DC Motor

- The permanent magnets are on the outside, and they stay fixed in place.
- The wires from each of the three coils are attached to three metal plates (commutator) at the end of the armature.
- As the rotor spins, the three plates come into contact with the positive and negative brushes.
- Electric current flows through the brushes into the coils.



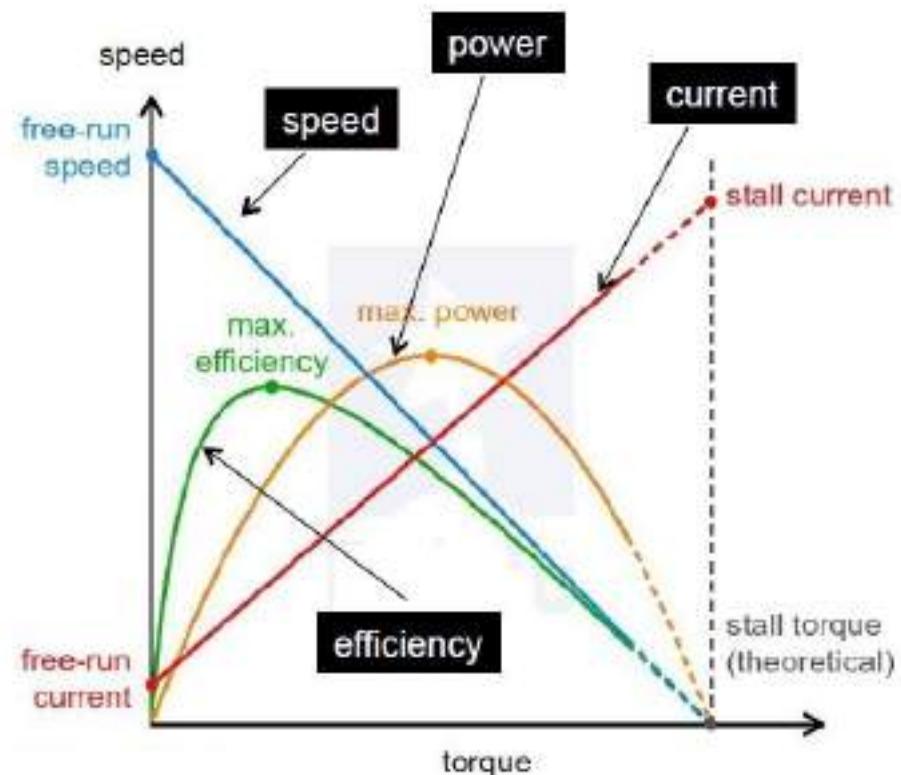
## The Principle of Brushed DC Motor

- The DC motor needs four transistors to operate.
- We used the DRV8833 chip to drive the motor with four transistors.
- The combination of transistors is called an H-Bridge, due to the obvious shape. (See diagram.)
- Transistors are switched diagonally to allow DC current to flow in the motor in either direction.
- The transistors can be Pulse Width Modulated to reduce the average voltage at the motor, useful for controlling current and speed.



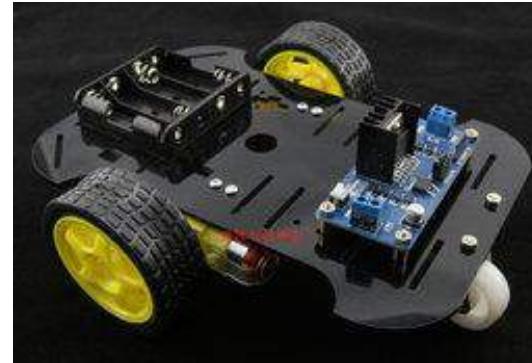
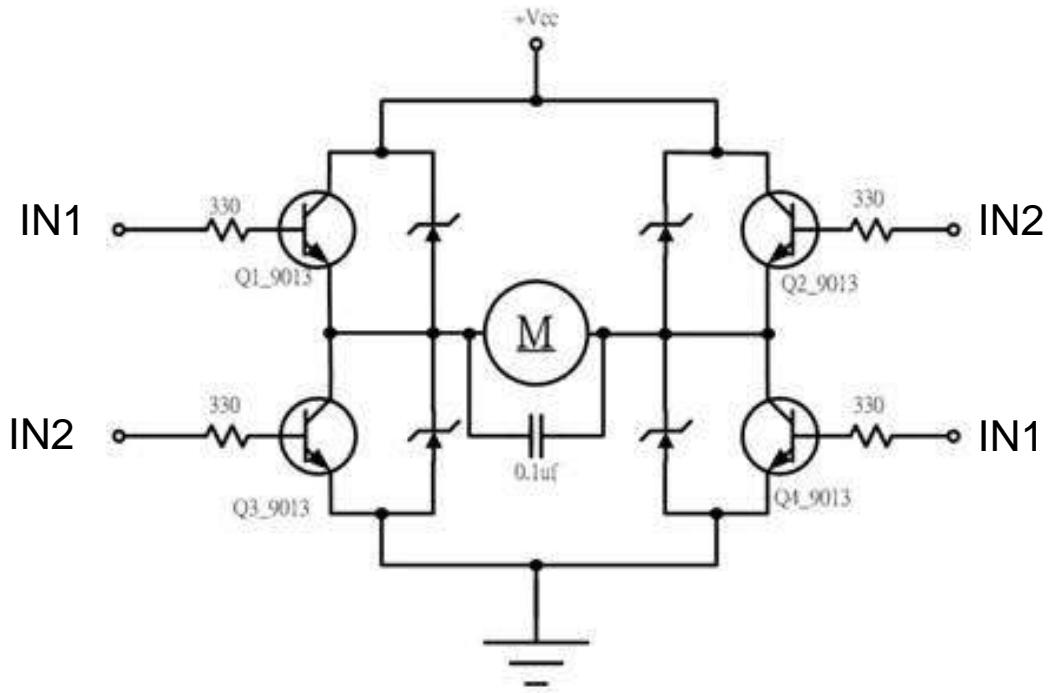
## Torque, Speed, Power, Efficiency

- Speed of a DC motor drops as torque increases.



## DC Motor Driving Method

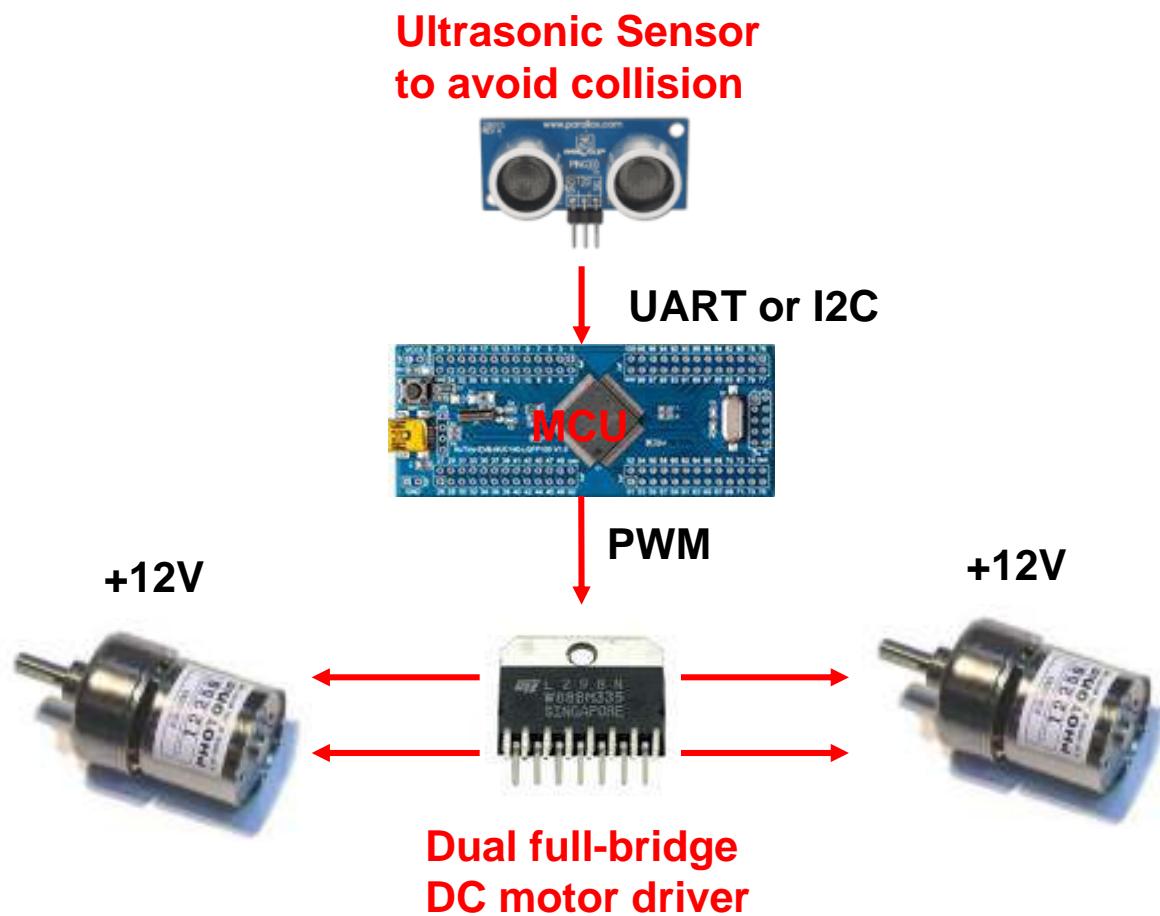
## DC Motor Full-Bridge Driver



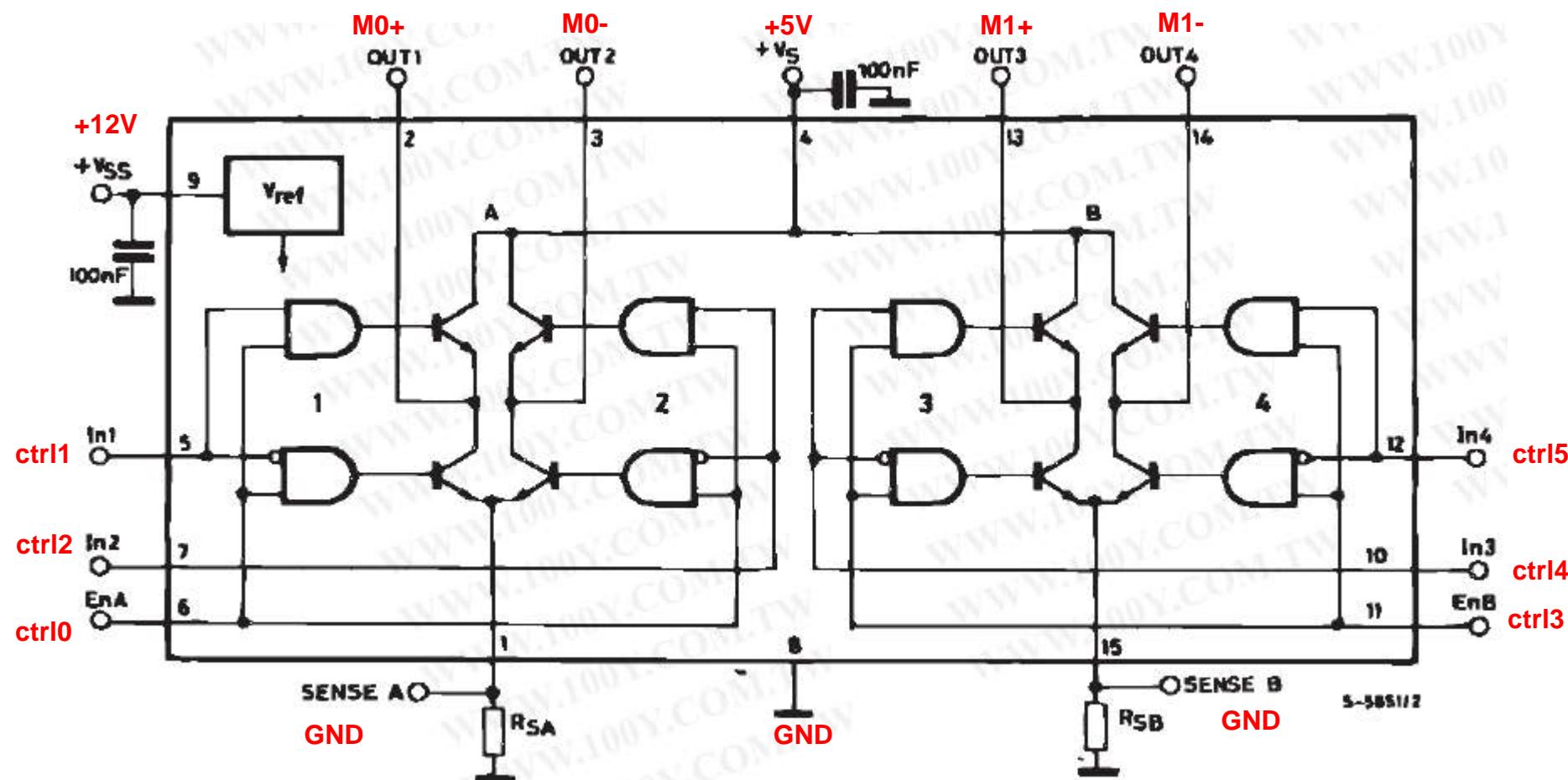
Direction A : IN1=1 && IN2=0

Direction B : IN1=0 && IN2=1

## DC Motor Control for Robot

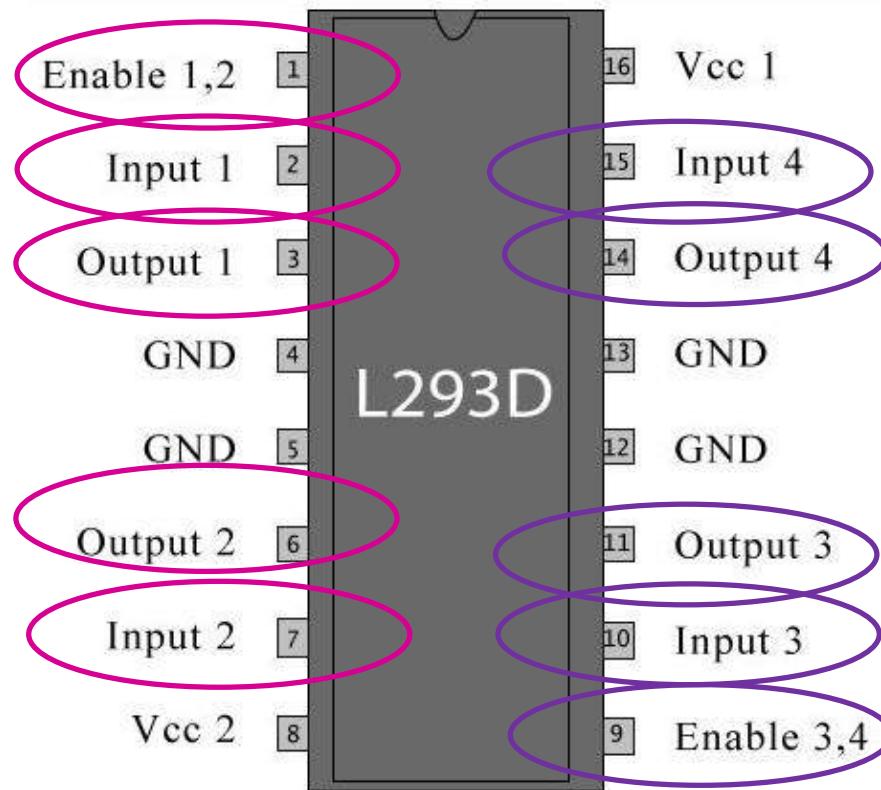


## Dual Full-Bridge DC Motor Driver: L293D



## Dual Full-Bridge DC Motor Driver: L293D

L293D H-Bridge IC



Note:

L293D is for small DC Motor most in range of 5 to 12V

L293N is for higher rating DC motors

1 – Enable pin for motor 1

2 & 7 – Control for motor 1

3 & 6 – Output to motor 1

9 – Enable pin for motor 2

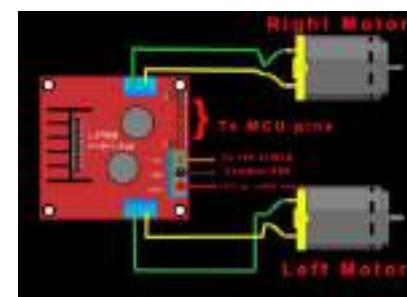
10 & 15 – Control for motor 2

11 & 14 – Output to motor 2

4, 5, 12 & 13 – Gnd

8 – +9-12V supply for motor

16 - +5V supply for IC



Ref:

<https://www.instructables.com/id/How-to-Use-L298n-to-Control-Dc-Motor-With-Arduino/>



# Dual Full-Bridge DC Motor Driver: L293D

## L293D Control Logic

**Table 2. Unidirectional DC Motor Control**

| EN | 3A | M1 <sup>(1)</sup>       | 4A | M2                      |
|----|----|-------------------------|----|-------------------------|
| H  | H  | Fast motor stop         | H  | Run                     |
| H  | L  | run                     | L  | Fast motor stop         |
| L  | X  | Free-running motor stop | X  | Free-running motor stop |

(1) L = low, H = high, X = don't care

| Enable | Input 1 | Input 2 | Motor1              |
|--------|---------|---------|---------------------|
| 0      | x       | x       | stop                |
| 1      | 0       | 0       | stop                |
| 1      | 0       | 1       | turn clockwise      |
| 1      | 1       | 0       | turn anti-clockwise |
| 1      | 1       | 1       | stop                |

| Enable | Input 3 | Input 4 | Motor2              |
|--------|---------|---------|---------------------|
| 0      | x       | x       | stop                |
| 1      | 0       | 0       | stop                |
| 1      | 0       | 1       | turn clockwise      |
| 1      | 1       | 0       | turn anti-clockwise |
| 1      | 1       | 1       | stop                |

## Dual Full-Bridge DC Motor Driver: L293D

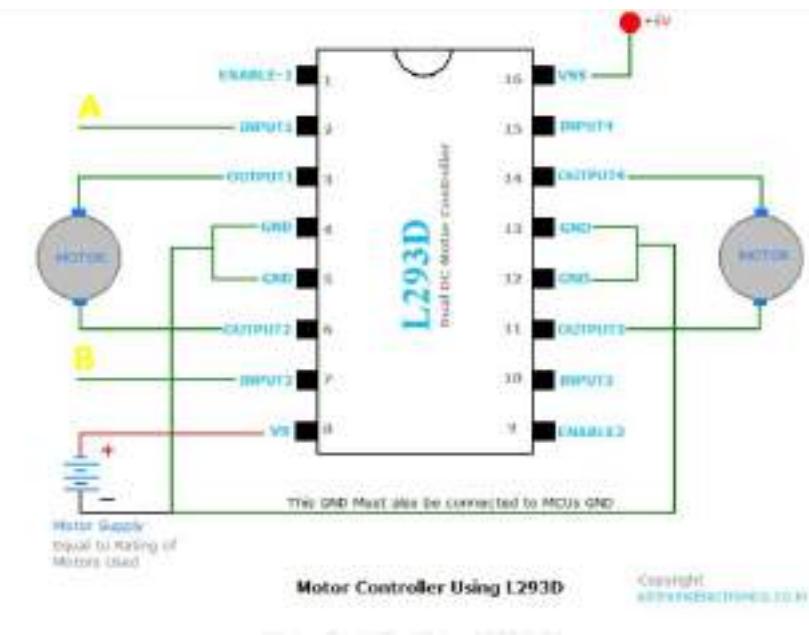
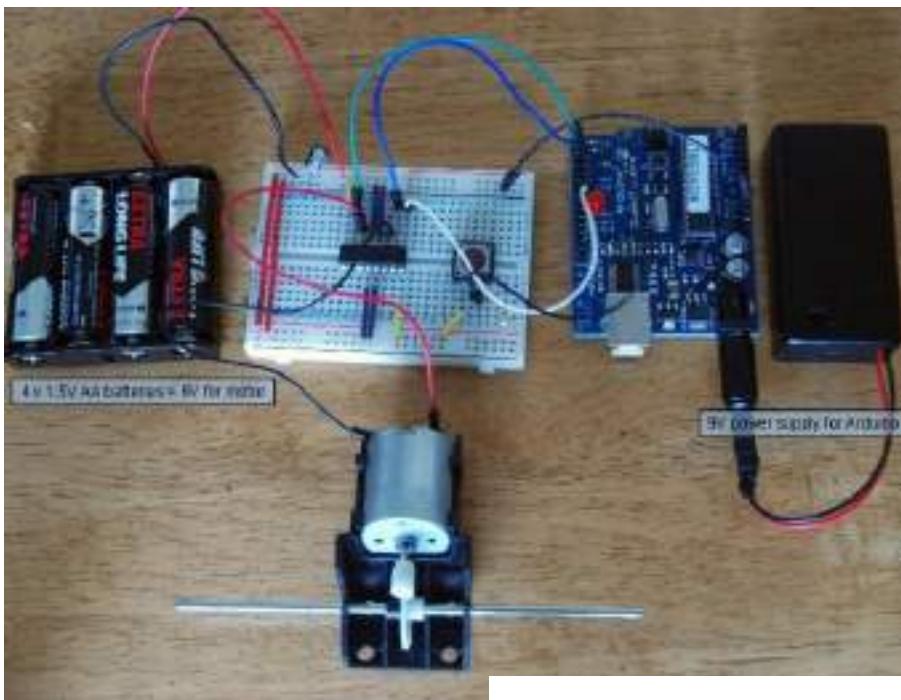


Table 2. Unidirectional DC Motor Control

| EN | 3A | M1 <sup>(1)</sup>       | 4A | M2                      |
|----|----|-------------------------|----|-------------------------|
| H  | H  | Fast motor stop         | H  | Run                     |
| H  | L  | run                     | L  | Fast motor stop         |
| L  | X  | Free-running motor stop | X  | Free-running motor stop |

(1) L = low, H = high, X = don't care

Ref: <http://luckylarry.co.uk/arduino-projects/control-a-dc-motor-with-arduino-and-l293d-chip/>



## DC Motor Solution (1/2)

```

int motor1Pin1 = 3;      // pin 2 on L293D (1A)
int motor1Pin2 = 4;      // pin 7 on L293D (2A)
int enablePin = 10;      // pin 1 on L293D (En)
float tempC;
int reading;
int tempPin = 0;
float temp_Ajd = 2.5; (to accelerate the result)
int RGBPin[] = {5, 6, 9}; // G B R
void setup() {
    Serial.begin(9600);
    // set all the other pins you're using as outputs:
    pinMode(motor1Pin1, OUTPUT);
    pinMode(motor1Pin2, OUTPUT);
    pinMode(enablePin, OUTPUT); // set enablePin high so that motor can turn on:
    digitalWrite(enablePin, LOW); // Read Temp
    analogReference(INTERNAL);
    //Setup Pin RGB
    pinMode(RGBPin[0], OUTPUT);
    pinMode(RGBPin[1], OUTPUT);
    pinMode(RGBPin[2], OUTPUT);
}

```

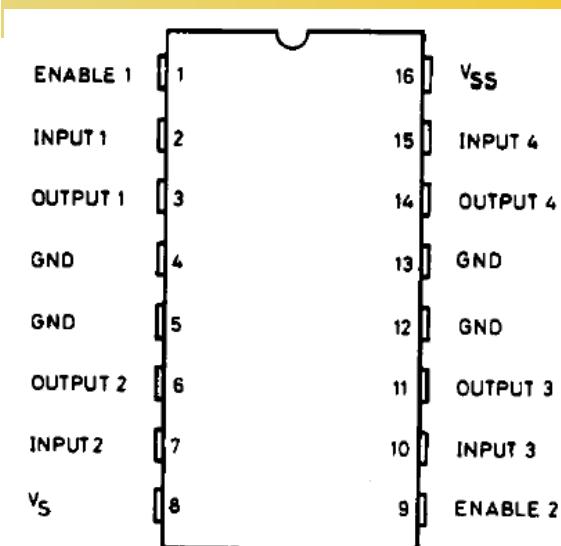
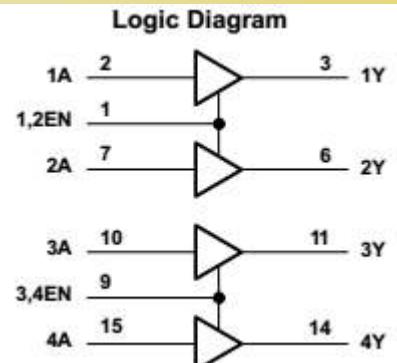


Table 2. Unidirectional DC Motor Control

| EN | 3A | M1 <sup>(1)</sup>       | 4A | M2                      |
|----|----|-------------------------|----|-------------------------|
| H  | H  | Fast motor stop         | H  | Run                     |
| H  | L  | run                     | L  | Fast motor stop         |
| L  | X  | Free-running motor stop | X  | Free-running motor stop |

(1) L = low, H = high, X = don't care



## DC Motor Solution (2/2)

```
void SetRGB(int r, int g, int b) {
    analogWrite(RGBPin[0], r);
    analogWrite(RGBPin[1], g);
    analogWrite(RGBPin[2], b);
}
void loop() {
    reading = analogRead(tempPin);
    tempC = reading / 9.31;
    tempC = tempC-temp_Ajd;
    Serial.print(tempC);
    Serial.println(" degrees Celsius ");
```

Table 2. Unidirectional DC Motor Control

| EN | 3A | M1 <sup>(1)</sup>       | 4A | M2                      |
|----|----|-------------------------|----|-------------------------|
| H  | H  | Fast motor stop         | H  | Run                     |
| H  | L  | run                     | L  | Fast motor stop         |
| L  | X  | Free-running motor stop | X  | Free-running motor stop |

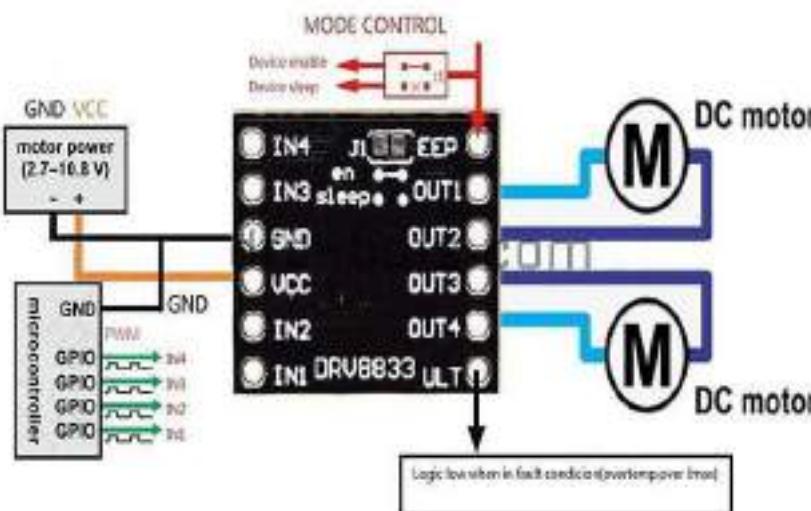
(1) L = low, H = high, X = don't care

```
if(tempC >30) //32
{
    //Power On at enablePin is to switch on Motor
    digitalWrite(enablePin, HIGH);
    digitalWrite(motor1Pin1, HIGH); // set pin 2 on L293D HIGH
    digitalWrite(motor1Pin2, LOW); // set pin 7 on L293D LOW
    Serial.println("HIGH Temp: LED RED");
    SetRGB(0, 255, 255); // Red // RGB Common Anode
    delay(500);
}
else if(tempC >=25 && tempC < 32 ){
    digitalWrite(enablePin, LOW);
    digitalWrite(motor1Pin1, LOW); // set pin 2 on L293D LOW
    digitalWrite(motor1Pin2, LOW); // set pin 7 on L293D LOW
    Serial.println("MEDIUM Temp: LED GREEN");
    SetRGB(255, 0, 255); // Green
    delay(500);
}
else if(tempC < 25){
    digitalWrite(enablePin, LOW);
    digitalWrite(motor1Pin1, LOW); // set pin 2 on L293D LOW
    digitalWrite(motor1Pin2, LOW); // set pin 7 on L293D LOW
    Serial.println("LOW Temp: LED BLUE");
    SetRGB(255, 255, 0); // Blue
    delay(500);
}
else {SetRGB(255, 255, 255); } // OFF RGB
}// End Loop
```

## DRV8833

# DRV8833

## 2 Channel DC Motor Driver 1.5A/3-10V Module



## 7.3.2 Bridge Control and Decay Modes

TheAIN1 andAIN2 input pins control the state of theAOUT1 andAOUT2 outputs; similarly, theBIN1 andBIN2 input pins control the state of theBOUT1 andBOUT2 outputs. Table 1 shows the logic.

Table 1. H-Bridge Logic

| xIN1 | xIN2 | xOUT1 | xOUT2 | FUNCTION         |
|------|------|-------|-------|------------------|
| 0    | 0    | Z     | Z     | Coast/soft decay |
| 0    | 1    | L     | H     | Reverse          |
| 1    | 0    | H     | L     | Forward          |
| 1    | 1    | L     | L     | Brake/slow decay |

The inputs can also be used for PWM control of the motor speed. When controlling a winding with PWM, when the drive current is interrupted, the inductive nature of the motor requires that the current must continue to flow. This is called recirculation current. To handle this recirculation current, the H-bridge can operate in two different states: fast decay or slow decay. In fast decay mode, the H-bridge is disabled and recirculation current flows through the body diodes; in slow decay, the motor winding is shorted.

To PWM using fast decay, the PWM signal is applied to one xIN pin while the other is held low; to use slow decay, one xIN pin is held high.

Table 2. PWM Control of Motor Speed

| xIN1 | xIN2 | FUNCTION                |
|------|------|-------------------------|
| PWM  | 0    | Forward PWM, fast decay |
| 1    | PWM  | Forward PWM, slow decay |
| 0    | PWM  | Reverse PWM, fast decay |
| PWM  | 1    | Reverse PWM, slow decay |



## DC/Servo/Stepper Motors

- **DC Motors:**

Fast, continuous rotation. Finally they are used for anything that needs to spin at a high RPM e.g. car wheels, fans, drills etc.

- **Servo Motors:**

May be very fast, high torque, very accurate rotation within a limited angle. Generally a high performance alternative to stepper motors, but more complicated setup with PWM tuning.

Suited for robotic arms/legs etc. Servos require a feedback mechanism and support circuitry to drive positioning.

- **Stepper Motors:**

Quite slow, precise rotation, easy set up and control. Advantage over servo motors in positional control where turn angle is not limiter.

Stepper motors are suited for 3D printers and similar devices where position is fundamental.

- **Stepper vs Servo: The Verdict**

Servo control systems best suit to high speed, high torque applications that involve dynamic load changes.

Stepper control systems are less expensive and are optimal for applications that require low-to-medium acceleration, high holding torque, and the flexibility of open or closed loop operation.

Ref: <https://totemmaker.net/blog/electric-motor-comparison-stepper-vs-servo-vs-dc/>



# DC Motor vs Servo Motor

## DC Motor

- Motion is continuous
- Speed controlled by applied voltage (PWM)
- Using digital or analog output via DC motor driver
- Examples: cheap toys and automotive applications such as Cranes, Air compressor, Lifts, Elevators, Winching system, Electric traction, Hair drier, Vacuum cleaner and in speed regulation application, power tools, Sewing machine, Electric footing

## Servo Motor

- Capable of holding a position
- Speed controlled by delay between position updates
- Position (or degree) controlled with feedback (stepper motor without feedback)
  - Using analog output via PWM
  - Examples: Hybrid of motor, gears and controller.

## Wearable Sensors & Smartwatch

- G. Arognam, N. Manivannan, and D. Harrison, “Review on Wearable Technology Sensors Used in Consumer Sport Applications”, April 2019
- ECG (Electrocardiography) and PPG (PhotoPlethysmoGraphy) for heart rate measurement:  
<https://www.youtube.com/watch?v=rM3a4xL3d50>  
<https://www.youtube.com/watch?v=Y9f-RpwXr4E>

# Electrocardiography and PhotoPlethysmoGraphy



# Photoplethysmography





# **ITCS447**

## **Lecture 7**

## **Busses**

### **Serial & Parallel Communications**

### **SPI & I2C & 1-Wire & I2S**

**Asst. Prof. Dr. Thitinan Tantidham**



มหาวิทยาลัยมหิดล  
Mahidol University

ห้ามมิให้นักศึกษาทำข้า ดัดแปลง หรือใช้ประโยชน์จากการอันมีลิขสิทธิ์ปราภูอยู่ในระบบการศึกษาอิเล็กทรอนิกส์ (E-Learning) ของมหาวิทยาลัย ไม่ว่าจะทั้งหมดหรือบางส่วน โดยไม่ได้รับอนุญาตจากมหาวิทยาลัย นอกจากนี้จากการศึกษาส่วนบุคคล ทั้งนี้ การทำข้า ดัดแปลง หรือเผยแพร่ต่อสาธารณะชื่องานอันมีลิขสิทธิ์ จะมีโทษปรับตั้งแต่ 20,000 บาท ถึง 200,000 บาท และหากเป็นการกระทำเพื่อการค้า จะมีโทษจำคุกตั้งแต่ 6 เดือน ถึง 10 ปี หรือปรับตั้งแต่ 100,000 บาท ถึง 800,000 บาท หรือทั้งจำทั้งปรับ

Unless allowed by Mahidol University, the User shall not copy, modify, or exploit in part or in whole of the copyrighted materials on the Platform, other than for your own individual study. The copy, modify or communication to public of the copyrighted materials shall be inflicted with a fine from 20,000 Baht up to 200,000 Baht. If the offence is committed with the commercial purpose, the offender shall be inflicted with imprisonment for a term from 6 months up to 10 years or a fine from 100,000 Baht up to 800,000 Baht or both imprisonment and fine.



# Outline



- Arduino Uno vs ESP32 vs ESP8266
- Basic Communications: Serial and Parallel
- Synchronous Protocols: SPI & I2C
- 1-Wire
- I2S

# Arduino Uno R3



- Power
    - VIN, 5V, 3V3, GND
  - Control
    - RESET
  - Analog In
    - A0, A1, A2, A3, A4, A5
  - Digital I/O
    - 0,1,2,3,4,5,6,7,8,9,10,11,12,13
  - Reference for Shields
    - AREF, IOREF
  - Programmable LED
    - 13
- 3.3V
- Serial
    - 0 (RX), 1 (TX)
  - External Interrupts
    - 2, 3
  - PWM
    - 3,5,6,9,10,11

(Analog output)
  - SPI
    - 10(SS), 11(MOSI), 12(MISO), 13(SCK)
  - TWI (I2C)
    - A4 (SDA), A5 (SCL)

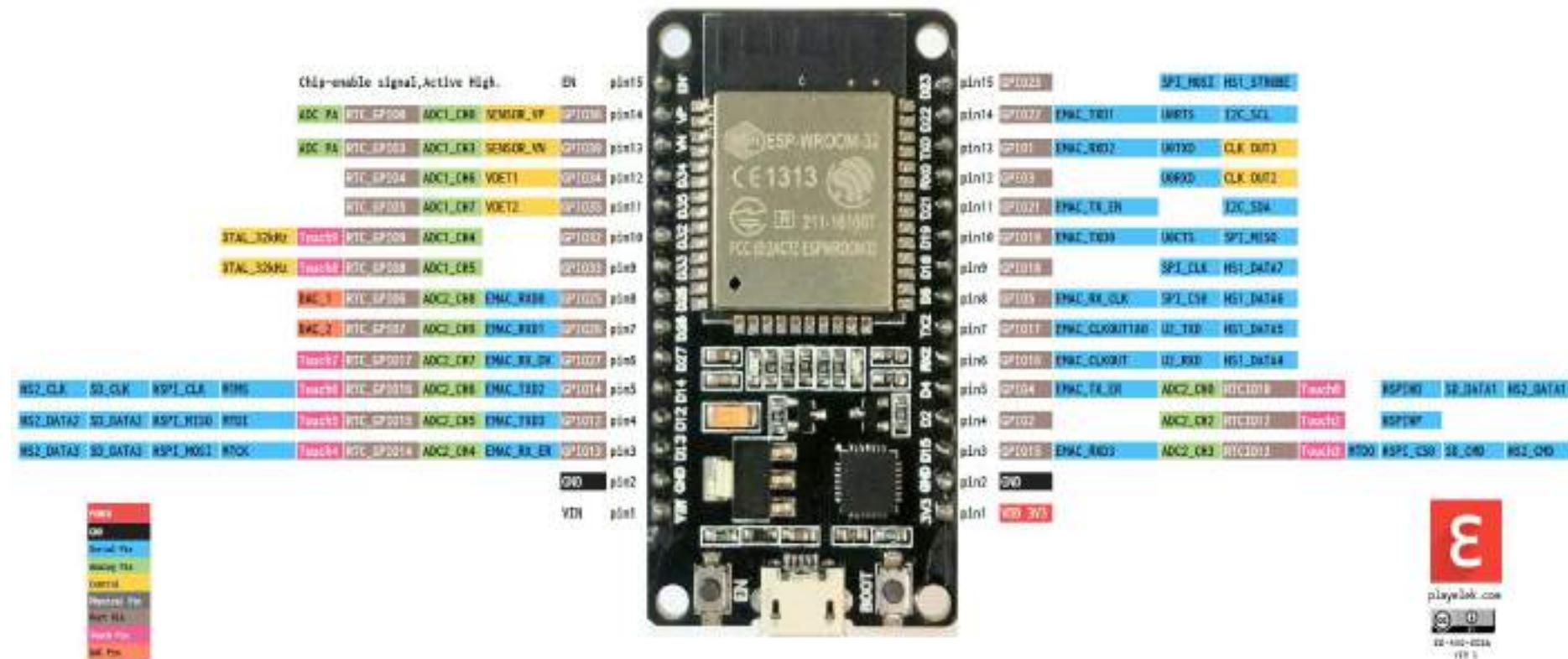
[http://www.adafruit.com/index.php?main\\_page=popup\\_image&pID=50](http://www.adafruit.com/index.php?main_page=popup_image&pID=50)

See: <http://learn.adafruit.com/arduino-tips-tricks-and-techniques/arduino-uno-faq>

## DOIT ESP32 DEVKIT V1 PINOUT

DOIT Esp32 DevKit V1 Documentation

UART(RX/TX): U0(12,13), U1(9,10), U2(16,17)  
 I2C(SDA/SCL): **21/22,13/12,11/10**



SPID=MOSI  
 SPIQ=MISO

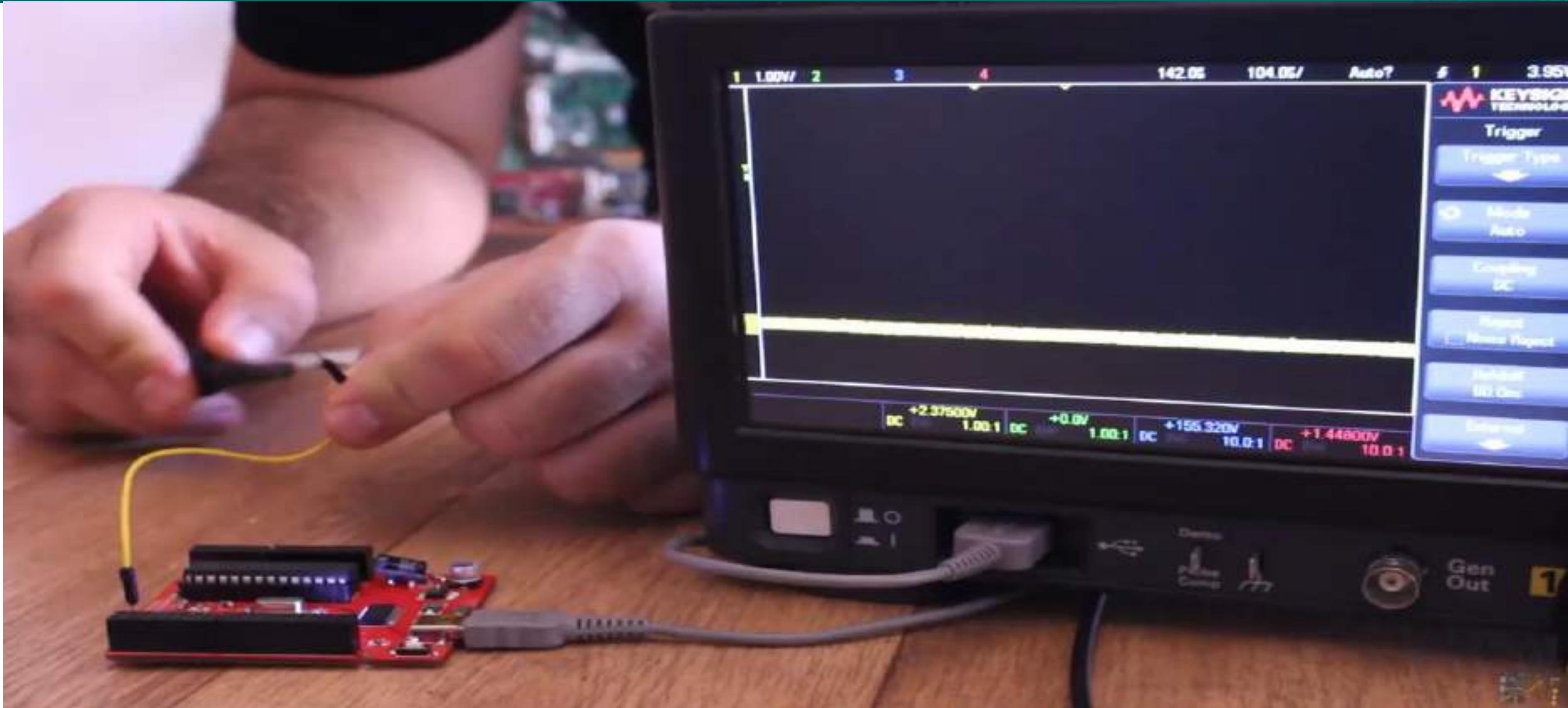
| SPI         | MOSI    | MISO    | CLK     | CS      |
|-------------|---------|---------|---------|---------|
| <b>VSPI</b> | GPIO 23 | GPIO 19 | GPIO 18 | GPIO 5  |
| <b>HSPI</b> | GPIO 13 | GPIO 12 | GPIO 14 | GPIO 15 |

<https://github.com/playelek/pinout-doit-32devkitv1>

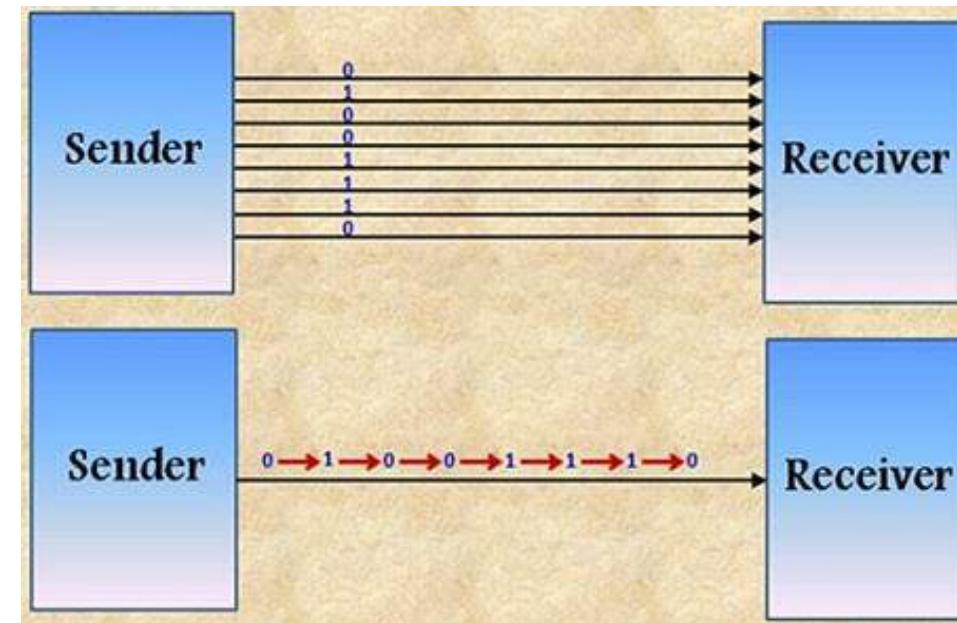


|                               | ESP8266                        | ESP32                                      |
|-------------------------------|--------------------------------|--------------------------------------------|
| <b>MCU</b>                    | Xtensa Single-core 32-bit L106 | Xtensa Dual-Core 32-bit LX6 with 600 DMIPS |
| <b>802.11 b/g/n Wi-Fi</b>     | HT20                           | HT40                                       |
| <b>Bluetooth</b>              | No                             | Bluetooth 4.2 and BLE                      |
| <b>Typical Frequency</b>      | 80 MHz                         | 160 MHz                                    |
| <b>SRAM</b>                   | No                             | Yes                                        |
| <b>Flash</b>                  | No                             | Yes                                        |
| <b>GPIO</b>                   | 17                             | 36                                         |
| <b>Hardware /Software PWM</b> | None / 8 channels              | None / 16 channels                         |
| <b>SPI/I2C/I2S/UART</b>       | 2/1/2/2                        | 4/2/2/2                                    |
| <b>ADC</b>                    | 10-bit                         | 12-bit                                     |
| <b>CAN</b>                    | No                             | Yes                                        |
| <b>Ethernet MAC Interface</b> | No                             | Yes                                        |
| <b>Touch Sensor</b>           | No                             | Yes                                        |
| <b>Temperature Sensor</b>     | No                             | Yes                                        |
| <b>Hall effect sensor</b>     | No                             | Yes                                        |
| <b>Working Temperature</b>    | -40°C to 125°C                 | -40°C to 125°C                             |

# Basic Communications



## Serial and Parallel Communications

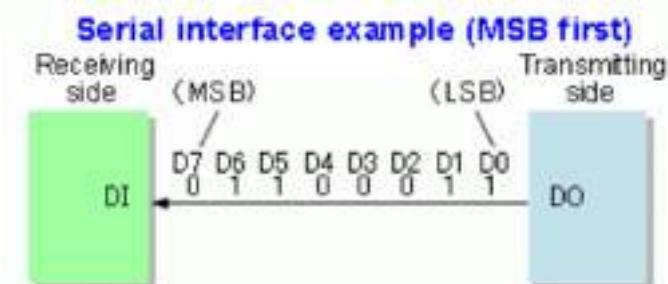
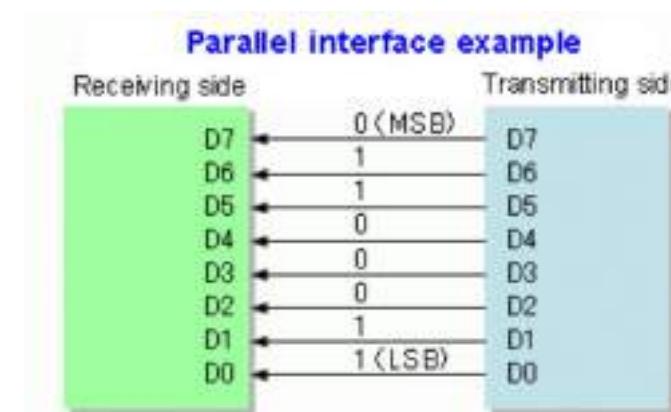


- Parallel interfaces have higher transmission rates but requires more lines.
- **Serial communications saves transmission lines.**



## Serial and Parallel Communications

| Parallel                                                                       | Serial                                                                                   |
|--------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|
| Conveying multiple binary digits (bits) simultaneously                         | Conveys only a single bit at a time                                                      |
| Many wires/traces<br>(number of electrical conductors)<br>Typically 1 wire/bit | Less wires/traces<br>Communication is based on rising and falling edges of a square wave |
| Examples: ISA, ATA, SCSI, PCI                                                  | Examples: UART, I2C, SPI                                                                 |





## Parallel Communications

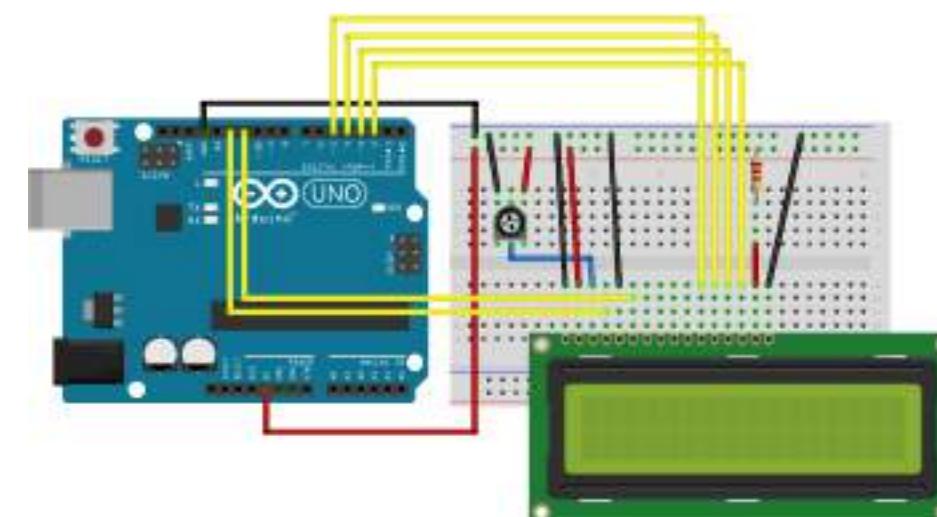
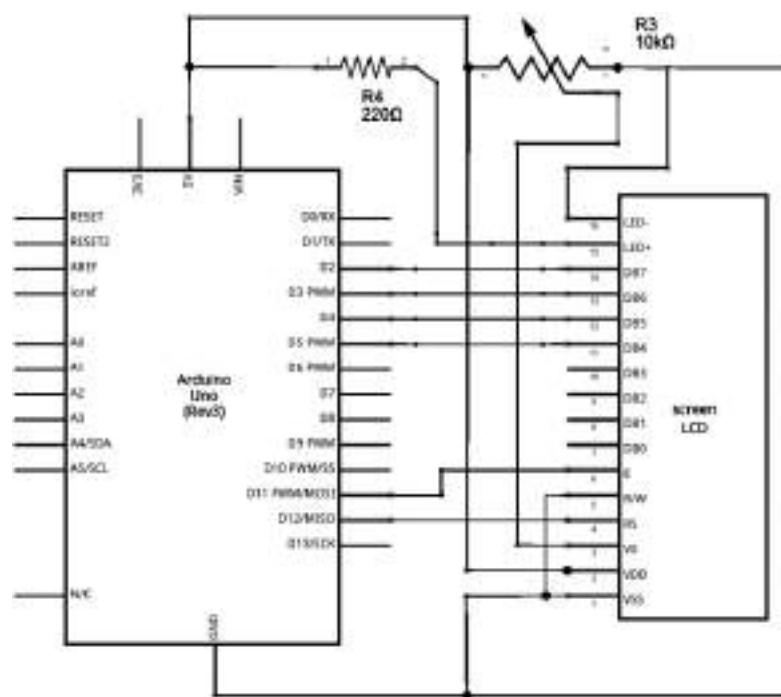
- Requires more traces and thus more space on the PCB
- Can be faster than serial communication
- Data transfer rate can be increased by increasing the number of connections
- Used in part to part communication in computers RAM <-> CPU <-> GPU



# Basic Communications

## Parallel LCD (4-bit data bus)

[This example can also be found in the Arduino IDE examples](https://www.arduino.cc/en/Tutorial>HelloWorld</a></p></div><div data-bbox=)





## Serial Communications

- Point-to-point communication between two devices
- Requires two wires minimum (TX and RX).
- Also recommended to connect the ground wires of the devices together.

## Modes of Connection

**One-way:** data sent in one direction only.

**Half-duplex:** two-way transfer, but only in one direction at a time.

**Full-duplex:** send and receive data simultaneously.

RS-232 and SPI is full-duplex.

I2C is half-duplex.

## Serial Communications

### Synchronous vs. Asynchronous

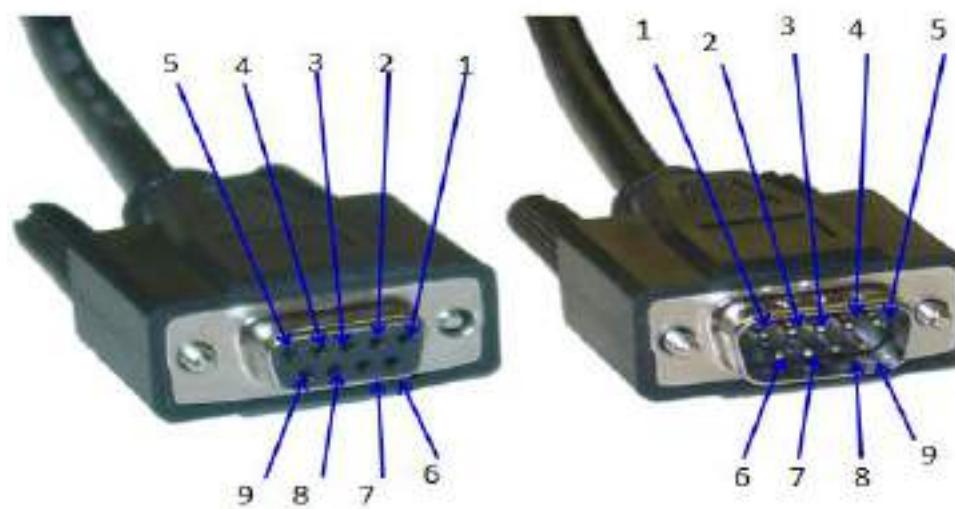
- **Asynchronous** has no clock, but speed must be agreed upon beforehand (baud rate)
- **Synchronous communication**, both the transmitter and receiver share a common clock.
- Thus, one line is assigned to the clock.

RS-232 (and also USB) is asynchronous

SPI and I2C are synchronous.

## Asynchronous: The RS-232 Interface

- RS-232: Recommended Standard 232
- Also called: “serial port” or COM port.
- Common interface to connect computer to other devices.
- Note: most modern laptops do not have serial port!



**The DB-9 com port has 9 pins numbered as shown.**

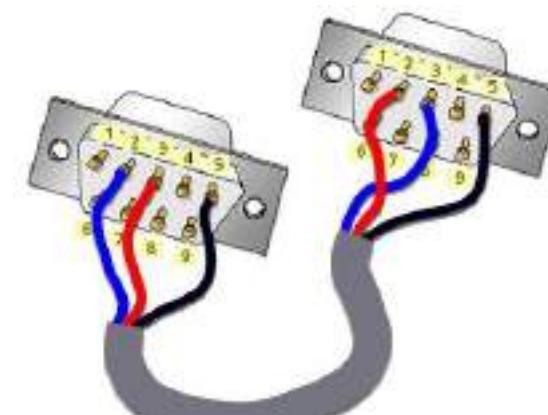
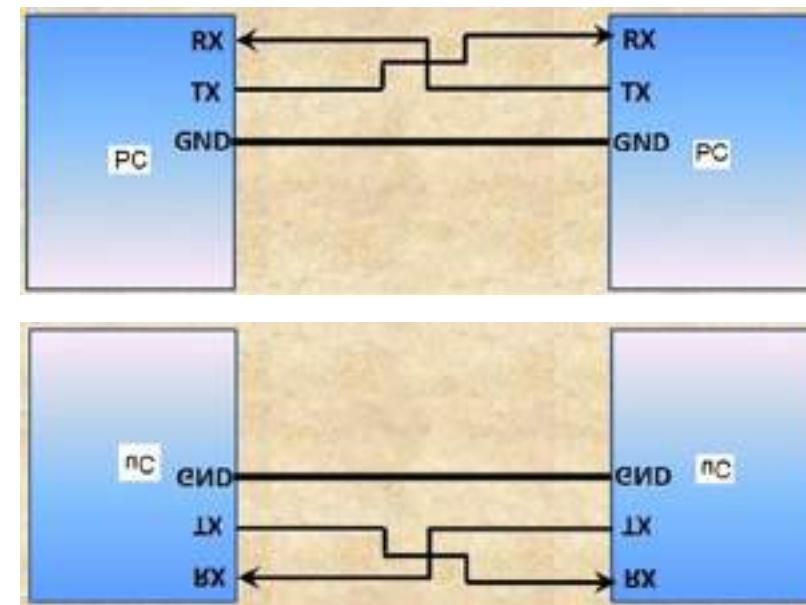
- Pin 2 – Tx (transmit data)
- Pin 3 – Rx (receive data)
- Pin 5 – Ground

The basic lines we need here



## Connecting two PCs or two Microcontrollers (Null Modem connection)

- To connect **two PCs**, the simplest way is to just connect
  - Pin 2 & Pin 3
  - Pin 3 & Pin 2
  - Pin 5 & Pin5
- This can be done with a cross cable.
- After that we should use a suitable software to implement the serial communications.





## What about connecting PC to microcontroller?

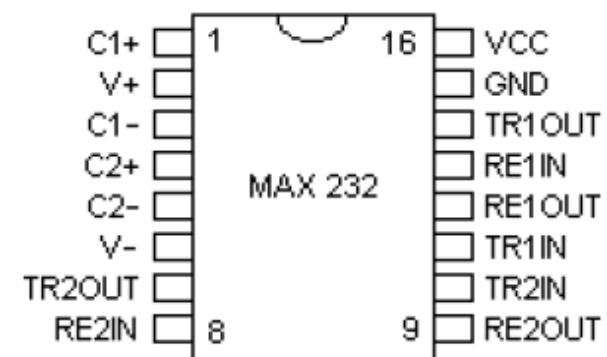
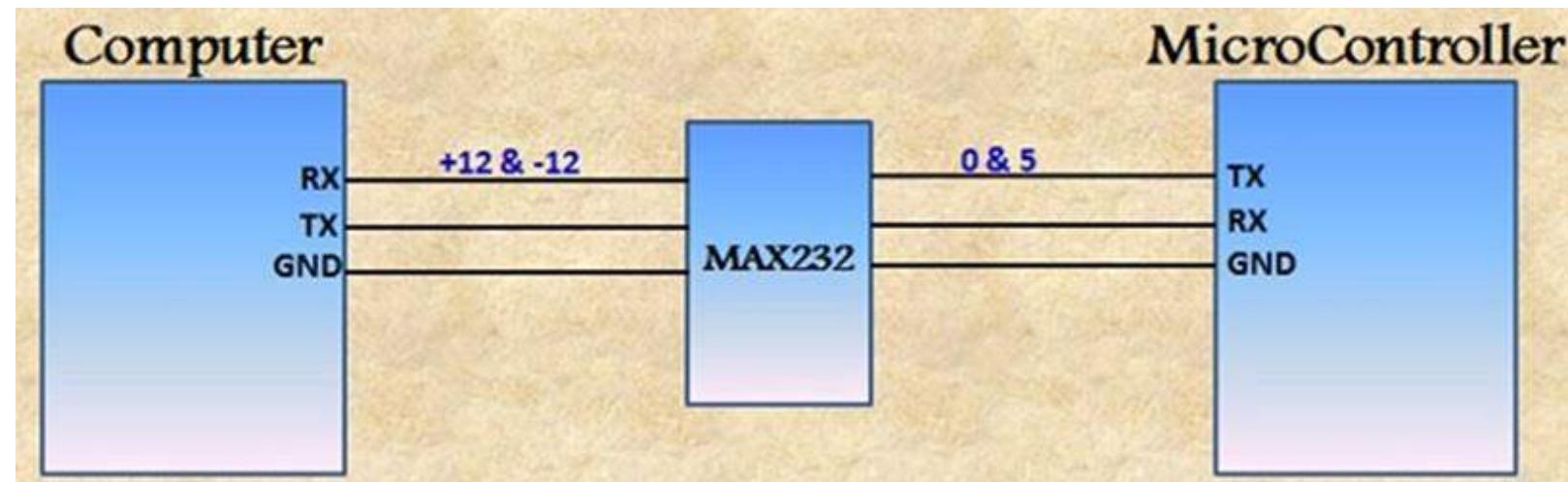
- This case is more difficult because **The RS 232 logic levels are different from the (TTL 0&5V) levels.**

| Logic Level | COM Port ( RS232 ) | UART          |
|-------------|--------------------|---------------|
| On State    | -3 To -25 Volt     | Around 5 Volt |
| Off State   | +3 To +25 Volt     | Around 0 volt |

- The region between +3 and -3 volts is undefined.
- The serial port in the microcontroller uses an IC called the **UART which uses the logic levels (0 and 5V) while the com port uses different levels.** So, we need to use **a converter**, e.g. MAX 232 IC.



## What about connecting PC to microcontroller?





## USB-Serial Adapter Adapter

```
#define RXD2 16
#define TXD2 17
char c;
String readString;
void setup() {
    // Note the format for setting a serial port is as
    follows: Serial2.begin(baud-rate, protocol, RX pin, TX pin);
    Serial.begin(115200);
    Serial2.begin(9600, SERIAL_8N1, RXD2, TxD2);
    Serial2.println("serial2test");
    Serial.println("Serial Txd is on pin: " + String(TX));
    Serial.println("Serial Rxd is on pin: " + String(RX));
}

void loop() {
    while (Serial2.available()) {
        c = Serial2.read();
        readString += c;
    }
    if (readString.length() > 0) {
        Serial.print(readString);
        Serial2.print(readString);
        //server.print(readString);
        readString = "";
    }
}
```



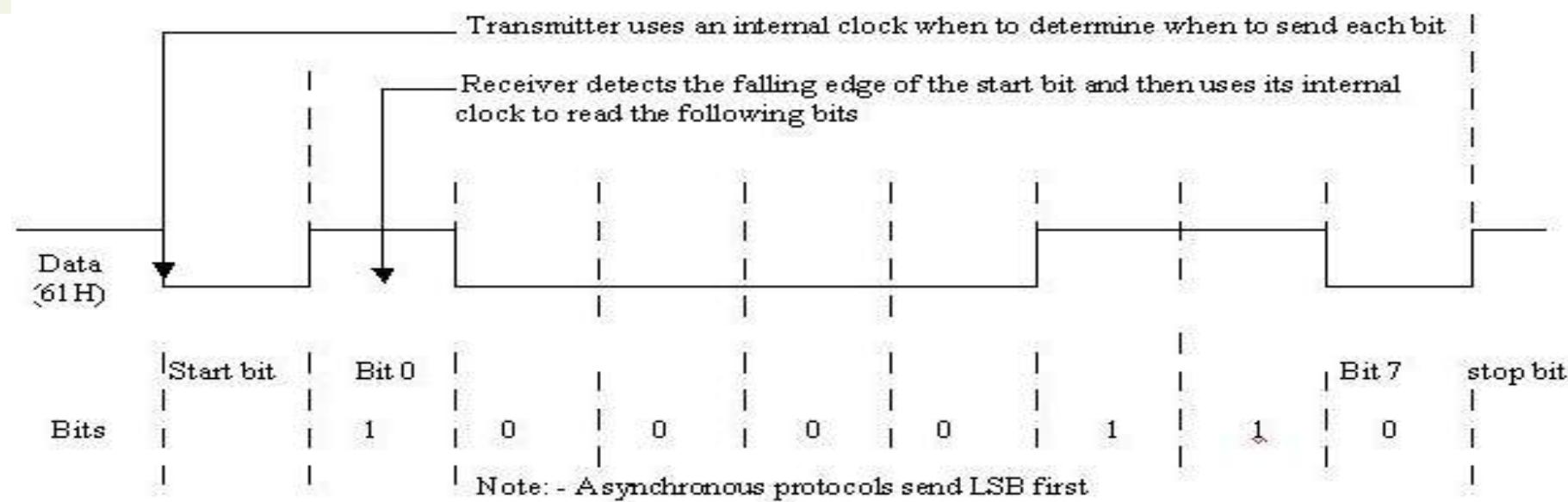
USB to TTL Converter UART Module  
CH340G CH340 3.3V 5V Switch



FT232RL for USB-serial Adapter

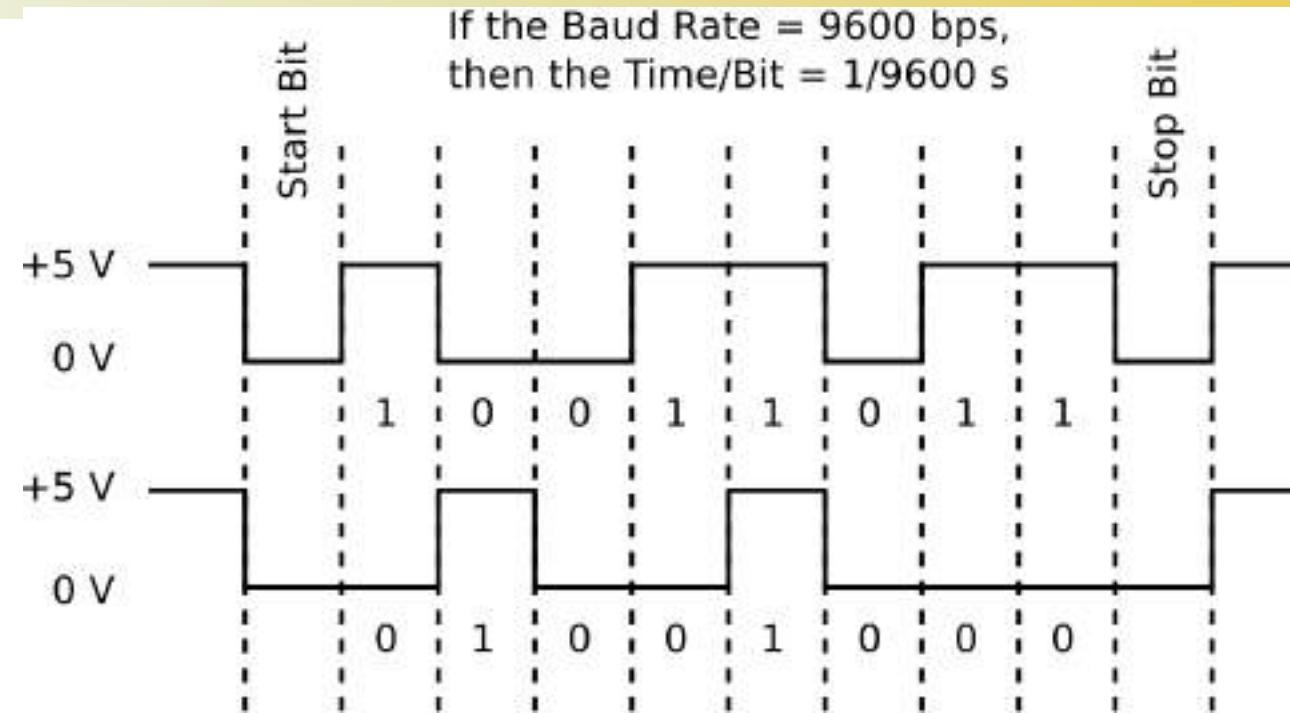


## How does Serial Data Transfer Work?



- The line is initially HIGH (no transmission).
- To start communication, the transmitter pull the line low for 1 bit interval called the start bit.
- A number of data bits (e.g., 8 bits) are then sent in ASCII format.
- The Tx then send 1 bit HIGH (stop bit) to retain the original state of the line.
- After sending the character, an interrupt is issued to the processor. The user may then uses suitable command to read the data.
- Thus, each transmitted character is packaged in a character frame.

## How does Serial Data Transfer Work?



Adjustable parameters

- Baud rate - Communication speed
- Data bits - Number of actual data bits in one byte
- Stop bit - Length of the stop after each sent byte
- Parity - Used for error checking

## A Character Frame Consists of:

- **Start bit:** the bit which signals the receiver that data is coming.
- **Data bits:** amount of actual data in a packet (5, 7, or 8 bits).
- **Stop bit:** is used to indicate the end of a single packet. Typical values are 1, 1.5, and 2 bits. The stop bits not only indicate the end of transmission **but also give the computers some room for error in the clock speeds.**
- **Parity:** It's used for error checking in serial communication. There are two types of parity: even and odd. The option of no parity is also available.

## How does Serial Data Transfer Work?

Serial data transfer requires specifying the following four parameters:

- 1) The speed of the transmission (**baud rate**)
- 2) The number of data bits encoding a character
- 3) The optional parity bit
- 4) The number of stop bits

## Baud Rate

- The Baud rate determines the transmission rate.
- Using the baud rate, both transmitter and receiver can adjust the time of each bit and get the data correctly.
- Note that if the baud rate is different in both Tx and Rx, the data will not be transmitted correctly.
- The Baud rate is not always the same as the bit rate.
- Baud rate: is the number of signal symbols occurring per second.
- The number of bits per symbol (baud) depends on the modulation scheme.
  - FSK: 1 bit per symbol
  - PSK: 4 bits per symbol
- So the bit rate (bps) and baud rate (baud per second) are related by the formula: **bps = baud per second x the number of bit per baud**



## Baud Rate: Numerical Example

Assume that the bit rate is 9600 bit/sec (bps). Find the time interval of one bit. How many bytes can be sent in one second? (keep in mind that we need start and stop bits).

### Answer:

- Interval of one bit =  $1/9600 = 104 \mu\text{sec}$ .
- Note that the Rx checks the line every half bit interval ( $104/2=52 \mu\text{sec}$ ) to collect the data bits.
- To send 8 bits of data, we need to transmit additional 1 start + 1 stop for a total of 10 bits. Hence we can send  $9600/10 = 960 \text{ Byte/sec}$ .

## Example: Arduino Serial Port

- `Serial.available()` Returns the number of bytes available in the serial input buffer.
- `Serial.write()` Write a single byte into the serial port
- `Serial.read()` Read a single byte from the input buffer

```
void setup()          // run once, when the sketch starts
{
    Serial.begin(9600); // set up Serial library at 9600 bps
    Serial.println("Hello world!");
    // prints hello with ending line break
}

void loop()          // run over and over again
{
    // do nothing!
}
```



## RS-232 Advantages and Disadvantages

### Advantages:

- RS-232 is a standard protocol.
- Noise immunity. (logic levels are relatively high  $\pm 25V$ ).
- At a baud rate of 9600 baud/sec, the cable length could be as long as 15m. For slower transmission rates, Longer distance can be used.

### Disadvantages:

- The need for logic level converters (e.g. MAX 232).
- RS232 has a slow rate which is more suitable for system-to-system communication rather than chip-to-chip applications.



## Applications

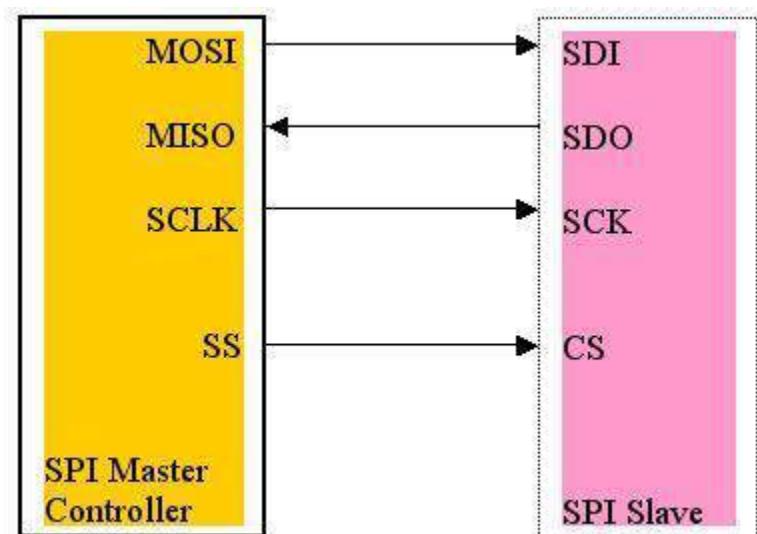
- Bus Type technology
- Used by many microprocessor/microcontroller to connect to peripheral devices such as ADC, memory modules, sensors, or other microprocessors.
- Speed up to 10Mbps.  
The SPI Bus is usually used only on the PCB.
- The desired recipient is select individually with dedicated wire
- Requires 2 or 3 wires for the communication +1 wire for each device in the bus
- Slaves send data to master at the same time when master is sending data to them
- Our example is the 3 wire spi (MISO, MOSI, SCLK pins)
- Works using the SPI -library in Arduino IDE



# SPI (Serial Peripheral Interface)

## Data and Control Lines of The SPI

Note: CE/CS (Chip Enable or Chip Select)



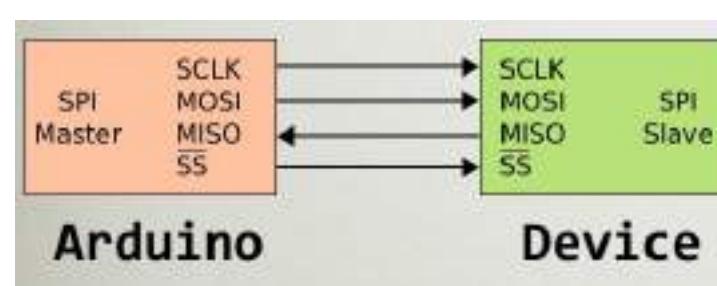
An SPI protocol specifies 4 signal wires:

- Master Out Slave In (MOSI)
- Master In Slave Out (MISO)
- Serial Clock (SCLK or SCK) - *generated by the Master*
- Slave Select (SS) from master to Chip Select (CS) of slave

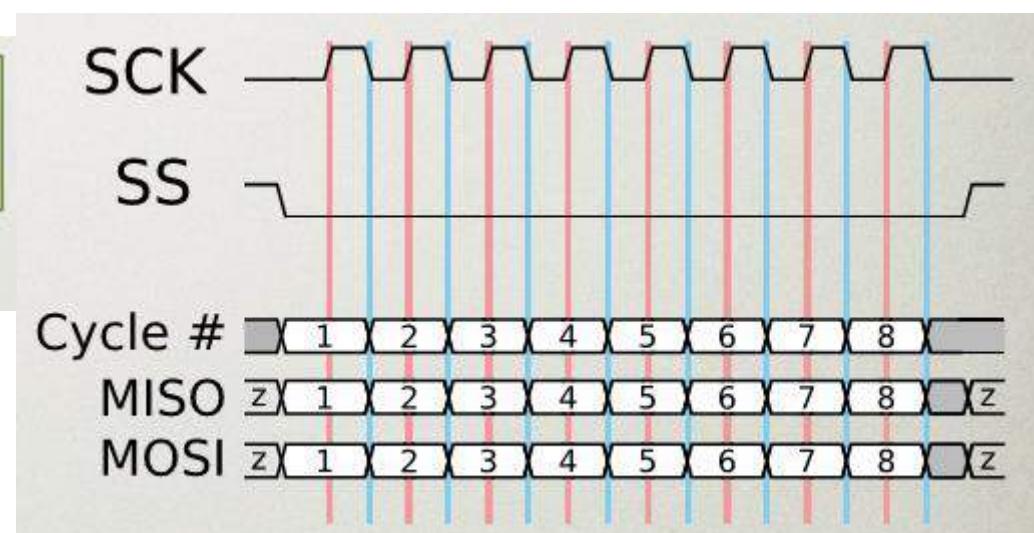
# SPI (Serial Peripheral Interface)

## SPI Protocol

- The communication is always initiated by the master.
- The master selects the desired slave by **pulling (SS) low**.
- Data transfer is organized by using **Shift register** in both transmitter and receiver.
- When there are no more data to be transmitted, the master stops its clock. Normally, it then rejects the slave.



SS: Slave Select



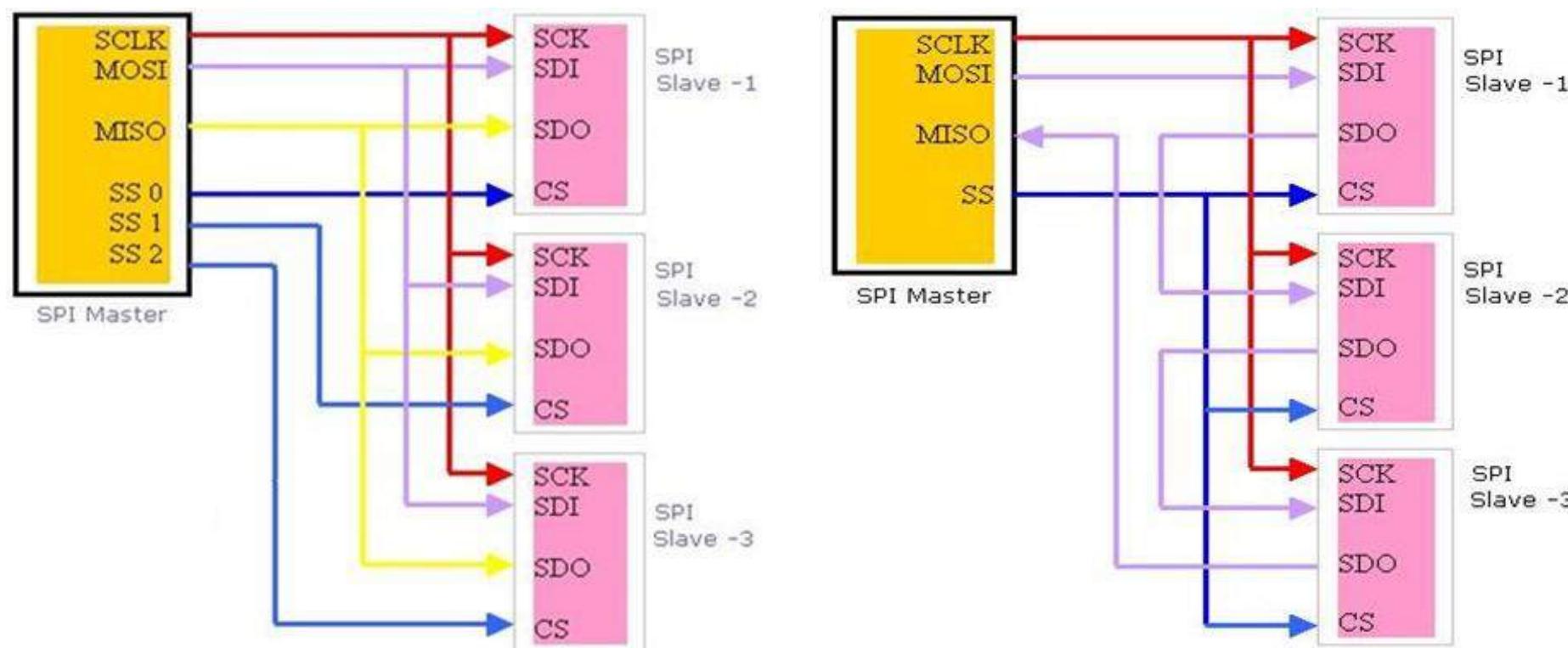
# SPI (Serial Peripheral Interface)



## Configurations for Multiple Slaves

Suppose a master-microcontroller needs to talk to multiple SPI Peripherals (slaves). There are 2 ways to set things up:

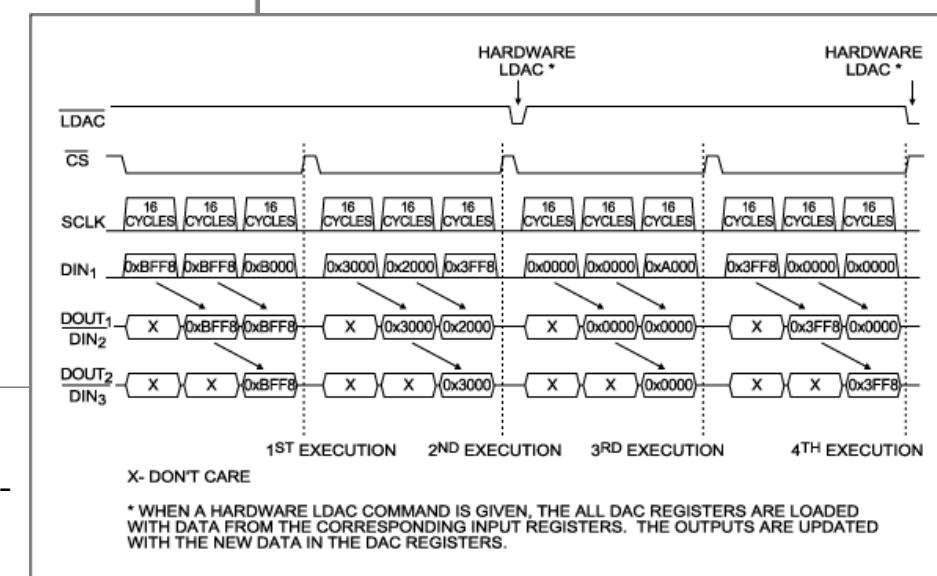
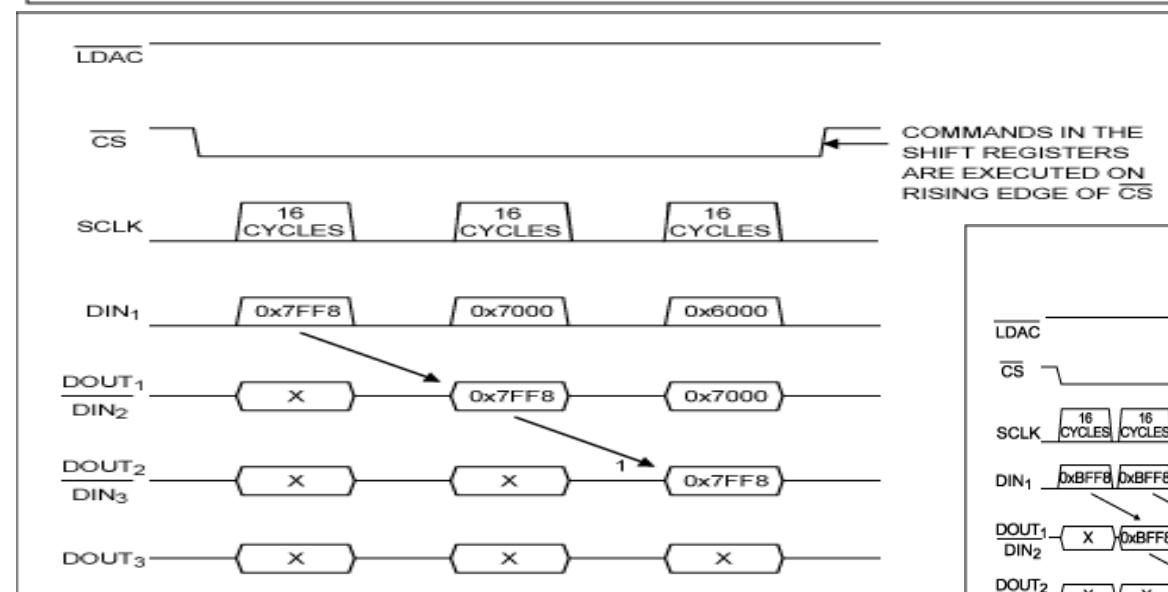
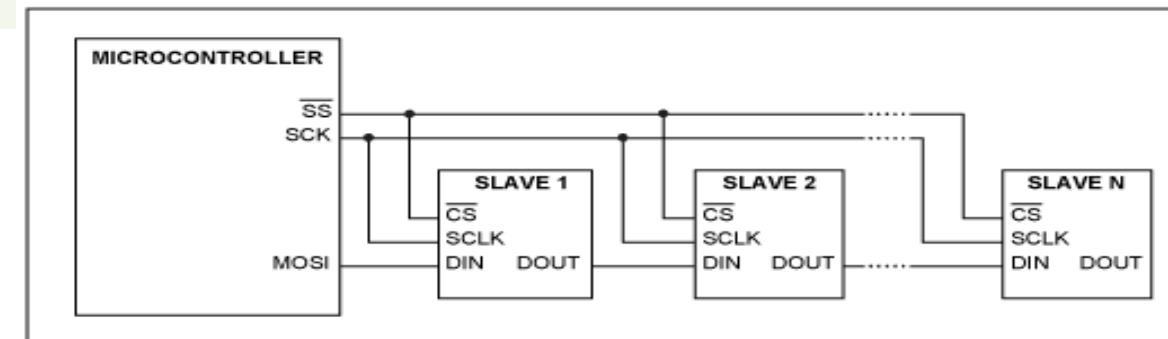
- Cascaded slaves or daisy-chained slaves
- Independent slaves or parallel configuration





# SPI (Serial Peripheral Interface)

## Daisy-Chained Slaves



Ref:

<https://www.maximintegrated.com/en/design/technical-documents/app-notes/3/3947.html>



## SPI Advantages and disadvantages

### Advantages:

1. Full duplex communication
2. Higher throughput than I<sup>2</sup>C protocol (that we will see next).
3. Slaves use the master's clock, and don't need precision oscillators.

### Disadvantages:

1. Requires more pins on IC packages than I<sup>2</sup>C
2. Only handles short distances compared to RS-232.

# SPI (Serial Peripheral Interface)



SPI



# SPI (Serial Peripheral Interface)



## Hardware

- SPI = Simple, 3+1 wire, full duplex, synchronous serial data transfer
- Interfaces to many devices, even many non-SPI peripherals
- Can be a master or slave interface

4 interface pins:

-MOSI      *master out slave in*  
-MIOS      *master in slave out*  
-SCK      *serial clock*  
-SS\_n      *slave select*

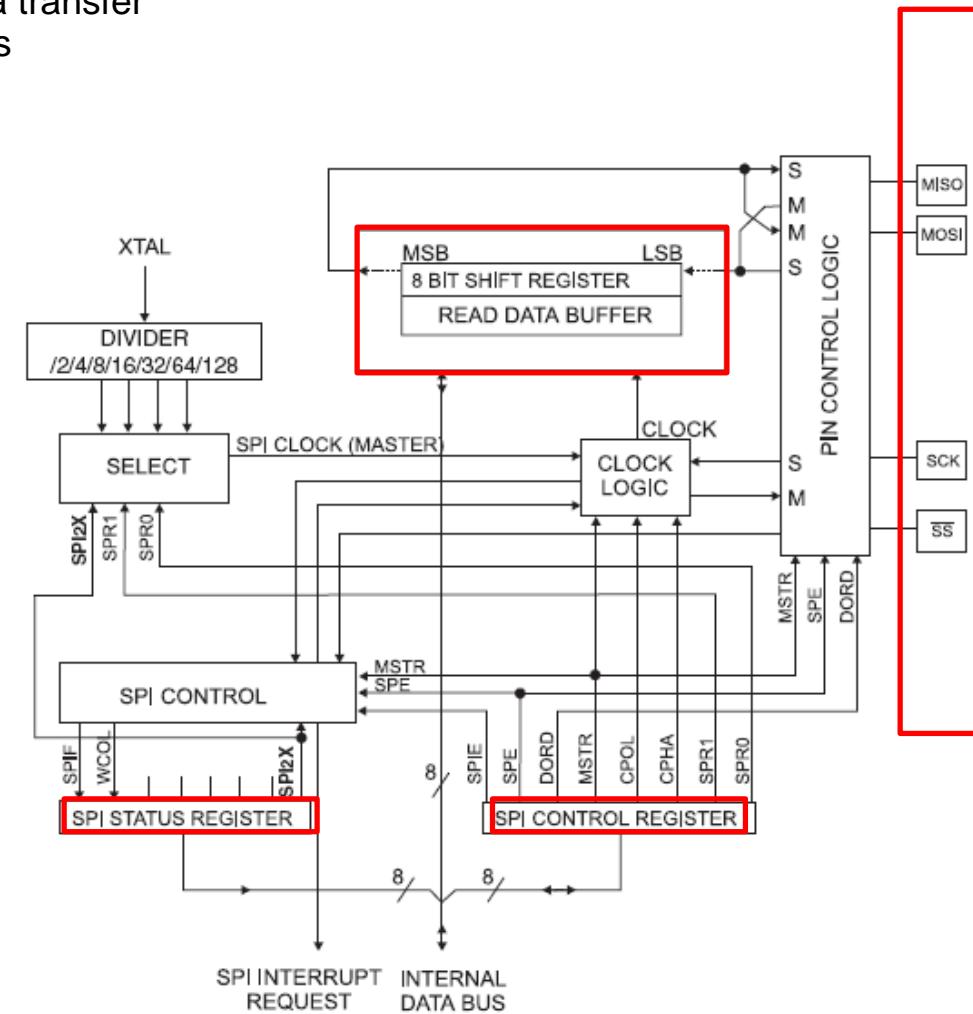
3 registers:

-SPCR      *control register*  
-SPSR      *status register*  
-SPDR      *data register*

1 - MISO      2 - +Vcc  
3 - SCK      4 - MOSI  
5 - Reset      6 - Gnd  
ICSP

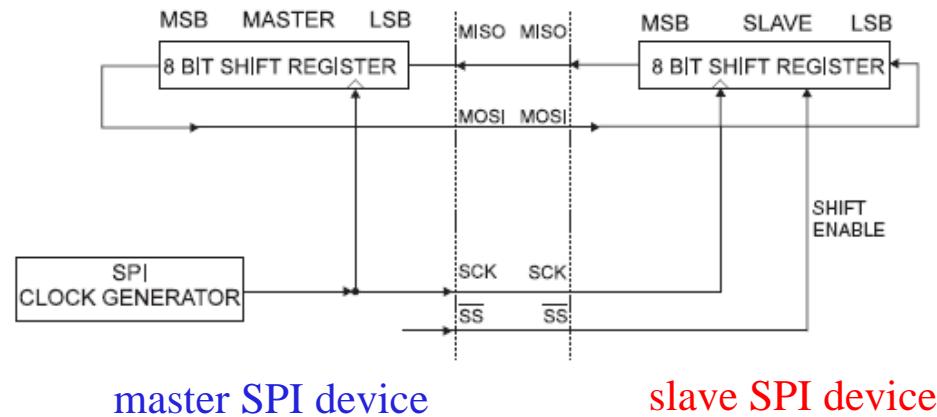


| Arduino Board      | MOSI         | MISO         | SCK          | SS (slave) |
|--------------------|--------------|--------------|--------------|------------|
| Uno or Duemilanove | 11 or ICSP-4 | 12 or ICSP-1 | 13 or ICSP-3 | 10         |



## Hardware

Full duplex, synchronous serial data transfer

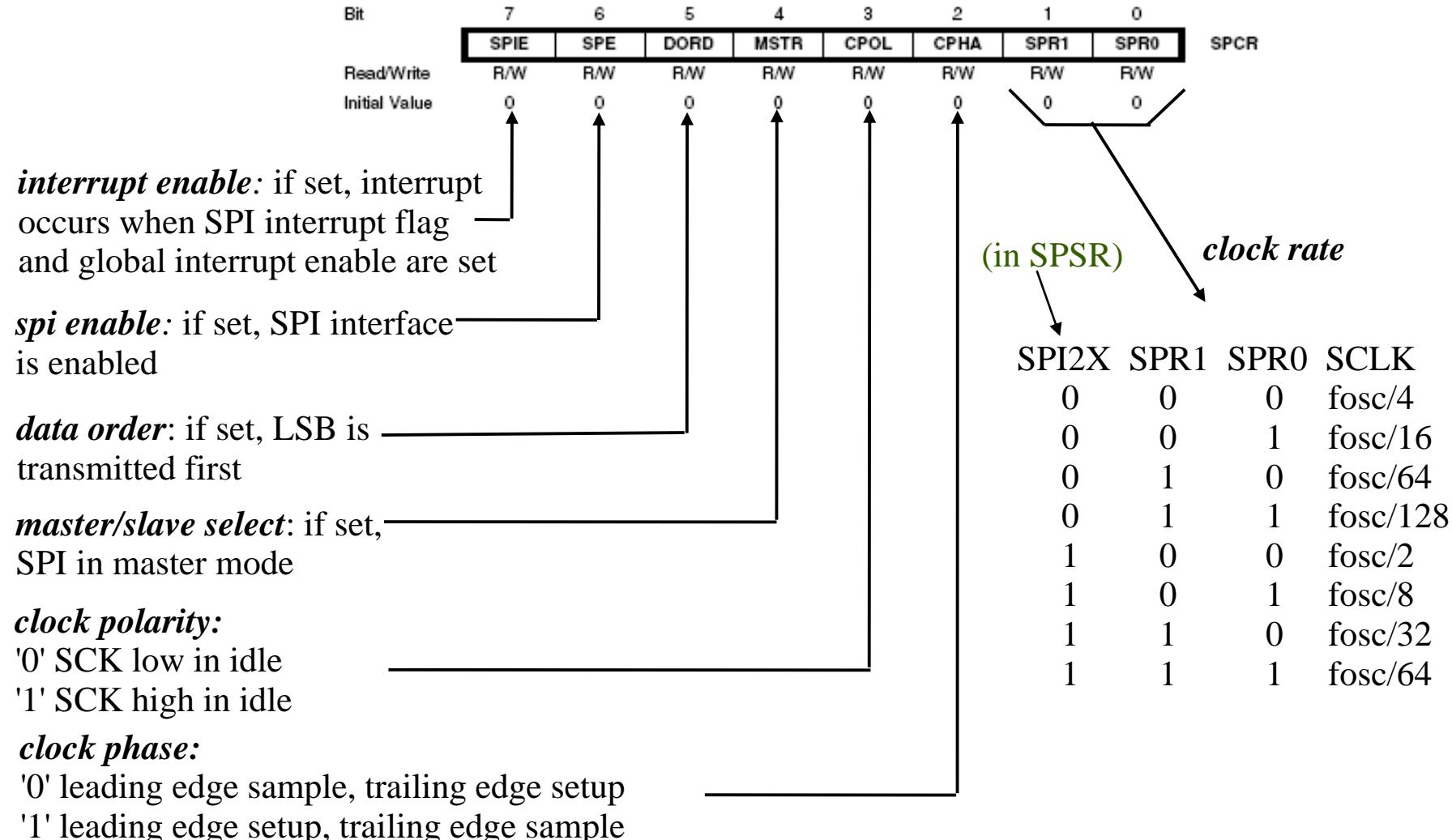


- Data is shifted out of the master's (mega128) MOSI pin and in its MISO pin
- Data transfer is initiated by simply writing data to the SPI data register.
- All data movement is coordinated by SCK.
- Slave select may or may not be used depending on interfacing device.
- To get input data only you send “junk” data to SPDR to start the clock.

# SPI (Serial Peripheral Interface)



## SPI Control Register (SPCR)





# SPI (Serial Peripheral Interface)

## The SPI Control Register (SPCR) 8 Bits

SPCR

|      |     |      |      |      |      |      |      |  |
|------|-----|------|------|------|------|------|------|--|
| 7    | 6   | 5    | 4    | 3    | 2    | 1    | 0    |  |
| SPIE | SPE | DORD | MSTR | CPOL | CPHA | SPR1 | SPR0 |  |

SPIE - Enables the SPI interrupt when 1

SPE - Enables the SPI when 1

DORD - Sends data least Significant Bit First when 1, most Significant Bit first when 0

MSTR - Sets the Arduino in master mode when 1, slave mode when 0

CPOL - Sets the data clock to be idle when high if set to 1, idle when low if set to 0

CPHA - Samples data on the falling edge of the data clock when 1, rising edge when 0

SPR1 and SPR0 - Sets the SPI speed, 00 is fastest (4MHz) 11 is slowest (250KHz)



# SPI (Serial Peripheral Interface)

## SPI Status Register (SPSR)

| Bit           | 7    | 6    | 5 | 4 | 3 | 2 | 1 | 0     |      |
|---------------|------|------|---|---|---|---|---|-------|------|
| Read/Write    | SPIF | WCOL | - | - | - | - | - | SPI2X | SPSR |
| Initial Value | 0    | 0    | 0 | 0 | 0 | 0 | 0 | 0     |      |
| reserved bits |      |      |   |   |   |   |   |       |      |

*interrupt flag:* set when serial transfer is complete

*write collision:* set if SPDR is written during a receive transfer

*2x clock rate:* if set, doubles clock rate in master mode

## SPI Data Register (SPDR)

| Bit           | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |           |
|---------------|-----|-----|-----|-----|-----|-----|-----|-----|-----------|
| Read/Write    | MSB |     |     |     |     |     |     | LSB | SPDR      |
| Initial Value | R/W | Undefined |

SPDR is a read/write register used for data transfer.

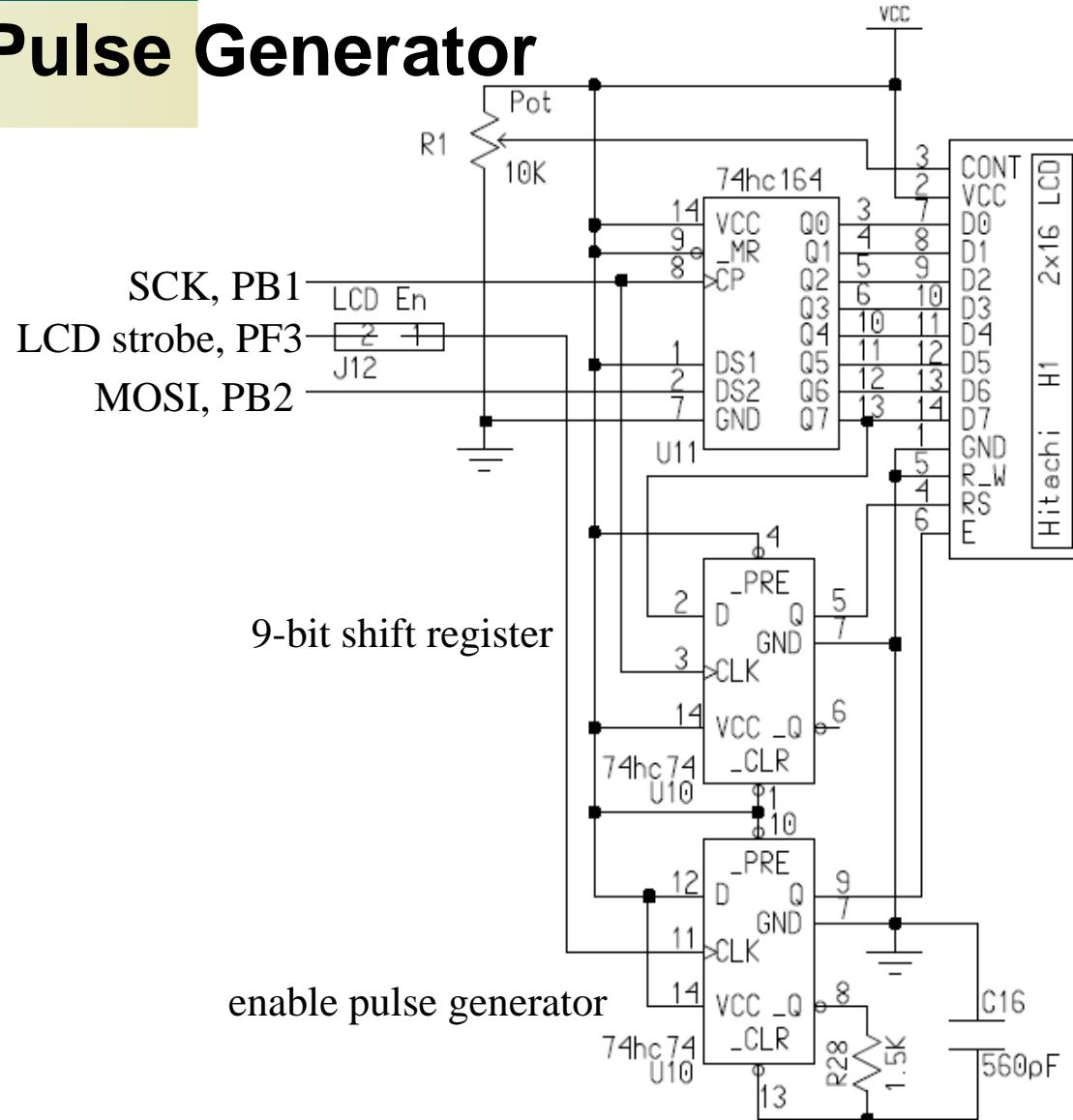
Writing SPDR sends data out MOSI.

Reading SPDR gets the data that was clocked into MISO.

# SPI (Serial Peripheral Interface)



## Shift Register and Pulse Generator



# SPI (Serial Peripheral Interface)



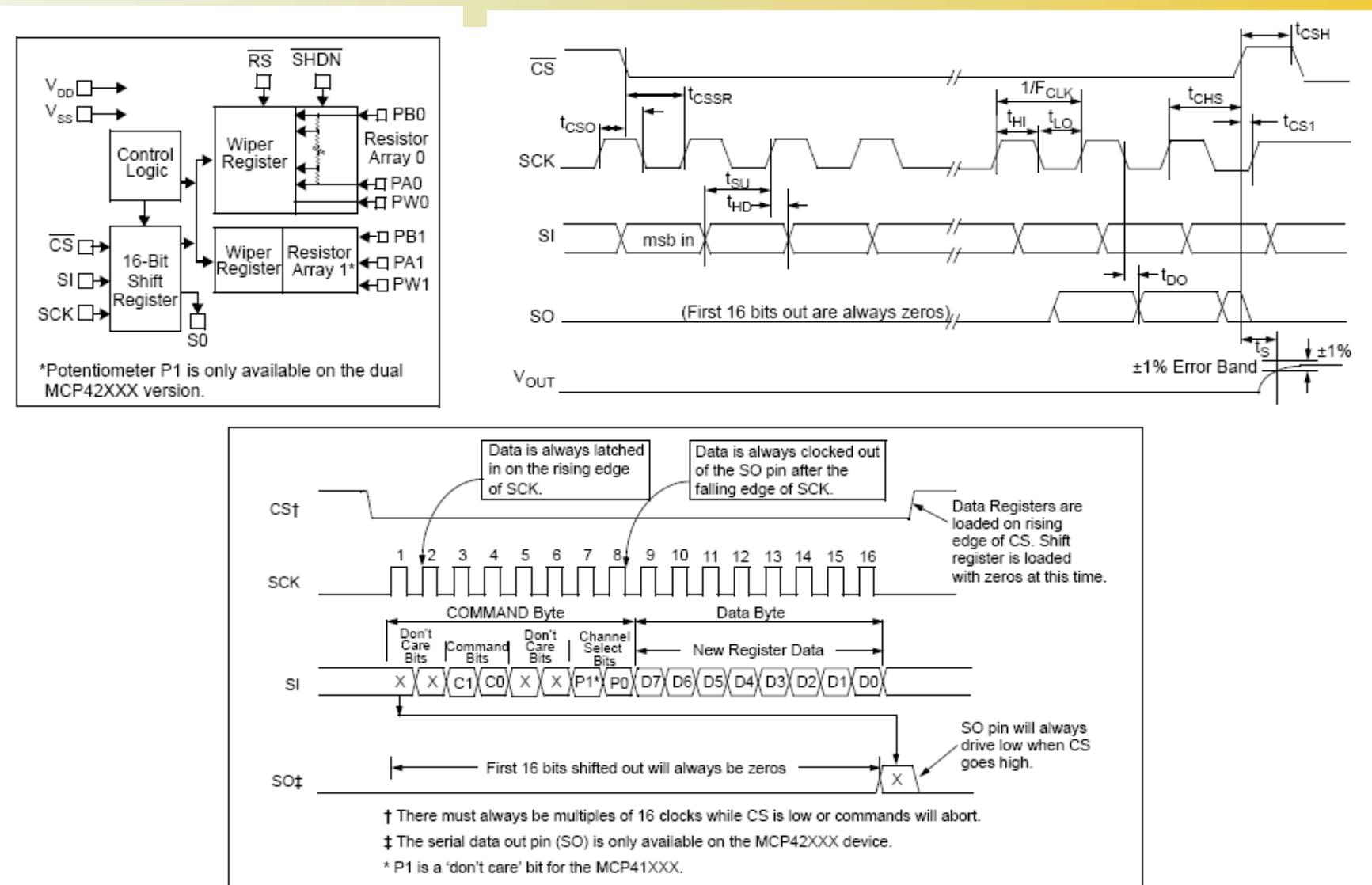
## SPI Application - Code

```
*****  
// spi_init  
//Initializes the SPI port on the mega128. Does not do any further  
//external device specific initializations.  
*****  
void spi_init(void){  
    DDRB = DDRB | 0x07;           //Turn on SS, MOSI, SCLK (SS is output)  
    SPCR |= (1<<SPEN) | (1<<MSTR); //spi enabled, master, low polarity, msb 1st  
    SPSR |= (1<<SPI2X);          //run at i/o clock div 2  
}//spi_init  
  
*****  
// digi_pot_send  
//Sends command and data to the digital pot. SPI device chip select is  
//active low and is connected to port F bit 2. Total of 16 bits are sent.  
//One byte for control and one byte as data passed in.  
*****  
void digi_pot_send(uint8_t data){  
    PORTF &= 0xFB;                //port F bit 2, assert active low  
    SPDR = 0x13;                  //send command byte (fixed value)  
    while (bit_is_clear(SPSR,SPIF)) {} //wait till data is sent out  
    SPDR = data;                  //send data byte  
    while (bit_is_clear(SPSR,SPIF)) {} //wait till data is sent out  
    PORTF |= 0x04;                 //port F bit 2, deassert to logic high  
} //digi_pot_send
```

# SPI (Serial Peripheral Interface)



Typical SPI IC (MCP42010)



# SPI (Serial Peripheral Interface)

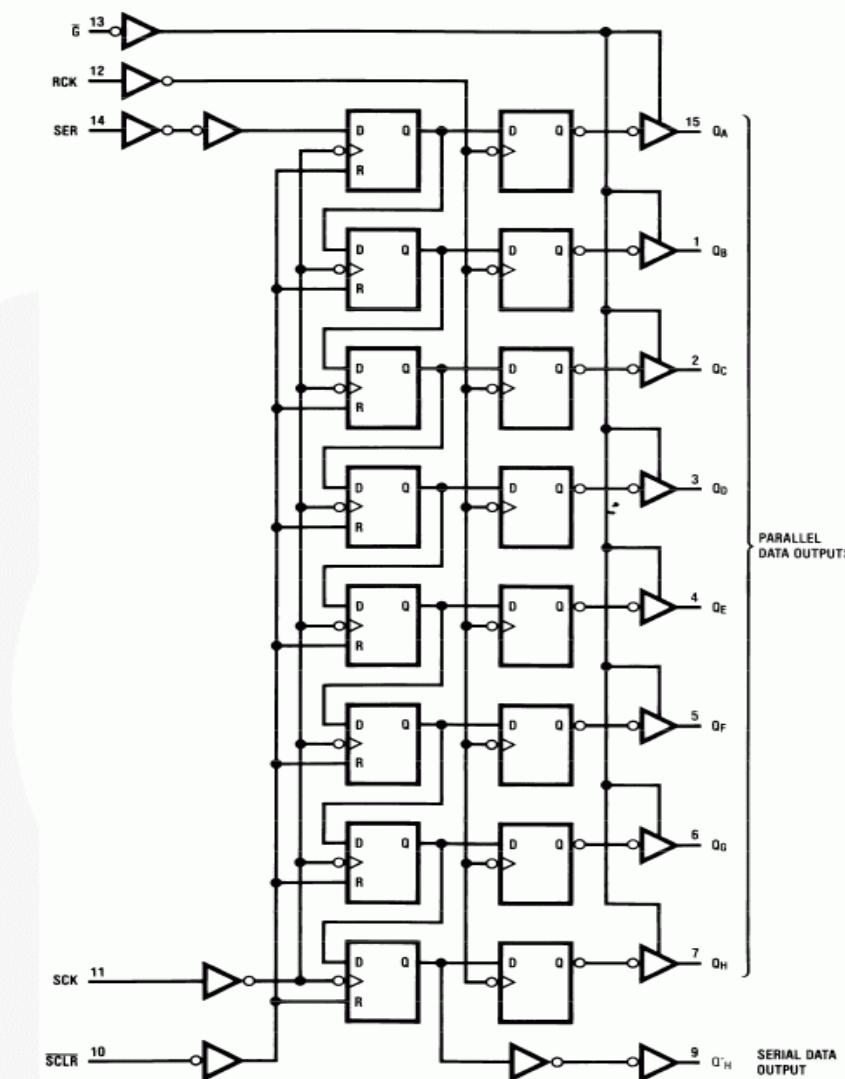


Figure 1. Logic Diagram (Positive Logic)

74HC595 – A perfectly fine SPI peripheral

# SPI (Serial Peripheral Interface)



## What if you want only to read the SPI port?

To get the SPI clock to run, a "dummy" write is made to the SPI SPDR register. This starts the clock running so the data on MISO is brought into the uC.

If no peripherals are selected, the outgoing data will be ignored. You can also send data out and bring data in at the same time.

```
*****  
//          spi_read  
//Reads the SPI port.  
*****  
uint8_t spi_read(void){  
    SPDR = 0x00;           // "dummy" write to SPDR  
    while (bit_is_clear(SPSR,SPIF)){} //wait till 8 clock cycles are done  
    return(SPDR);          //retrun incoming data from SPDR  
}//read_spi
```

## Some Troubles and Solutions

“Now my board won’t program.”

SPI shares SCK with programming interface. If it won’t program anymore, you likely messed up SCK.

“SPI acts totally weird.”

Often a symptom of SS\_n being configured as an input and being left to float or allowed to go high. SPI goes in and out between slave and master modes.

“I never get data to the SPI device.”

Is clock correctly oriented ? Did you assert the device chip select?

(**hint**: put SPI write inside a “tight” loop and check with scope.

Watch SCK, data, and chip select)

“SPI device interactions:”

- When programming, the programmer first does a chip reset.
- When the mega128 resets, all pins are set to input with high impedance (floating).
- If a SPI device is on the SPI bus, its chip-select may float low and enable the device, and SPI data will crash the programming data.
- Adding a pull-up resistor to chip selects will solve this problem.

# SPI (Serial Peripheral Interface)



## Example 1

```
#include "SPI.h"                                // necessary library
#define del 200
void setup()
{
    pinMode(10, OUTPUT);                         // we use this for SS pin
    SPI.begin();                                 // wake up the SPI bus.
    SPI.setBitOrder(MSBFIRST);                   // MSB byte first
}

void setValue(int value)
{
    digitalWrite(10, LOW);                      // using digital pin 10 for SPI slave select
    SPI.transfer(0);                            // send command byte
    SPI.transfer(value);                        // send value (0~255)
    digitalWrite(ss, HIGH);
}

void loop()
{
    for (int a=255; a>=0; --a)
    {
        setValue(a);
        delay(del);
    }
}
```

### SPI in Arduino

| Arduino Board      | MOSI         | MISO         | SCK          | SS (slave) |
|--------------------|--------------|--------------|--------------|------------|
| Uno or Duemilanove | 11 or ICSP-4 | 12 or ICSP-1 | 13 or ICSP-3 | 10         |

# SPI (Serial Peripheral Interface)

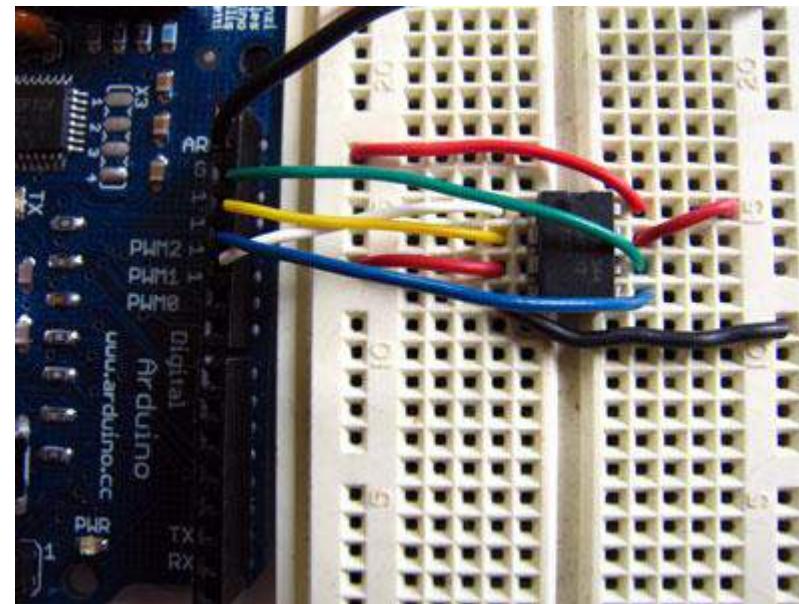
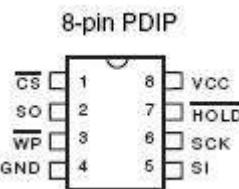


## Example 2: Interfacing a Serial EEPROM Using SPI

<http://arduino.cc/en/Tutorial/SPIEEPROM>

### Pin Configurations

| Pin Name | Function              |
|----------|-----------------------|
| CS       | Chip Select           |
| SCK      | Serial Data Clock     |
| SI       | Serial Data Input     |
| SO       | Serial Data Output    |
| GND      | Ground                |
| VCC      | Power Supply          |
| WP       | Write Protect         |
| HOLD     | Suspends Serial Input |



AT25HP512 Serial EEPROM chip (or similar)  
Hookup wire  
Arduino Microcontroller Module

The AT25HP512 is a 65,536 byte serial EEPROM.

It supports SPI modes 0 and 3, runs at up to 10MHz at 5v and can run at slower speeds down to 1.8v. Its memory is organized as 512 pages of 128 bytes each. It can only be written 128 bytes at a time, but it can be read 1-128 bytes at a time. The device also offers various degrees of write protection and a hold pin, but we won't be covering those in this tutorial.

The device is enabled by pulling the Chip Select (CS) pin low. Instructions are sent as 8 bit operational codes (opcodes) and are shifted in on the rising edge of the data clock. It takes the EEPROM about 10 milliseconds to write a page (128 bytes) of data, so a 10ms pause should follow each EEPROM write routine.



# Example 2

```

#define DATAOUT 11//MOSI
#define DATAIN 12//MISO
#define SPICLOCK 13//sck
#define SLAVESELECT 10//ss
//opcodes
#define WREN 6
#define WRDI 4
#define RDSR 5
#define WRSR 1
#define READ 3
#define WRITE 2
byte eeprom_output_data;
byte eeprom_input_data=0;
byte clr;
int address=0;
//data buffer
char buffer [128];
void fill_buffer()
{
    for (int I=0;I<128;I++)
    {
        buffer[I]=I;
    }
}
char spi_transfer(volatile char data)
{
    SPDR = data;                      // Start the transmission
    while (!(SPSR & (1<<SPIF)))     // Wait the end of the transmission
    {
    };
    return SPDR;                      // return the received byte
}

void loop()
{
    eeprom_output_data = read_eeprom(address);
    Serial.print(eeprom_output_data,DEC);
    Serial.print('\n',BYTE);
    address++;
    if (address == 128)
        address = 0;
    delay(500); //pause for readability
}

```

1

3

```

void setup()
{
    Serial.begin(9600);

    pinMode(DATAOUT, OUTPUT);
    pinMode(DATAIN, INPUT);
    pinMode(SPICLOCK,OUTPUT);
    pinMode(SLAVESELECT,OUTPUT);
    digitalWrite(SLAVESELECT,HIGH); //disable device
    // SPCR = 01010000
    //interrupt disabled,spi enabled,msb 1st,master,clk low when idle,
    //sample on leading edge of clk,system clock/4 rate (fastest)
    SPCR = (1<<SPE)|(1<<MSTR);
    clr=SPSR;
    clr=SPDR;
    delay(10);
    //fill buffer with data
    fill_buffer();
    //fill eeprom w/ buffer
    digitalWrite(SLAVESELECT,LOW);
    spi_transfer(WREN); //write enable
    digitalWrite(SLAVESELECT,HIGH);
    delay(10);
    digitalWrite(SLAVESELECT,LOW);
    spi_transfer(WRITE); //write instruction
    address=0;
    spi_transfer((char)(address>>8)); //send MSByte address first
    spi_transfer((char)(address)); //send LSByte address
    //write 128 bytes
    for (int I=0;I<128;I++)
    {
        spi_transfer(buffer[I]); //write data byte
    }
    digitalWrite(SLAVESELECT,HIGH); //release chip
    //wait for eeprom to finish writing
    delay(3000);
    Serial.print('h',BYTE);
    Serial.print('i',BYTE);
    Serial.print('\n',BYTE);//debug
    delay(1000);
}
byte read_eeprom(int EEPROM_address)
{
    //READ EEPROM
    int data;
    digitalWrite(SLAVESELECT,LOW);
    spi_transfer(READ); //transmit read opcode
    spi_transfer((char)(EEPROM_address>>8)); //send MSByte address first
    spi_transfer((char)(EEPROM_address)); //send LSByte address
    data = spi_transfer(0xFF); //get data byte
    digitalWrite(SLAVESELECT,HIGH); //release chip, signal end transfer
    return data;
}

```

# SPI (Serial Peripheral Interface)

## Example 3

```
#include <SPI.h>
// The DS1306 CE (chip enable) pin is connected to Arduino digital pin 10.
int chipEnablePin = 10;
void setup() {
    pinMode(chipEnablePin, OUTPUT); // Set the Chip enable pin as OUTPUT.
    // The DS1306 datasheet states the chip enable signal must be asserted HIGH
    // during a read or a write so set it LOW before the SPI interface is set up.
    digitalWrite(chipEnablePin, LOW);
    SPI.begin();
    Serial.begin(9600);
    // The DS1306 datasheet states that data is clocked into or out of the registers
    // Most Significant Bit first and that it Supports Motorola SPI Modes 1 and 3. We'll use
    // Mode 1.
    SPI.setBitOrder(MSBFIRST);
    SPI.setDataMode(SPI_MODE1);           // or SPI_MODE3
    digitalWrite(chipEnablePin, HIGH); // Initialize RTC Control Register
    // Enable the DS1306 by taking the Chip Enable pin HIGH
    SPI.transfer(0x8f);
    // Address the Control Register for a Write.
    SPI.transfer(B00000100);           // bit 2 set = 1Hz (pin 7) ON
    digitalWrite(chipEnablePin, LOW); // All done so take the Chip Enable LOW.
}
void loop() {
    digitalWrite(chipEnablePin, HIGH); //Enable the Chip Enable by taking it HIGH
    SPI.transfer(0x0f); // Address the Control Register but this time for a READ.
    byte inByte = SPI.transfer(0x0f); // Read the data into 'inByte' variable.
    digitalWrite(chipEnablePin, LOW);
    Serial.println(inByte, BIN);
    // Show the register in Binary on the Arduino Monitor.
    delay(5000); }
```

Ref: <http://www.vwlowen.co.uk/arduino/ds1306/ds1306.htm>

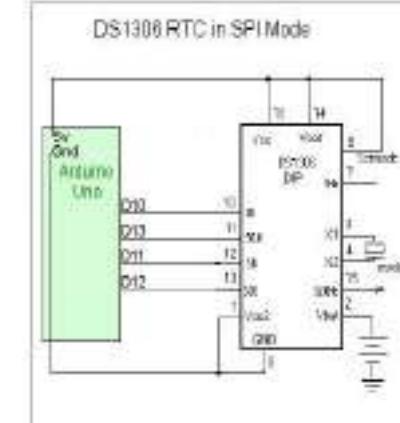
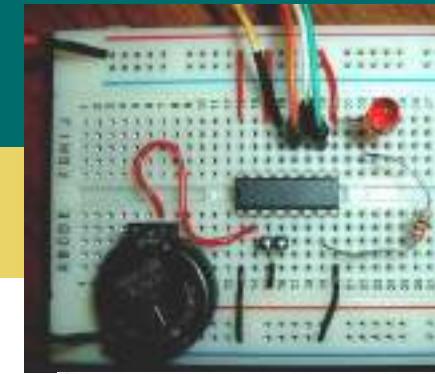


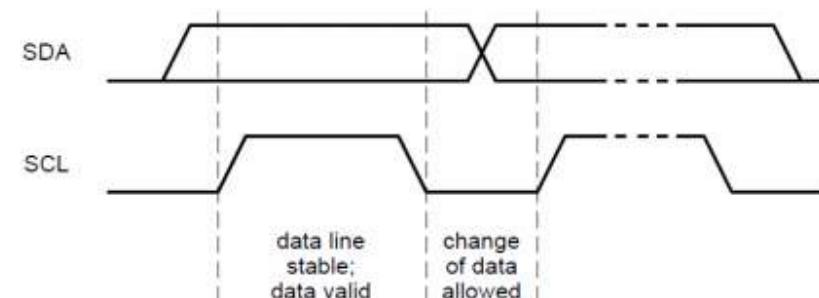
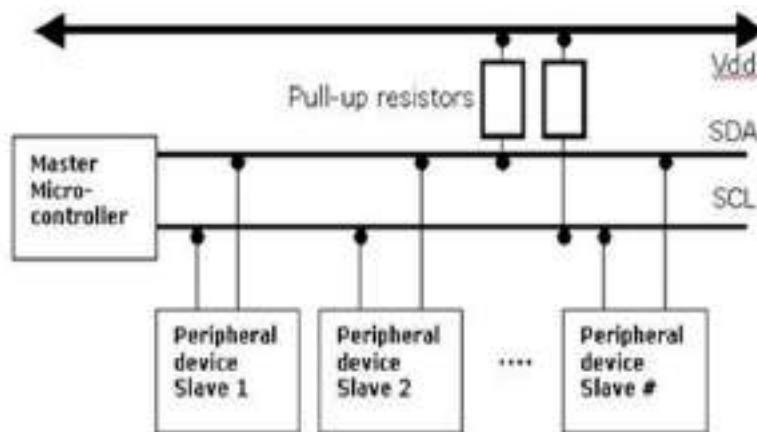
Figure 2. RTC REGISTERS AND ADDRESS MAP

| HEX ADDRESS | Bit7  | Bit6 | Bit5 | Bit4   | Bit3 | Bit2  | Bit1 | Bit0  | RANGE       |
|-------------|-------|------|------|--------|------|-------|------|-------|-------------|
| READ        | WRITE |      |      |        |      |       |      |       |             |
| 00h         | B0h   | 0    |      | 10 SEC |      |       | SEC  |       | 00-59       |
| 01h         | 81h   | 0    |      | 10 MIN |      |       | MIN  |       | 00-59       |
| 02h         | 82h   | 0    | 12   | P      | A    | 10-HR |      | HOURS | 01-12 + P/A |
|             |       |      | 24   | 10     |      |       |      |       | 00-23       |



## I<sup>2</sup>C Bus interface

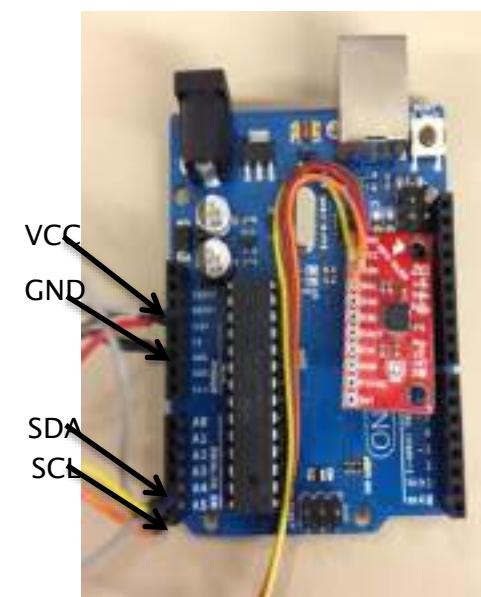
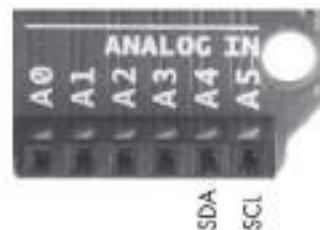
I<sup>2</sup>C aka Two-Wire-Interface (TWI) bus is a bi-directional 2-wire bus widely used for serial communication between integrated circuits on the same circuit board and to attach low-speed peripheral devices, sensors, and components to embedded systems.



- Bust Technology
- Both signals (SDA and SCL) are bidirectional
- Specifies connections, protocols, formats, addresses and procedures
- For each clock pulse, one bit of data is transferred
- Lines can only change when SCL is low

SDA: Data Line

SCL: Clock Line





## I2C Bus interface: Introduction

A large white cloud shape centered on a blue background. Inside the cloud, the letters "UCI" are written in a bold, black, sans-serif font.

**UCI**

University of  
California, Irvine



## I2C Bus interface: Transaction

A large white cloud shape centered on a blue background. Inside the cloud, the letters "UCI" are written in a bold, black, sans-serif font.

**UCI**

University of  
California, Irvine



## I2C Bus interface: Master





## I2C Bus Interface: Slave





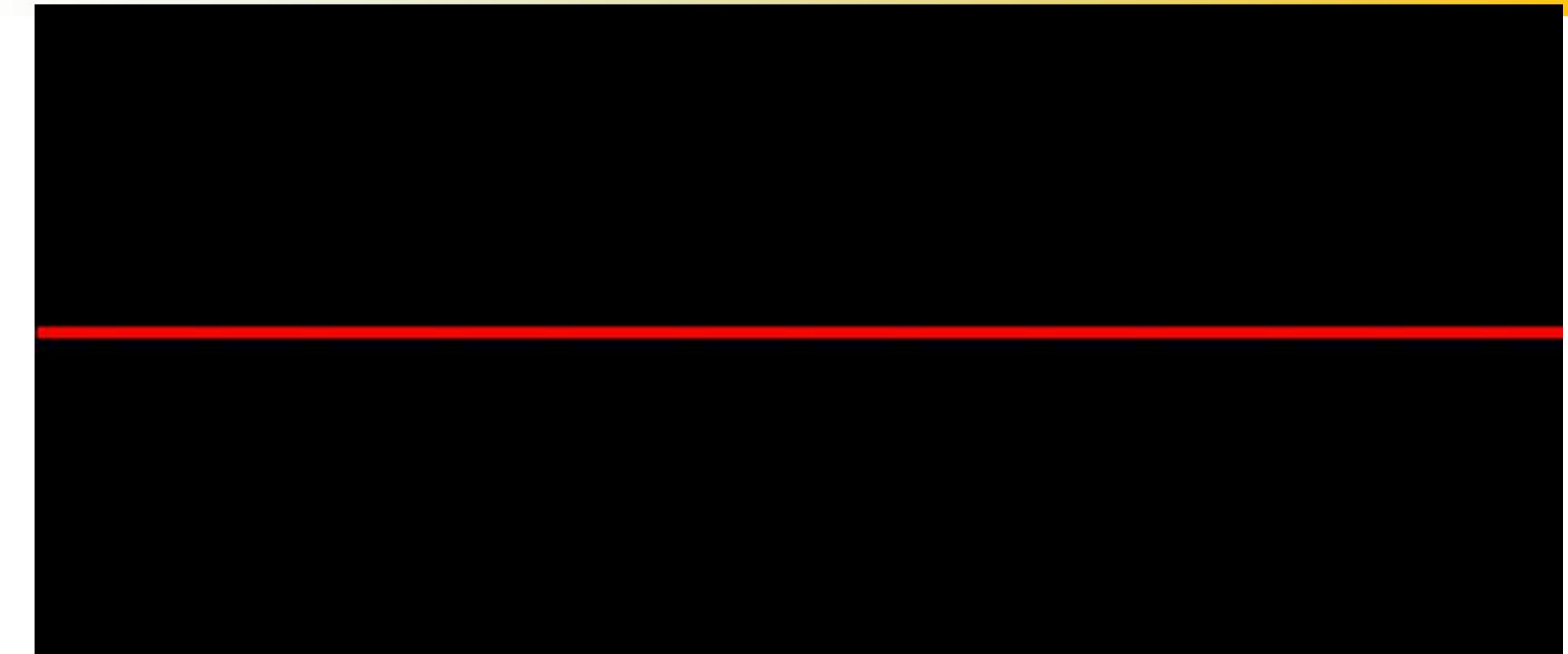
## I2C Bus Interface: Wire Library



University of  
California, Irvine



## I2C Bus interface



Ref: <https://www.youtube.com/watch?v=F4IzRPF7oOI>



## I2C Bus interface

I2C



## I2C Bus interface

<http://www.totalphase.com/support/articles/200349176-7-bit-8-bit-and-10-bit-I2C-Slave-Addressing>

- I2C has two bi-directional lines: Serial Data line (SDA) and Serial Clock line (SCL).
- I<sup>2</sup>C bus is a multi-master bus: more than one IC/device capable of initiating a data transfer can be connected to it.
- The device that initiates communication is called MASTER, the device being addressed by the Master is called SLAVE.
- It is the master's responsibility to generate the clock.
- I<sup>2</sup>C has a 7-bit address space with 16 reserved addresses, which makes the maximum number of nodes 112.

| 7bit address reservation |         |                                                  |  |
|--------------------------|---------|--------------------------------------------------|--|
| Slave Address            | R/W Bit | Description                                      |  |
| 000 0000                 | 0       | General call address                             |  |
| 000 0000                 | 1       | START byte <sup>(1)</sup>                        |  |
| 000 0001                 | X       | CBUS address <sup>(2)</sup>                      |  |
| 000 0010                 | X       | Reserved for different bus format <sup>(3)</sup> |  |
| 000 0011                 | X       | Reserved for future purposes                     |  |
| 000 01XX                 | X       | Hs-mode master code                              |  |
| 111 10XX                 | X       | 10-bit slave addressing                          |  |
| 111 11XX                 | X       | Reserved for future purposes                     |  |

**Valid Address Range**: 0x08 to 0x7f

**Reserved Addresses**: 0x00 to 0x07, 000 0XXX, 111 1XXX

**Count**: (8) Reserved Addresses, (112) Valid Address Range, (8) Reserved Addresses

## How the Master Communicate with Slave?

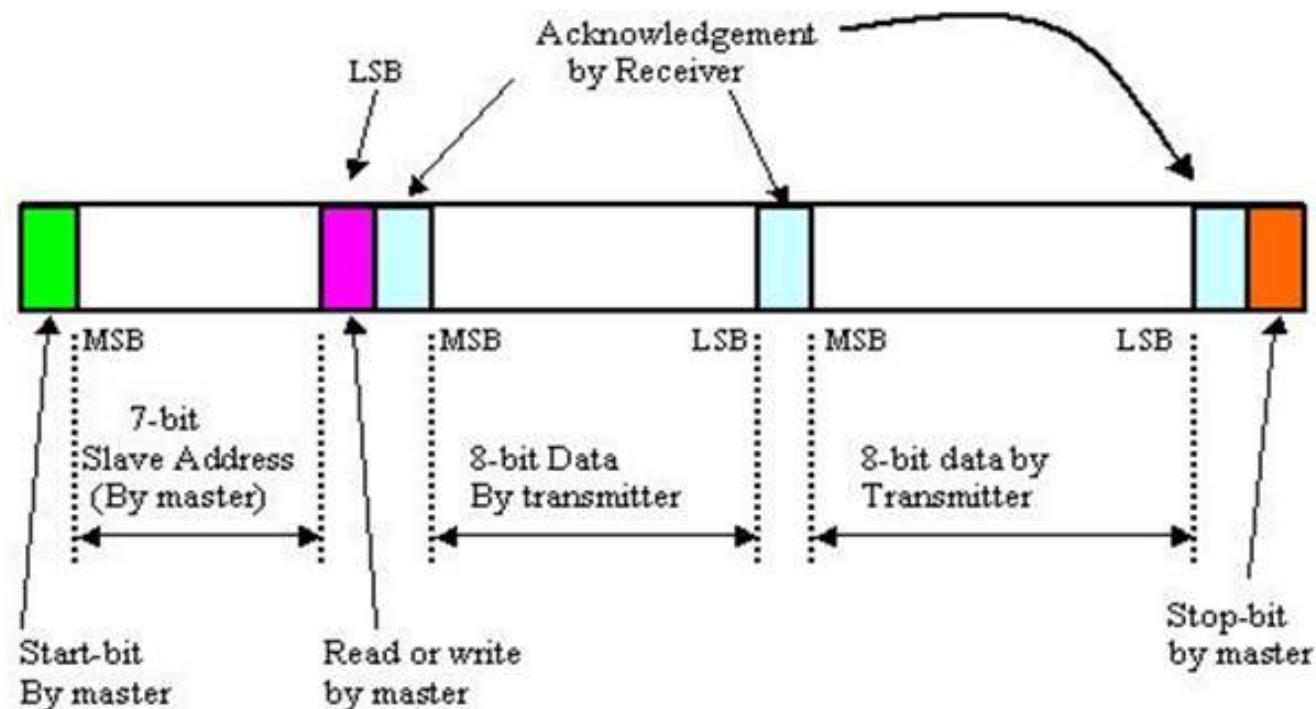


Fig-2 (I<sup>2</sup>C communication protocol)



## How the master communicate with slave?

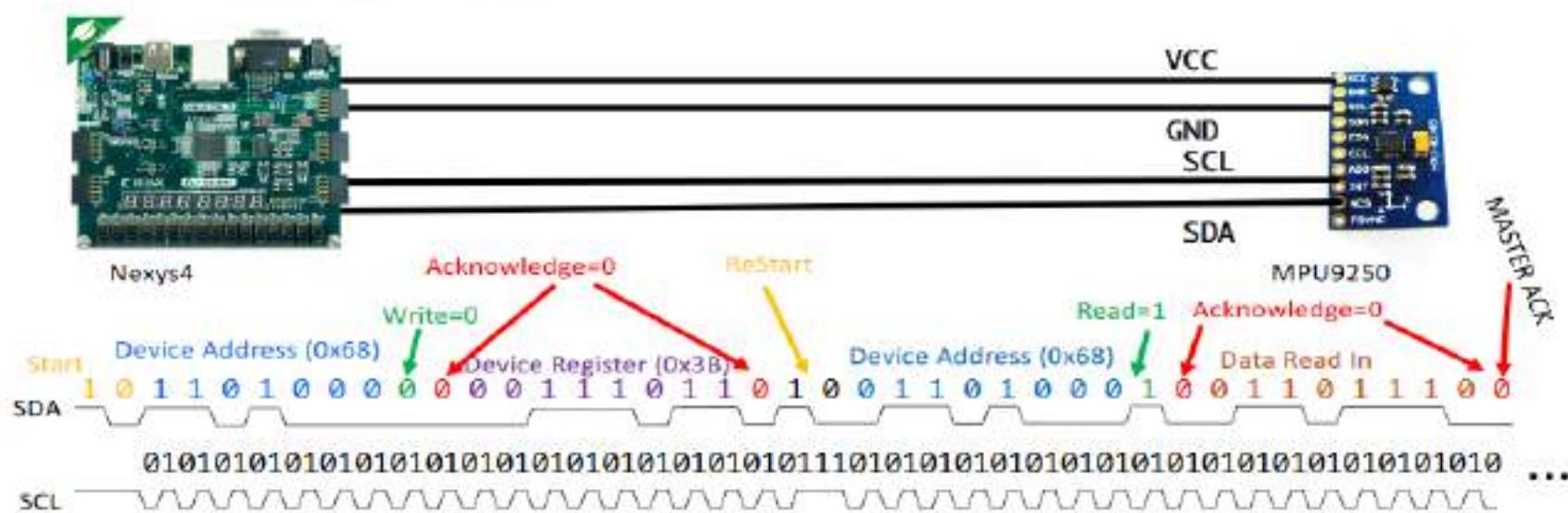
1. The MASTER waits until it sees *no activity* on the I2C bus (SDA and SCL are both HIGH).
2. The Master issues a start condition, informing other devices that it started to use the bus. This condition informs all the slave devices to listen on the serial data line for instructions/data.
3. The Master Provides the clock signal on the SCL line.
4. The Master sends the unique binary address of the target device it wants to access.
5. Master puts a one-bit message on the bus telling whether it wants to SEND or RECEIVE data from slave.
6. The Slave device with the matching address responds back with an acknowledgement signal.
7. The master then starts transmitting or receiving and the data communication proceeds between the Master and the Slave on the data bus. The transmitter sends 8-bits of data to the receiver, which replies with a 1-bit acknowledgement. And the action of data transfer continues.
8. When the communication is complete, the master issues a stop condition indicating that everything is done. This action free ups the bus.

Note that only two devices exchange data during one 'conversation'.



## How the master communicate with slave?

### Communication Part



6.111 Fall 2017

42

## I2C Advantage and Disadvantage

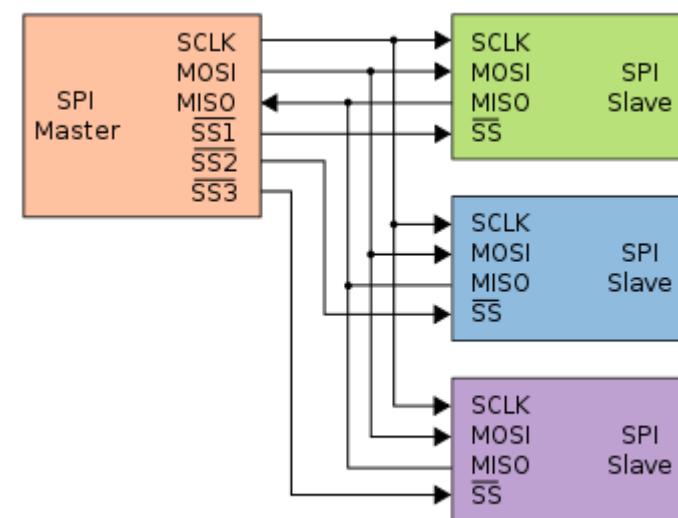
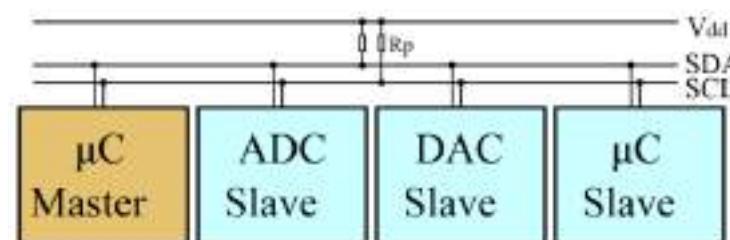
### Advantage

Only two wires are required. So, I2C is well suited for boards with many devices connected on the bus. This reduces cost & complexity of the circuit.

### Disadvantage

However, due to the presence of only two wires, there is additional overhead of addressing and acknowledgments. This can be inefficient in simple configurations and a direct-link interface such as SPI might be preferred.

| I2C (TWI)                                                                                                                                       | SPI                                                                                                                                                                                             |
|-------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Pins: SDA (A4), SCL (A5)                                                                                                                        | Pins: MISO (11), MOSI (12), SCK (13), SS (10/any)                                                                                                                                               |
| Up to 127 devices. Devices have an address (0x00 to 0x7F) which determines where data is sent/received from. SDA for both sending and receiving | Devices are chosen based on SS (slave select pin). One line for sending (MOSI, master out, slave in) and one for receiving (MISO, master in, slave out). Data can go both ways at the same time |
| Wire.h                                                                                                                                          | SPI.h                                                                                                                                                                                           |
| Easier to daisy-chain                                                                                                                           | Faster                                                                                                                                                                                          |



## Example 1

```
#include <Wire.h>
byte val = 0;
void setup()
{
    Wire.begin(); // join i2c bus
}
void loop()
{
    Wire.beginTransmission(44);
    // transmit to device #44 (0x2c)
    // device address is specified in datasheet
    Wire.write(val);          // sends value byte
    Wire.endTransmission();   // stop transmitting
    val++;                  // increment value
    if(val == 64) // if reached 64th position (max)
    {
        val = 0;      // start over from lowest value
    }
    delay(500);
}
```

```
#include <Wire.h>
void setup()
{
    Wire.begin();
    // join i2c bus (address optional for master)
    Serial.begin(9600); // start serial for output
}
void loop()
{
    Wire.requestFrom(2, 6);
    // request 6 bytes from slave device #2

    while(Wire.available())
    // slave may send less than requested
    {
        char c = Wire.read();
        // receive a byte as character
        Serial.print(c); // print the character
    }
    delay(500);
}
```

## I2C in Arduino

<http://arduino.cc/en/Reference/WireWrite>  
<http://arduino.cc/en/Reference/WireRead>

Board

Uno

I2C pins

A4 (SDA), A5 (SCL)



## Example 2: Digital POT MCP4018

```
int dt = 2000;
byte rval = 0x00;

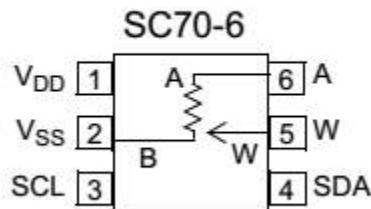
#include "Wire.h"
#define pot_address 0x2F

void setup(){
    Wire.begin();
    Serial.begin(9600);
}

void potLoop(){
    for(rval = 0; rval < 128; rval++){
        Wire.beginTransmission(pot_address);
        Wire.write(rval);
        Wire.endTransmission();
        Serial.print("sent - ");
        Serial.println(rval, HEX);
        delay(dt);
    }
}

void loop(){
    potLoop();
}
```

**MCP4018**



### I2C in Arduino

Board  
Uno

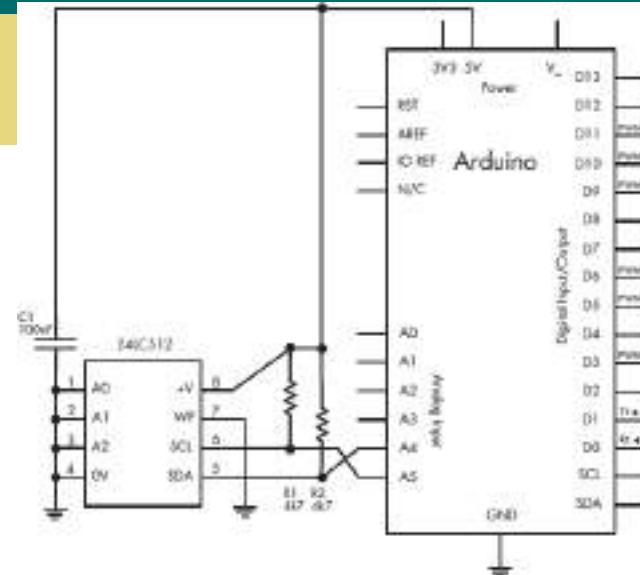
I2C pins  
A4 (SDA), A5 (SCL)

# I2C (Inter-Integrated Circuit)



## Example 3: EEPROM 24LC512

```
#include <Wire.h>
#define chip1 0x50
unsigned int pointer;
byte d=0;
void setup()
{ Serial.begin(9600);
  Wire.begin();
}
void writeData(int device, unsigned int address, byte data)
// writes a byte of data 'data' to the EEPROM at I2C address 'device'
// in memory location 'address'
{ Wire.beginTransmission(device);
  Wire.write((byte)(address >> 8)); // left part of pointer address
  Wire.write((byte)(address & 0xFF)); // and the right
  Wire.write(data);
  Wire.endTransmission();
  delay(10);
}
byte readData(int device, unsigned int address)
// reads a byte of data from memory location 'address'
// in chip at I2C address 'device'
{ byte result; // returned value
  Wire.beginTransmission(device);
  Wire.write((byte)(address >> 8)); // left part of pointer address
  Wire.write((byte)(address & 0xFF)); // and the right
  Wire.endTransmission();
  Wire.requestFrom(device,1);
  result = Wire.read();
  return result; // and return it as a result of the function readData
}
```

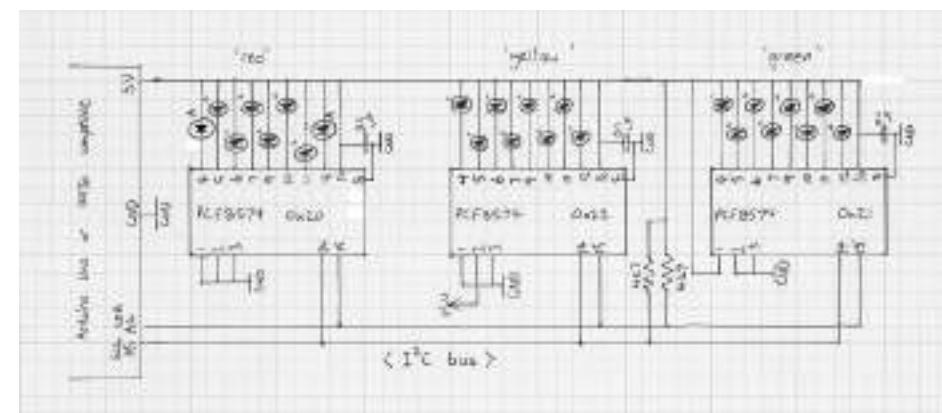


```
void loop()
{ Serial.println("Writing data...");
  for (int a=0; a<20; a++)
  { writeData(chip1,a,a);
  }
  Serial.println("Reading data...");
  for (int a=0; a<20; a++)
  { Serial.print("EEPROM position ");
    Serial.print(a);
    Serial.print(" holds ");
    f = readData(chip1,a);
    Serial.println(d, DEC);
  }
}
```



## Example 4: PCF8574 I/O Expansion (1/3)

```
#include "Wire.h"
#define redchip 0x20 // device addresses for PCF8547Ns on each LED colour bank
#define yellowchip 0x22
// addresses in this example match the published schematic in the tutorial
#define greenchip 0x21
// you will need to change addresses if you vary from the schematic
int dd=20; // used for delay timing void setup()
{
    wire.begin();
    allOff();
    // the PCF8574N defaults to high, so this functions turns all outputs off}
    // remember that the IC "sinks" current, that is current runs fro +5v through the LED and
    // then to I/O pin
    // this means that 'high' = off, 'low' = on.
}
void testfunc()
{
    Wire.beginTransmission(redchip);
    Wire.send(0);
    Wire.endTransmission();
    delay(dd+50);
    Wire.beginTransmission(redchip);
    Wire.send(255);
    Wire.endTransmission();
    delay(dd+50);
    Wire.beginTransmission(yellowchip);
    Wire.send(0);
    Wire.endTransmission();
    delay(dd+50);
    Wire.beginTransmission(yellowchip);
```



Ref: [http://8bitmicro.blogspot.com/2010\\_10\\_01\\_archive.html](http://8bitmicro.blogspot.com/2010_10_01_archive.html)



## Example 4: PCF8574 I/O Expansion (2/3)

```
Wire.send(255);
Wire.endTransmission();
delay(dd+50);
Wire.beginTransmission(greenchip);
Wire.send(0);
Wire.endTransmission();
delay(dd+50);
Wire.beginTransmission(greenchip);
Wire.send(255);
Wire.endTransmission();
delay(dd+50);
}
void testfunc2()
{ for (int y=1; y<256; y*=2)
  { Wire.beginTransmission(redchip);
    Wire.send(255-y);
    // we need the inverse, that is high = off
    Wire.endTransmission();
    delay(dd);
    Wire.beginTransmission(redchip);
    Wire.send(255);
    Wire.endTransmission();
    delay(dd);
  }
  for (int y=1; y<256; y*=2)
  { Wire.beginTransmission(yellowchip);
    Wire.send(255-y);
    Wire.endTransmission();
    delay(dd);
    Wire.beginTransmission(yellowchip);
    Wire.send(255);
    Wire.endTransmission();
    delay(dd);
  }
}
```

Ref: [http://8bitmicro.blogspot.com/2010\\_10\\_01\\_archive.html](http://8bitmicro.blogspot.com/2010_10_01_archive.html)

## Example 4: PCF8574 I/O Expansion (3/3)

```
void testfunc3()
{ Wire.beginTransmission(redchip);
Wire.send(0);
Wire.endTransmission();
Wire.beginTransmission(yellowchip);
Wire.send(0);
Wire.endTransmission();
Wire.beginTransmission(greenchip);
Wire.send(0);
Wire.endTransmission();
delay(dd+50);
allOff();
delay(dd+50);
}
void allOff()
{ Wire.beginTransmission(redchip);
Wire.send(255);
Wire.endTransmission();
Wire.beginTransmission(yellowchip);
Wire.send(255);
Wire.endTransmission();
Wire.beginTransmission(greenchip);
Wire.send(255);
Wire.endTransmission();
}

void loop()
{ for (int z=0; z<10; z++)
{ testfunc();
}
for (int z=0; z<10; z++)
{ testfunc2();
}
for (int z=0; z<10; z++)
{ testfunc3();
}
```

Ref: [http://8bitmicro.blogspot.com/2010\\_10\\_01\\_archive.html](http://8bitmicro.blogspot.com/2010_10_01_archive.html)



| Description                                                                                                   | I2C                           | SPI                                                         | UART                                                 |  |
|---------------------------------------------------------------------------------------------------------------|-------------------------------|-------------------------------------------------------------|------------------------------------------------------|--|
| Invented by                                                                                                   | 1982 by Philips Semiconductor | 1970 by Motorola                                            | 1960 by Gordon Bell at Digital Equipment Corporation |  |
| Synchronous data transfer<br>A clock line is required to synchronize the communication                        | True                          | True                                                        | False                                                |  |
| Asynchronous data transfer<br>Instead of a clock signal the data stream itself contain start and stop signals | False                         | False                                                       | True                                                 |  |
| Throughput                                                                                                    | 10,000 to 1,000,000 bits/s    | Up to 10,000,000 bits/s                                     | Up to 115,200 bits/s                                 |  |
| Slave need unique address                                                                                     | True                          | False                                                       | False                                                |  |
| Number of pins required                                                                                       | 2                             | 4+                                                          | 2                                                    |  |
| Error checking protocol                                                                                       | True                          | False                                                       | True                                                 |  |
| Multi-master<br>You can have multiple masters controlling one or multiple slaves                              | True                          | False                                                       | False                                                |  |
| Multi-slave<br>You can connect multiple slaves to a single master                                             | True                          | True                                                        | False                                                |  |
| Packet-switched<br>The transferred data is grouped in packages / messages, made of a header and a payload     | True                          | False                                                       | False                                                |  |
| Single-ended<br>The data is transferred by a single wire                                                      | True Serial Data (SDA)        | False Master in Slave Out (MISO) Master Out Slave In (MOSI) | False                                                |  |
| Serial connection<br>Data is transferred bit by bit along a single wire                                       | True                          | True                                                        | True                                                 |  |

- Created by Dallas (or Maxim) semiconductor
- 1-Wire is slower and smaller range than I2C.
- To allow remote devices to be connected to a microcontroller with two wires: (1) ground (GND) or return wire and (2) combined power and data
- For examples: printer cartridge identity devices, EEPROM flash memory, Analog-to-Digital Converters (ADC), DS18B20, DHT
- It provides for low-cost remote sensing by supplying power over the same wire used for data communications
  - Each sensor can accept power from the data line while the data line is in the high state
  - When the data line is active (going low), the sensor chips continue to run off of their internal capacitors (in parasitic mode)
  - The device also provides an optional VDD pin, allowing power to be supplied to it directly. This is sometimes used when parasitic mode doesn't work well enough. This, of course, requires an added wire, which adds to the cost of the circuit.
- We need to install additional onewire library from sketch menu.

## Line Driving

- Some voltage  $V$  (typically, +5 V) is applied to the 1-Wire bus through the **pull-up resistor**  $R_{\text{pullup}}$ .
- When the transistor M2 is in the Off state, **the voltage on the bus remains high because of the pull-up resistor.**
- However, when the master device activates transistor M2, **current is caused to flow from the bus to the ground**, acting like a signal short-circuit. Slave devices attached to the bus will see a voltage near zero.

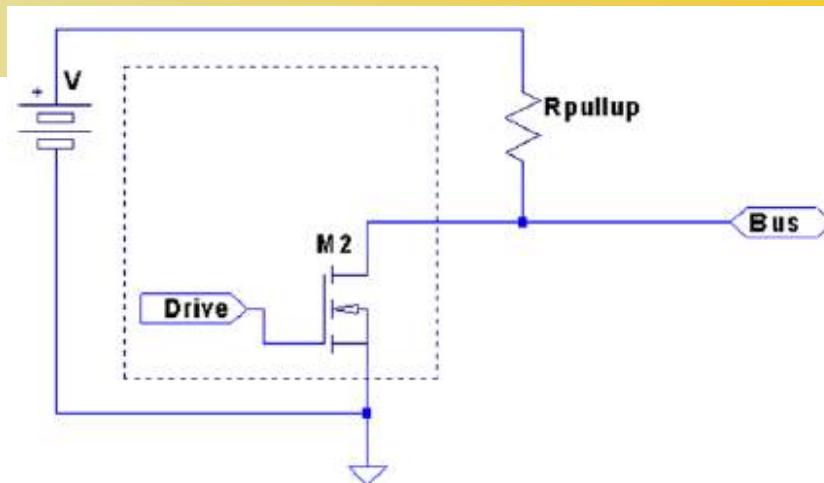


Figure 13-1. 1-Wire driver circuit

**Note** The Raspbian Linux 1-Wire bus uses GPIO 4 (GPCLK0) pin P1-07.

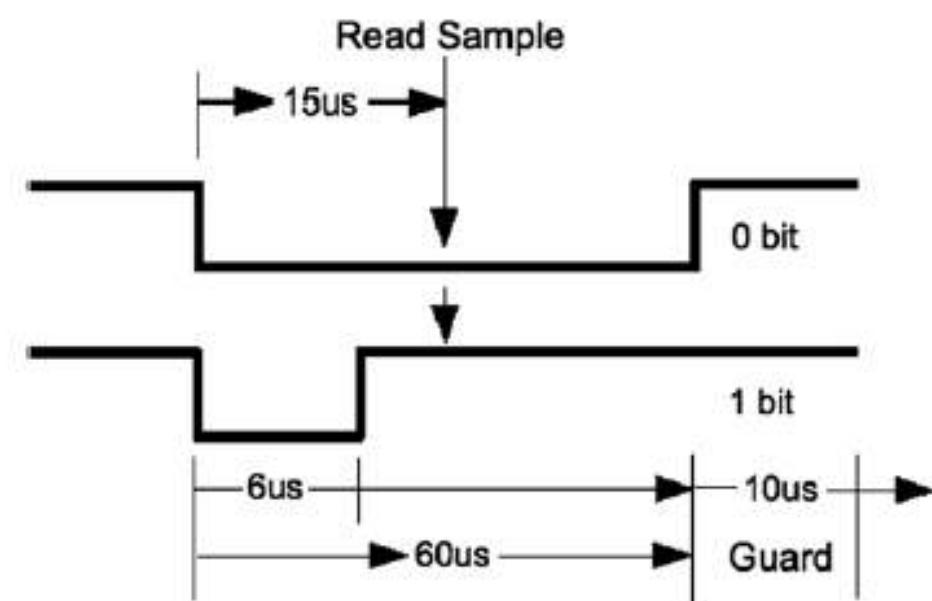


## Master and Slave

- The master device (controller) is always in control of the 1-Wire bus. Slaves (up to 255 devices) speak only to the master, and only when requested.
- There is never slave-to-slave device communication.
- If the master finds that communication becomes difficult for some reason, it may force a bus reset. This corrects for an errant slave device that might be jabbering on the line.
- Each slave device is given a unique ID known as its address during manufacturing, so it can be identified on the bus when there are multiple slaves. This address is 64 bits in length.
- It can be switched from being an input to being an output by the master to allow two-way communication.
- It has just a single data line and uses long and short pulses to signify 1s and 0s e.g. A pulse of 60 microS signifies a 0 and 15 microS indicates a 1.

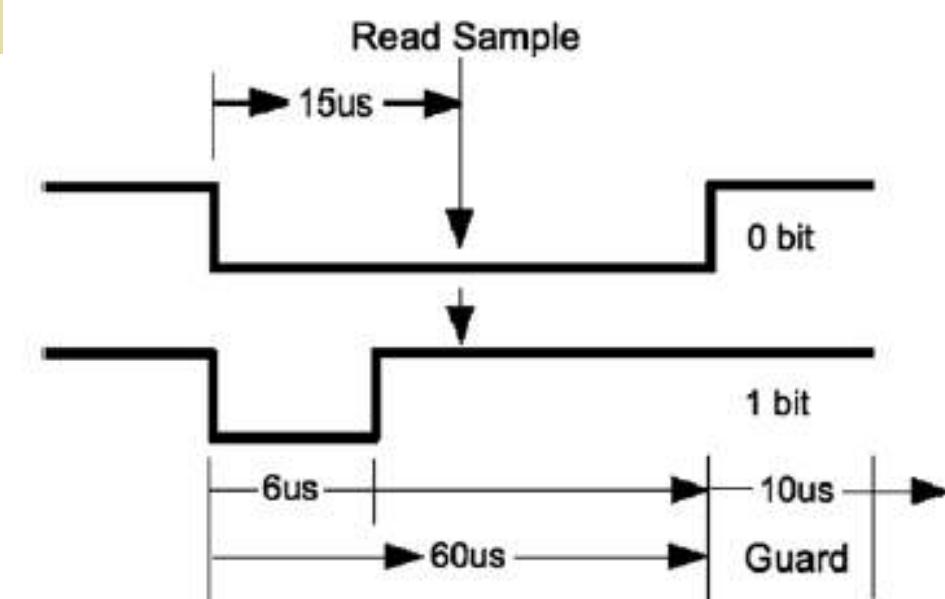
## Data I/O Protocol (1/2)

- Whether writing a 0 or 1 bit, the sending device brings the bus line low.
- This announces the start of a data bit.
- When a 0 is being transmitted, the line is held low for approximately 60 microsec. Then the bus is released and allowed to return high.
- When a 1 bit is being transmitted, the line is held low for only about 6 microsec before releasing the bus.
- Another data bit is not begun until 70 microsec after the start of the previous bit. This leaves a guard time of 10 microsec between bits. The receiver then has ample time to process the bit and gains some signal noise immunity



## Data I/O Protocol (2/2)

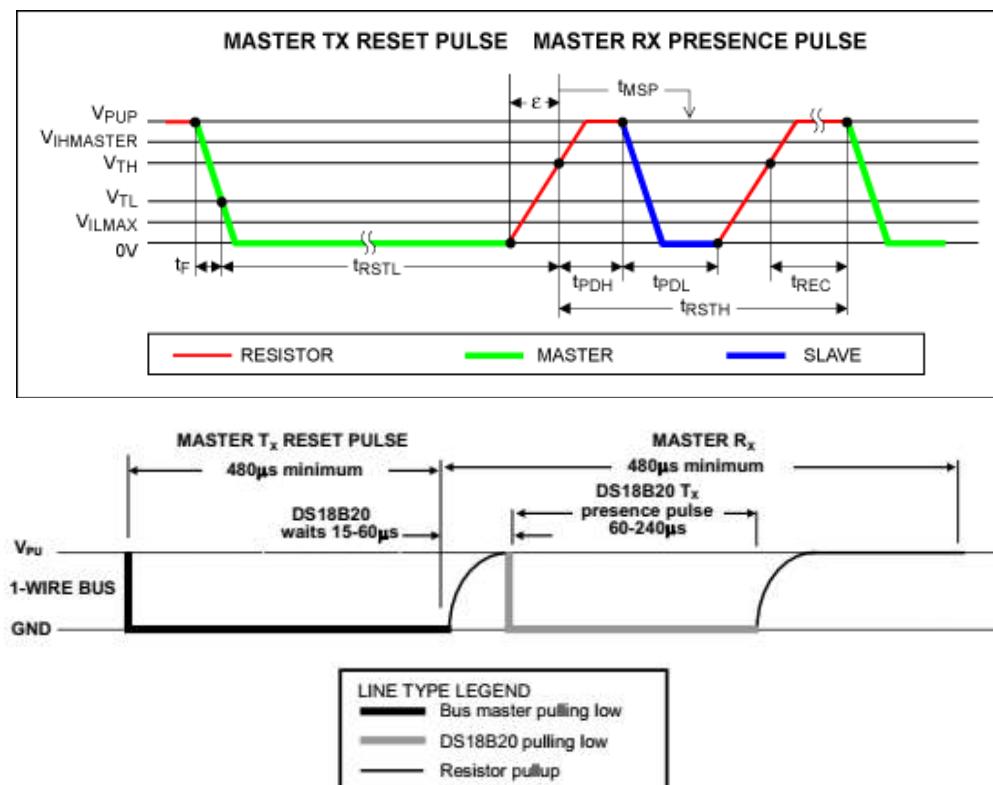
- The receiver notices a data bit is coming **when the line drops low**.
- It then starts a timer and samples the bus at approximately 15 microsec.
  - If the **bus is still in the low state**, a **0 data bit** is registered.
  - Otherwise, the **data bit is interpreted as a 1**.



- Having registered a data bit, the receiver then waits further until the line returns high (in the case of a 0 bit).
- The receiver **remains idle until it notices the line going low again**, announcing the start of the next bit.
- The sender can be either the master or the slave, but the **master always has control**.
- Slaves do not write data to the bus unless the master has specifically requested it.



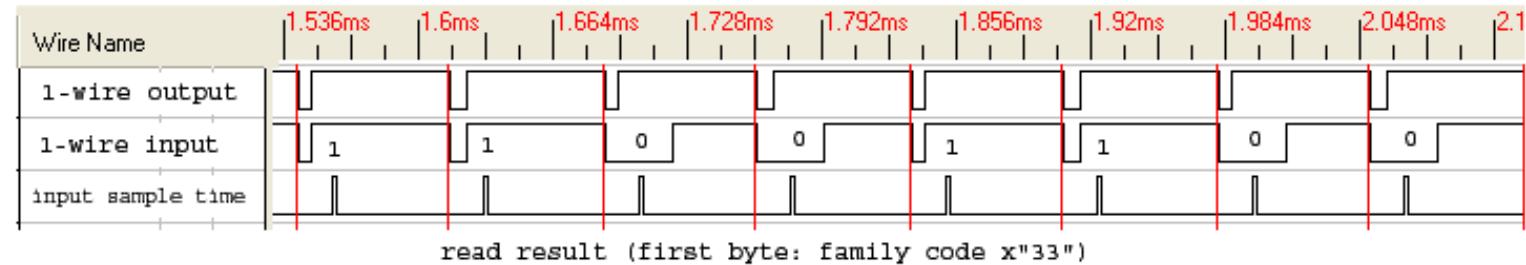
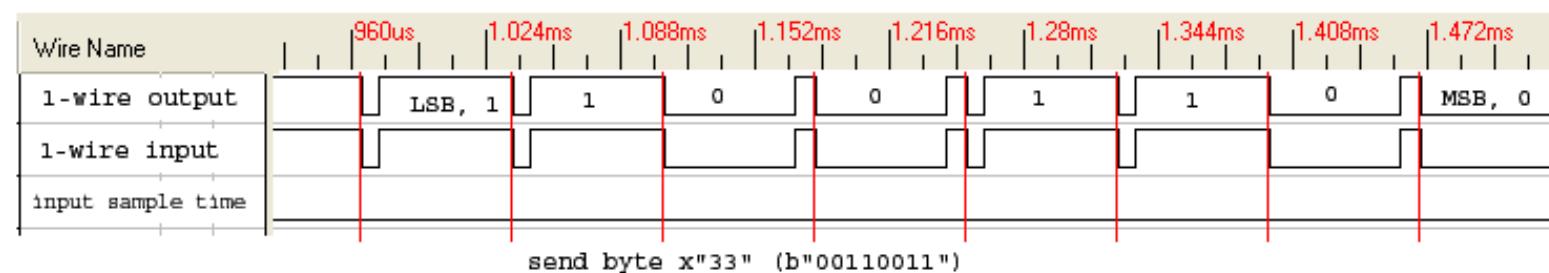
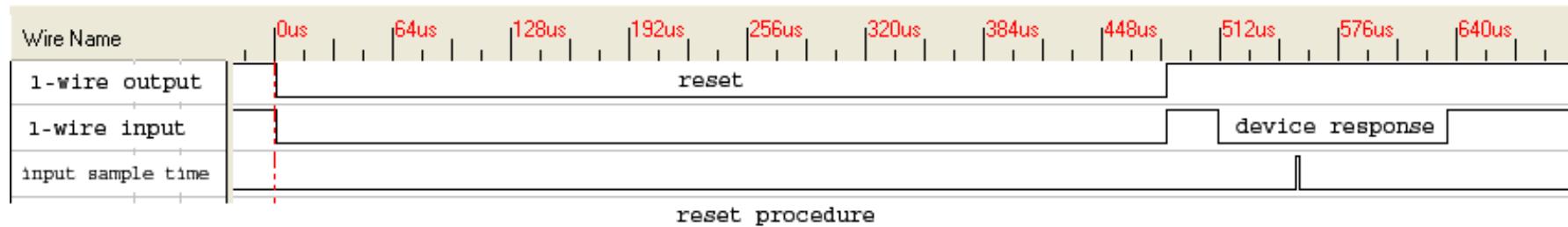
- The data line is normally HIGH, but when the master needs to send a command to the device, it sends a special “reset” LOW pulse, of at least 480 microseconds. The stream of 1 and 0 pulses then follow this.



<https://www.maximintegrated.com/en/app-notes/index.mvp/id/74>

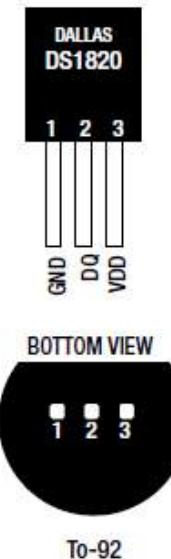


## 1 Wire reset, write and read example with DS2432



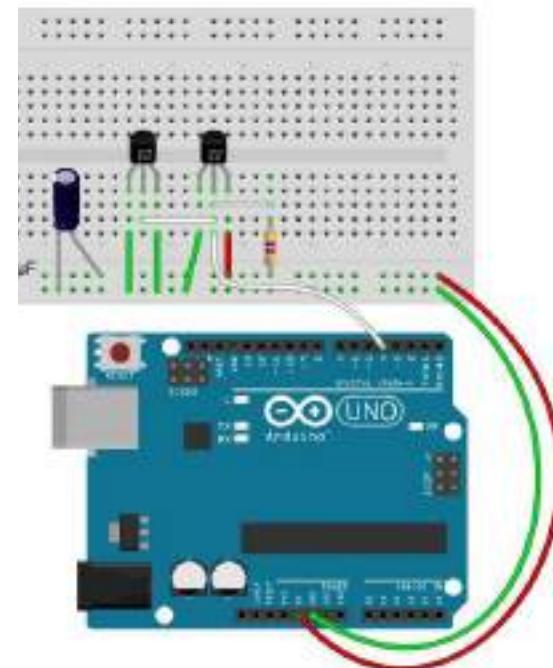
## DS18b20 Slave Device

- Slave devices are identified by a pair of digits representing the product family, followed by a hyphen and serial number in hexadecimal.
- The ID 28-00000478d75e is an example. You might also want to try different devices, like the similar DS18S20.
- DS18B20: relatively cheap, very easy to find, easy to use, and supply readings accurate to +/-0.5 degrees across the range -10 to +85 degrees Celsius.
- Reading temperature readings from multiple sensors (DS18b20) down one wire is possible because each DS18b20 sensor has a **unique serial number** coded into it at manufacture which the Arduino can be used to identify them by.



### PIN DESCRIPTION

|                 |                        |
|-----------------|------------------------|
| GND             | - Ground               |
| DQ              | - Data In/Out          |
| V <sub>DD</sub> | - Power Supply Voltage |

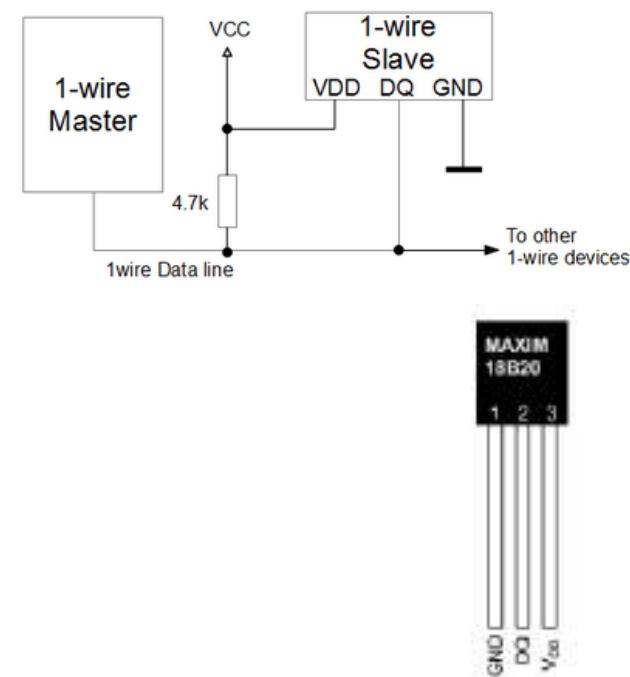
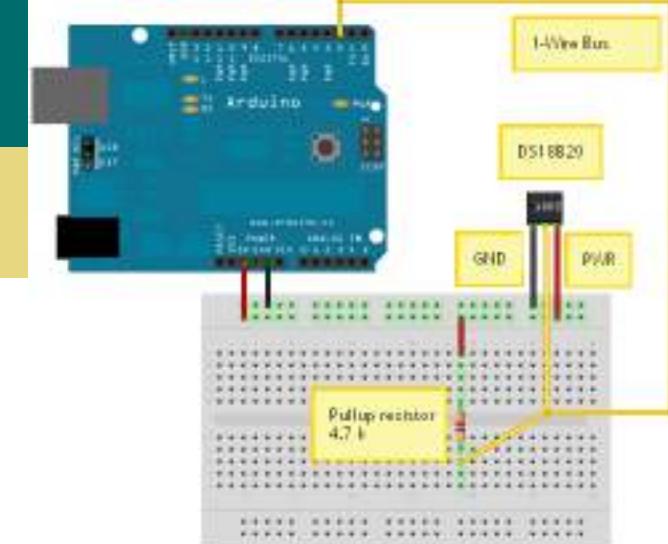


# 1-Wire



## Example: DS18S20 (1/3)

```
#include <OneWire.h>
/* DS18S20 Temperature chip i/o */
OneWire ds(10); // on pin 10
void setup(void) {
    Serial.begin(9600);
}
void loop(void) {
    byte i;
    byte present = 0;
    byte data[12];
    byte addr[8];
    if ( !ds.search(addr) ) {
        Serial.print("No more addresses.\n");
        ds.reset_search();
        delay(250);
        return;
    }
}
```



Ref: [https://www.pjrc.com/teensy/td\\_libs\\_OneWire.html](https://www.pjrc.com/teensy/td_libs_OneWire.html)

## Example: DS18S20 (2/3)

```
Serial.print("R=" );
for( i = 0; i < 8; i++) {
    Serial.print(addr[i], HEX);
    Serial.print(" ");
}
if ( OneWire::crc8( addr, 7) != addr[7]) {
    Serial.print("CRC is not valid!\n");
    return;
}
if ( addr[0] != 0x10) {
    Serial.print("Device is not a DS18S20 family device.\n");
    return;
}
```



## Example: DS18S20 (3/3)

```
// The DallasTemperature library can do all this work for you!
ds.reset();
ds.select(addr);
ds.write(0x44,1); // start conversion, with parasite power on at the end

delay(1000);      // maybe 750ms is enough, maybe not
// we might do a ds.depower() here, but the reset will take care of it.
present = ds.reset();
ds.select(addr);
ds.write(0xBE);      // Read Scratchpad

Serial.print("P=");
Serial.print(present,HEX);
Serial.print(" ");
for ( i = 0; i < 9; i++) { // we need 9 bytes
    data[i] = ds.read();
    Serial.print(data[i], HEX);
    Serial.print(" ");
}
Serial.print(" CRC=");
Serial.print( OneWire::crc8( data, 8 ), HEX );
Serial.println();
}
```

# Inter-Integrated Circuit Sound

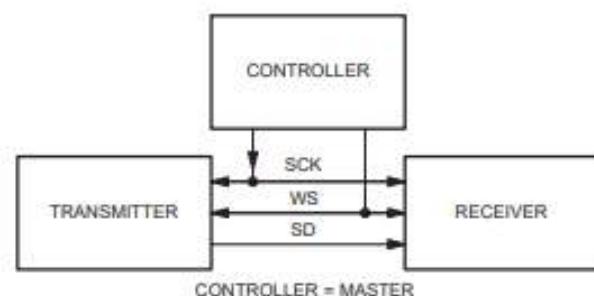
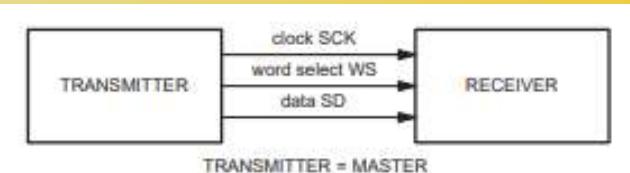
- I2S is used to connect digital audio devices. It is also an electrical bus interface standard.
- The I2S bus when used separates data and clock signals. Hence it results in very low jitter connection. I2S bus consists of following three lines:
  - a word-select line (WS)
  - a clock line (SCK) – (master clock line)
  - a multiplexed serial data line (SD)
- The word select line indicates the channel being transmitted:
  - For WS = 0 ► channel 1 (left)
  - For WS = 1 ► channel 2 (right)
- Serial data is transmitted in two's complement with the MSB first.

See More Examples:

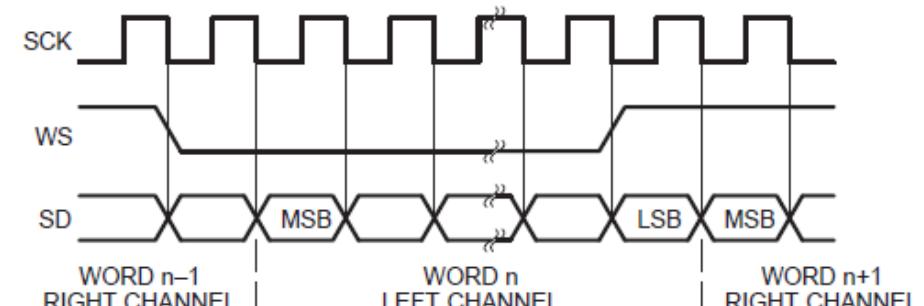
<https://diyi0t.com/i2s-sound-tutorial-for-esp32/>

<https://bitluni.net/esp32-i2s-camera-ov7670>

<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/peripherals/i2s.html>

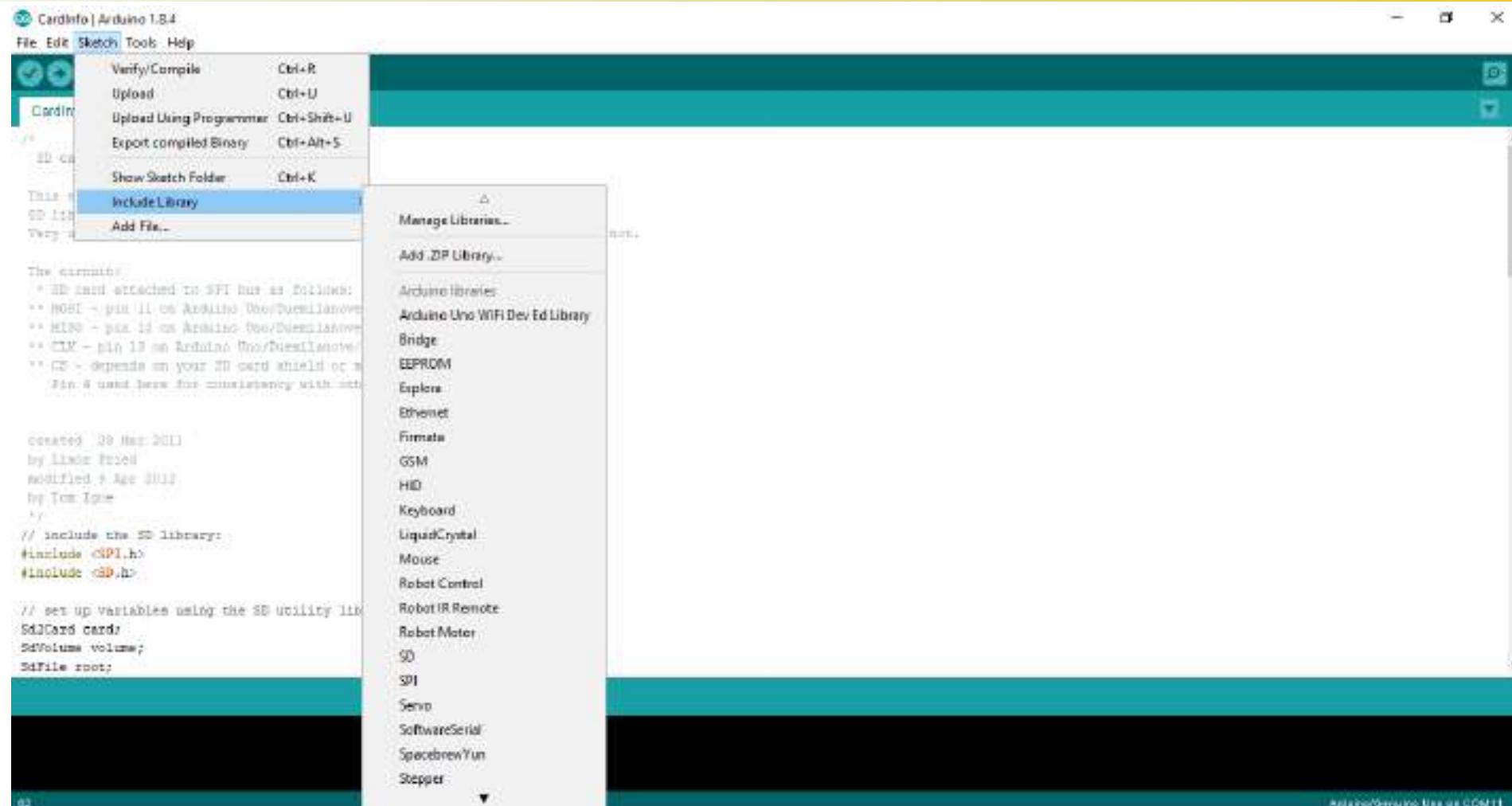


I2S configurations





# Additional Library Installation





## Serial Communications

### Arduino:

- <http://arduino.cc/en/reference/serial>
- <http://www.avrbeginners.net/architecture/twi/twi.html>
- <http://playground.arduino.cc/Main/InterfacingWithHardware>
- <https://www.arduino.cc/en/Tutorial/StringConstructors>

### Raspberry Pi:

- [http://elinux.org/RPi\\_Tutorials/](http://elinux.org/RPi_Tutorials/)
- [http://elinux.org/RPi\\_Expansion\\_Boards](http://elinux.org/RPi_Expansion_Boards)

### I<sup>2</sup>C: <https://www.arduino.cc/en/Reference/Wire>

- [http://www-us2.semiconductors.philips.com/acrobat/various/I2C\\_BUS\\_SPECIFICATION\\_1995.pdf](http://www-us2.semiconductors.philips.com/acrobat/various/I2C_BUS_SPECIFICATION_1995.pdf)
- <http://www.esacademy.com/faq/i2c/index.htm>
- <http://www.embedded.com/story/OEG20020528S0057>

### SPI: <https://www.arduino.cc/en/Reference/SPI>

- MC68HC11 manual
- <http://www.mct.net/faq/spi.html>
- <http://links.epanorama.net/links/serialbus.html>
- <http://www.embedded.com/story/OEG20020124S0116>

### Books:

[Mar12] Michael Margolis, **Arduino Cookbook**, O'Reilly, 2012.

[Box13] John Boxall, Arduino Workshop: A Hands-On Introduction with 65 Projects,  
No Starch Press, San Francisco, 2013.

[Mon14] Programming Arduino: Next Steps: Going Further with Sketches, McGrawHill 2014.



# **ITCS447**

## **Lecture 8**

### **Multitask Program**

### **Interrupt-FSM-PID-FREERTOS\***

Asst. Prof. Dr. Thitinan Tantidham



มหาวิทยาลัยมหิดล  
Mahidol University

ห้ามมิให้นักศึกษาทำข้า ดัดแปลง หรือใช้ประโยชน์จากการอันมีลิขสิทธิ์ปราภูอยู่ในระบบการศึกษาอิเล็กทรอนิกส์ (E-Learning) ของมหาวิทยาลัย ไม่ว่าจะทั้งหมดหรือบางส่วน โดยไม่ได้รับอนุญาตจากมหาวิทยาลัย นอกจากนี้จากการศึกษาส่วนบุคคล ทั้งนี้ การทำข้า ดัดแปลง หรือเผยแพร่ต่อสาธารณะชื่องานอันมีลิขสิทธิ์ จะมีโทษปรับตั้งแต่ 20,000 บาท ถึง 200,000 บาท และหากเป็นการกระทำเพื่อการค้า จะมีโทษจำคุกตั้งแต่ 6 เดือน ถึง 10 ปี หรือปรับตั้งแต่ 100,000 บาท ถึง 800,000 บาท หรือทั้งจำทั้งปรับ

Unless allowed by Mahidol University, the User shall not copy, modify, or exploit in part or in whole of the copyrighted materials on the Platform, other than for your own individual study. The copy, modify or communication to public of the copyrighted materials shall be inflicted with a fine from 20,000 Baht up to 200,000 Baht. If the offence is committed with the commercial purpose, the offender shall be inflicted with imprisonment for a term from 6 months up to 10 years or a fine from 100,000 Baht up to 800,000 Baht or both imprisonment and fine.



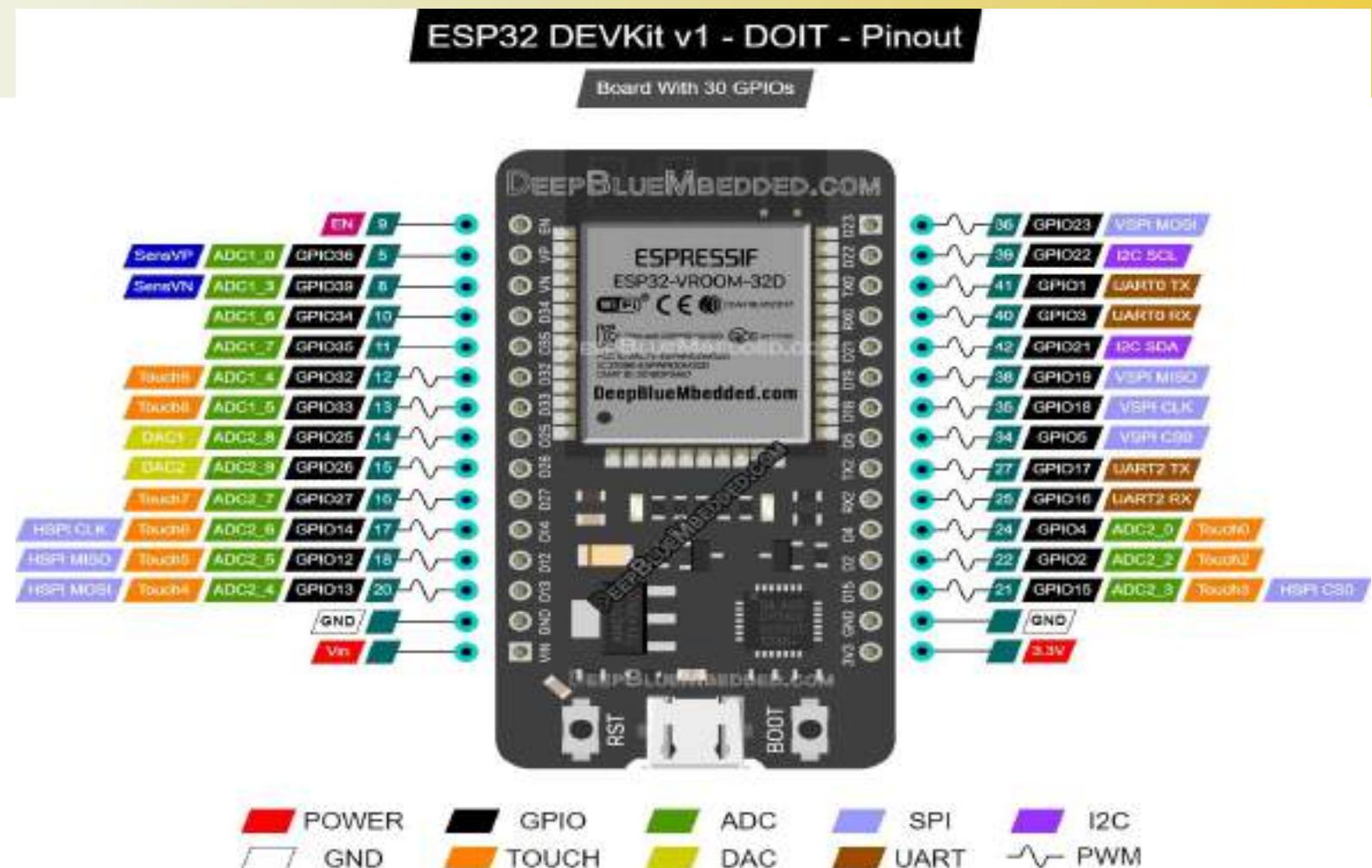
- Interrupt
- Finite State Machine
- Proportional Integral Derivative (PID)
- Dual Core on ESP32 (for two tasks)



# Interrupt

- Review: ESP32 Pin Layout
- What is Interrupt?
- ISR
- Interrupts in ESP32
- External Interrupts on Different Microcontrollers
- External Interrupt Programming:  
`attachInterrupt(GPIO, ISR, mode)`
- Examples

# Review: ESP32 Pin Layout



38 Pins including 3v3, En, 5v, 3xGNDs  
32 Pins for GPIOs



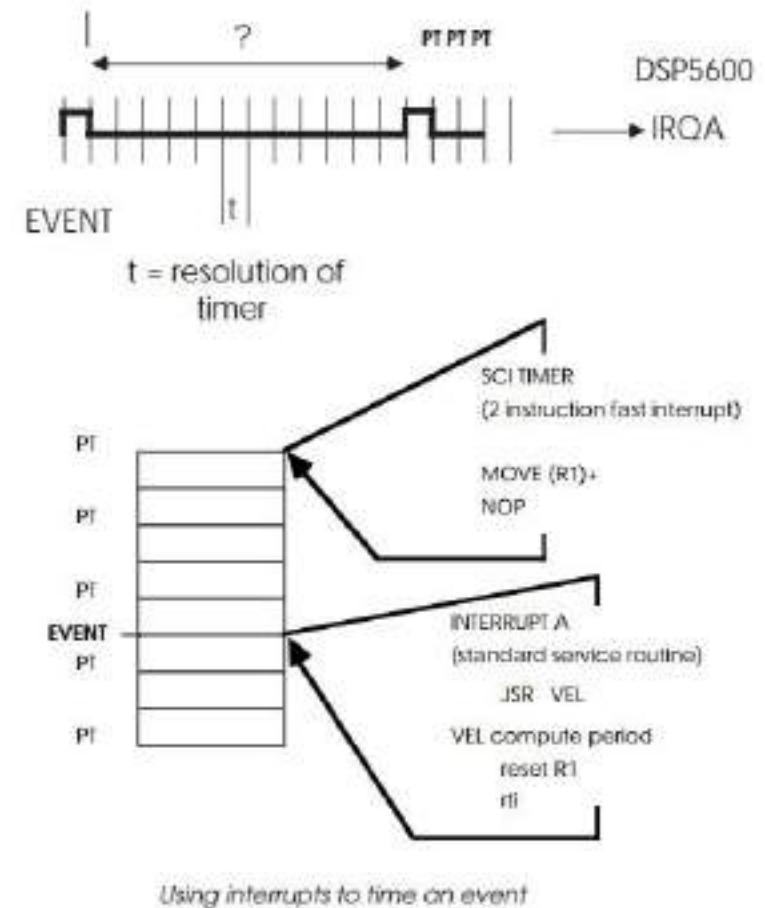
# What is Interrupt?

- An interrupt is a signal that interrupts the current processor activity.
- It may be triggered by an **external event** (change in pin state) or an **internal event** (a timer or a software signal).
- Once triggered, an interrupt pauses the current activity and causes the program to execute a different function.
  - This function is called an interrupt handler or an interrupt service routine (ISR) or Interrupt Handler.
- Once the function is completed, the program returns to what it was doing before the interrupt was triggered.

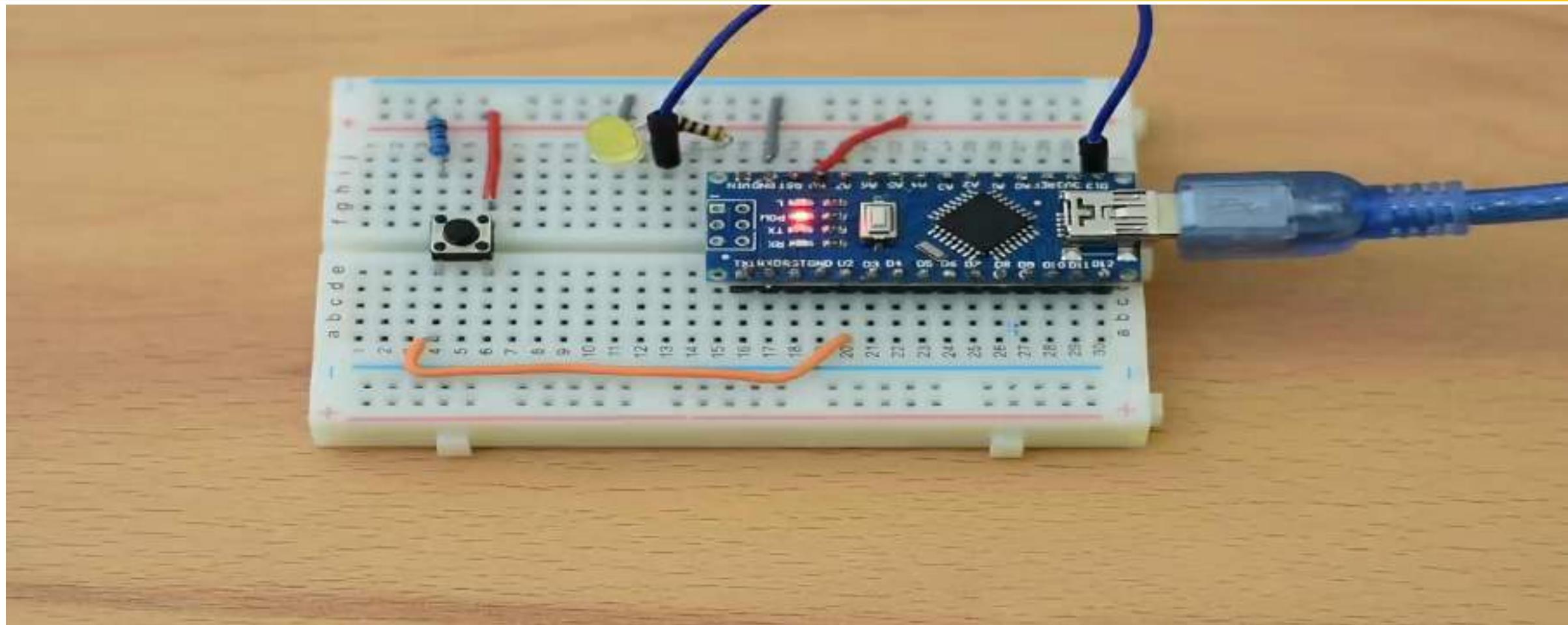
# Interrupt Service Routine (ISR)



- Interrupts are used to detect important real-time events, which occur during the normal code execution of the code, without continuously checking them, like pushing a button.
- ISR should be as short as possible and the return type of it is void.
- ISR should have the `IRAM_ATTR` attribute to declare that the compiled code will be placed in the Internal RAM (IRAM) in order to execute quickly as possible.
  - Some of normal functions do not work or behave differently in the ISR, for example, `delay()` function does not work in the ISR.
  - `delay()` is a blocking function. Blocking functions prevent a program from doing anything else until that particular task is completed. If you need multiple tasks to occur at the same time, you cannot use `delay()`, but uses `millis()` function instead
  - Variables used in the ISR must be volatile variables.



# Interrupts





- The ESP32 offers up to 32 interrupt slots for each core.
- Each interrupt has a certain priority level and can be categorized into two types:
  - **Hardware Interrupts**
    - These occur in response to an external event by hardware peripherals or sources, external GPIO pins.
    - For example, GPIO Interrupt(when a key is pressed down) or a Touch Interrupt(when touch is detected)
  - **Software Interrupts**
    - These occur in response to a software instruction by user or programmer
    - For example, a Simple Timer Interrupt or WatchDog Timer (WDT) Interrupt(when timer times out)

# External Interrupts



## Microcontrollers

| Arduino Boards                    | Digital Pins Usable For Interrupts                                               |
|-----------------------------------|----------------------------------------------------------------------------------|
| Uno, Nano, Mini, other 328-based  | 2, 3                                                                             |
| Uno WiFi Rev.2, Nano Every        | all digital pins                                                                 |
| Mega, Mega2560, MegaADK           | 2, 3, 18, 19, 20, 21                                                             |
| Micro, Leonardo, other 32u4-based | 0, 1, 2, 3, 7                                                                    |
| Zero                              | all digital pins, except 4                                                       |
| MKR Family boards                 | 0, 1, 4, 5, 6, 7, 8, 9, A1, A2                                                   |
| Nano 33 IoT                       | 2, 3, 9, 10, 11, 13, 15, A5, A7                                                  |
| Nano 33 BLE, Nano 33 BLE Sense    | all pins                                                                         |
| Due                               | all digital pins                                                                 |
| 101                               | all digital pins (Only pins 2, 5, 7, 8, 10, 11, 12, 13 work with <b>CHANGE</b> ) |

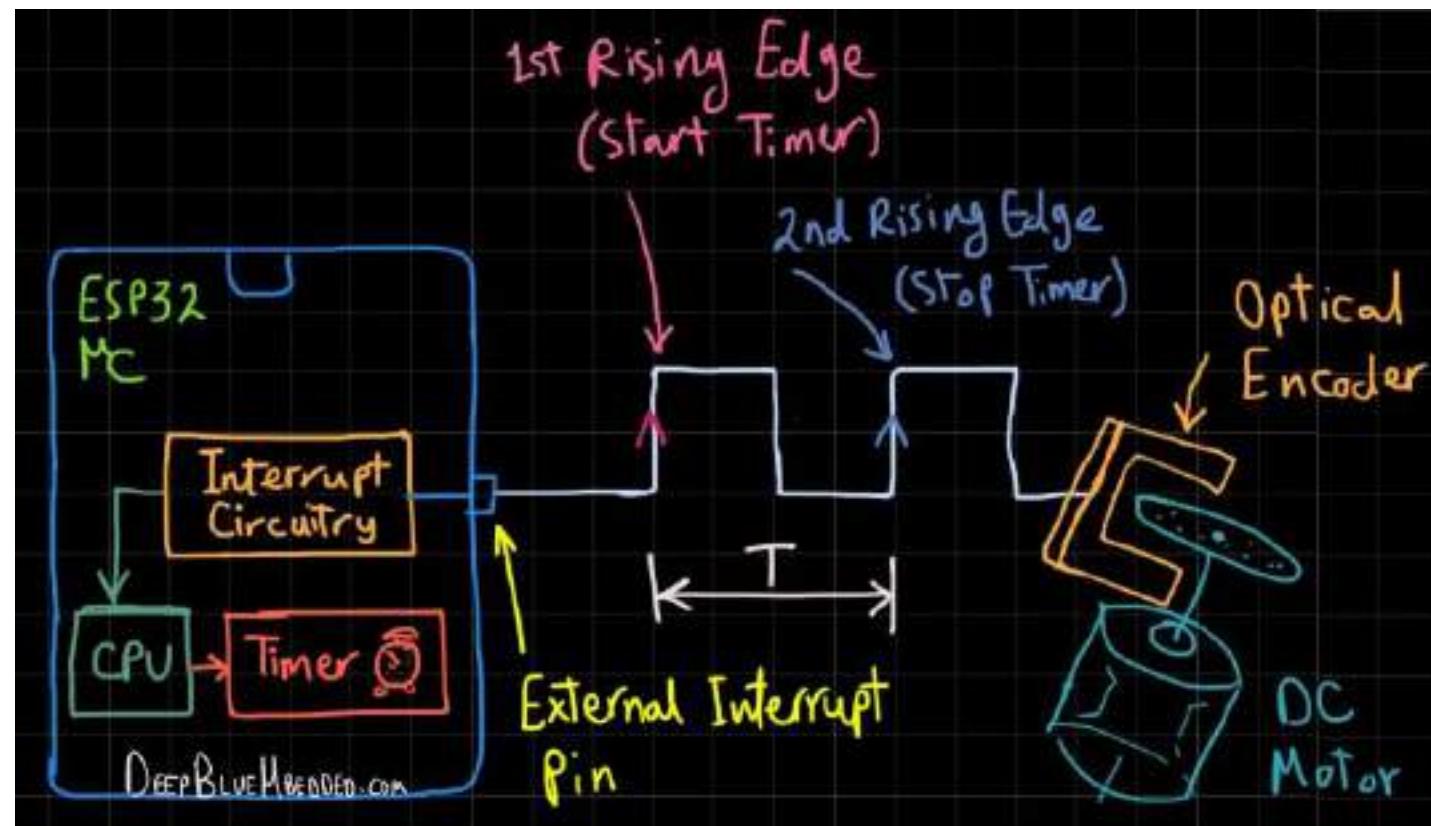


<https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/>

# External Interrupts

## Example

- As you can see in the diagram, the motor rotation causes the optical encoder to generate a square wave signal. By reading (measuring) the frequency of this signal, we can deduce the motor's speed (RPM).
- The digital signal is being measured in this application example by using an external interrupt pin + a Timer module.
- On the first rising edge, an interrupt occurs, so the CPU suspends the main program execution and starts a timer, then it resumes back the main program.
- On the next rising edge, an interrupt is fired, the CPU suspends the main program and stops the timer.
- Now, the timer count can tell us the total time ( $T$ ) or period of the signal. To get the frequency we'll do ( $F = 1/T$ ) and we're done.



<https://deepbluembedded.com/esp32-external-interrupts-pins-arduino-examples/>  
<https://lastminuteengineers.com/handling-esp32-gpio-interrupts-tutorial/>

# External Interrupt Programming



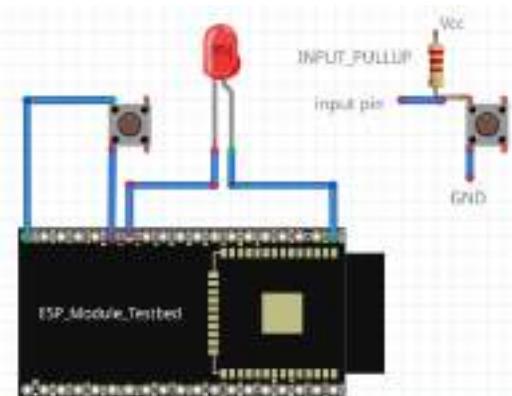
```
digitalPinToInterrupt(GPIO)
attachInterrupt(GPIO, ISR, mode)
```

- GPIO: the number of a pin number where the interrupt signal generating device will be attached
- ISR – the name of a function of interrupt service routine,
- Mode - defines when interrupt signal is triggered. There are five basic mode values:
  - LOW - interrupt is triggered when the pin value is LOW
  - HIGH - interrupt is triggered when the pin value is HIGH,
  - CHANGE - interrupt is triggered when the pin value is changed,
  - RISING - interrupt is triggered when the pin value is changed from LOW to HIGH.
  - FALLING – interrupt is triggered when the pin value is changed from HIGH to LOW
- detachInterrupt (GPIO) :  
It is an **option** when there is no longer to monitor GPIO

# Example 1

```
byte ledPin = 14;
byte interruptPin = 12;          /* pin that is attached to interrupt */
volatile byte state = LOW;      /* hold the state of LED when toggling */
void setup() {
    pinMode(ledPin, OUTPUT);
    /* set the interrupt pin as input pullup */
    pinMode(interruptPin, INPUT_PULLUP);
    /* attach interrupt to the pin function blink will be invoked when interrupt
       occurs interrupt occurs whenever the pin change value */
    attachInterrupt(digitalPinToInterrupt(interruptPin), blink, CHANGE);
}

void loop() {
}
/* interrupt function toggle the LED */
void blink()
{
    state = !state;
    digitalWrite(ledPin, state);
}
```



# Example 2: Button Input (1/4)



## Interrupt Handler for Button Input

```
int button_interrupt = 0;      // Interrupt 0 is on pin 2 !!
int toggle_on = false;        // Button click switches state

void setup() {
    Serial.begin(9600);
    attachInterrupt(button_interrupt, handle_click, RISING); // Register handler
}

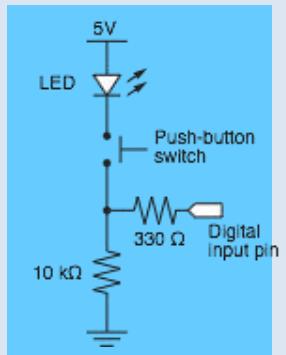
void loop() {
    if ( toggle_on ) {
        Serial.println("on");
    } else {
        Serial.println("off");
    }
}

void handle_click() {

    static unsigned long last_interrupt_time = 0;          // Zero only at start
    unsigned long interrupt_time = millis();                // Read the clock

    if ( interrupt_time - last_interrupt_time > 200 ) { // Ignore when < 200 msec
        toggle_on = !toggle_on;
    }

    last_interrupt_time = interrupt_time;
}
```



# Example 2: Button Input (2/4)



## Interrupt Handler for Button Input

```
int button_interrupt = 0;      // Interrupt 0 is on pin 2 !!
int toggle_on = false;         // Button click switches state

void setup() {
    Serial.begin(9600);
    attachInterrupt( button_interrupt, handle_click, RISING); // Register handler
}

void loop() {
    if ( toggle_on )
        Serial.println("on");
    else {
        Serial.println("off");
    }
}

void handle_click() {

    static unsigned long last_interrupt_time = 0;           // Zero only at start
    unsigned long interrupt_time = millis();                // Read the clock

    if ( interrupt_time - last_interrupt_time > 200 ) {   // Ignore when < 200 msec
        toggle_on = !toggle_on;
    }

    last_interrupt_time = interrupt_time;
}
```

button\_interrupt is the ID or number of the interrupt. It must be 0 or 1

Interrupt handler must be registered when program starts

A RISING interrupt occurs when the pin changes from LOW to HIGH

The interrupt handler, handle\_click, is a user-written function that is called when an interrupt is detected

# Example 2: Button Input (3/4)



## Interrupt Handler for Button Input

```
int button_interrupt = 0;      // Interrupt 0 is on pin 2 !!
int toggle_on = false;        // Button click switches state

void setup() {
    Serial.begin(9600);
    attachInterrupt(button_interrupt, handle_click, RISING); // Register handler
}

void loop() {
    if (toggle_on) {
        Serial.println("on");
    } else {
        Serial.println("off");
    }
}

void handle_click() {

    static unsigned long last_interrupt_time = 0;          // Zero only at start
    unsigned long interrupt_time = millis();               // Read the clock

    if (interrupt_time - last_interrupt_time > 200) {   // Ignore when < 200 msec
        toggle_on = !toggle_on;
    }

    last_interrupt_time = interrupt_time;
}
```

toggle\_on is a global variable that remembers the “state”. It is either true or false (1 or 0).

The loop() function only checks the state of toggle\_on. The value of toggle\_on is set in the interrupt handler, handle\_click.

The value of toggle\_on is flipped only when a *true* interrupt even occurs. De-bouncing is described in the next slide.

# Example 2: Button Input (4/4)



## Interrupt Handler for Button Input

```
int button_interrupt = 0;      // Interrupt 0 is on pin 2 !!
int toggle_on = false;        // Button click switches state

void setup() {
  Serial.begin(9600);
  attachInterrupt( button_interrupt, handle_click, RISING); // Register handler
}

void loop() {
  if ( toggle_on ) {
    Serial.println("on");
  } else {
    Serial.println("off");
  }
}

void handle_click() {
  static unsigned long last_interrupt_time = 0;
  unsigned long interrupt_time = millis();

  if ( interrupt_time - last_interrupt_time > 200 ) { // Ignore when < 200 msec
    toggle_on = !toggle_on;
  }

  last_interrupt_time = interrupt_time;
}
```

Value of a *static* variable is always retained

Use *long*: the time value in milliseconds  
can become large

Clock time when current interrupt occurs

Ignore events that occur in less than 200 msec from  
each other. These are likely to be mechanical bounces.

Save current time as the new “last” time



## Example 3: 2 ISRs

```
void setup() {  
    pinMode(32, OUTPUT);  
    pinMode(33, OUTPUT);  
    pinMode(35, INPUT);  
    pinMode(34, INPUT);  
    attachInterrupt(35,pin35_ISR,CHANGE);  
    attachInterrupt(34,pin34_ISR,CHANGE);  
}  
void loop() {  
    digitalWrite(33,1);delay(100);  
    digitalWrite(33,0);delay(100);  
}  
void pin35_ISR()  
{  
    digitalWrite(32, HIGH);  
}  
void pin34_ISR()  
{  
    digitalWrite(32, LOW);  
}
```



## Example 4: IRAM\_ATTR (1)

```
struct Button {
    const uint8_t PIN;
    uint32_t numberKeyPresses;
    bool pressed;
};

Button button1 = {18, 0, false};

void IRAM_ATTR isr() {
    button1.numberKeyPresses += 1;
    button1.pressed = true;
}

void setup() {
    Serial.begin(115200);
    pinMode(button1.PIN, INPUT_PULLUP);
    attachInterrupt(button1.PIN, isr, FALLING);
}

void loop() {
    if (button1.pressed) {
        Serial.printf("Button 1 has been pressed %u times\n", button1.numberKeyPresses);
        button1.pressed = false;
    }

    //Detach Interrupt after 1 Minute
    static uint32_t lastMillis = 0;
    if (millis() - lastMillis > 60000)
    { lastMillis = millis();
        detachInterrupt(button1.PIN);
        Serial.println("Interrupt Detached!");
    }
}
```



# Example 5: IRAM\_ATTR (2)

## Timer Interrupts

```
volatile int interrupts;
int totalInterrupts;

hw_timer_t * timer = NULL;
portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;

void setup() {
    Serial.begin(115200);

    // Configure Prescaler to 80, as our timer runs @ 80Mhz
    // Giving an output of 80,000,000 / 80 = 1,000,000 ticks / second
    timer = timerBegin(0, 80, true);
    timerAttachInterrupt(timer, &onTime, true);
    // Fire Interrupt every 1m ticks, so 1s
    timerAlarmWrite(timer, 1000000, true);
    timerAlarmEnable(timer);
}

void IRAM_ATTR onTime() {
    portENTER_CRITICAL_ISR(&timerMux);
    interrupts++;
    portEXIT_CRITICAL_ISR(&timerMux);
}

void loop() {
    if (interrupts > 0) {
        portENTER_CRITICAL(&timerMux);
        interrupts--;
        portEXIT_CRITICAL(&timerMux);
        totalInterrupts++;
        Serial.print("totalInterrupts");
        Serial.println(totalInterrupts);
    }
}
```

# Example 7: Software Interrupt



# ESP32: Deep Sleep → TimeWakeup

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** TimerWakeUp | Arduino 1.8.13
- File Menu:** File Edit Search Tools Help
- Sketchbook:** Examples
- Code Area:** The code for the TimerWakeUp example is displayed. It includes a call to `esp_deep_sleep_start()` and a `Serial.println("This is a test")` line.
- Serial Monitor:** Shows the output: "This is a test".
- Context Menu (over DeepSleep example):** A context menu is open over the line `esp_deep_sleep_start()`. The menu items are: ExternalWakeUp, TimerWakeUp, and TouchWakeUp.
- Example List:** A list of examples for the ESP32 board is shown, including:
  - 01.Basics
  - 02.Digital
  - 03.Analog
  - 04.Communication
  - 05.Control
  - 06.Sensors
  - 07.Display
  - 08.Strings
  - 09.USB
  - 10.StarterKit\_BasicKit
  - 11.ArduinoISP
- Board Manager:** A list of boards is shown, with ESP32 selected:
  - Arduino054
  - BluetoothSerial
  - DNSServer
  - EEPROM
  - ESP32
  - ESP32 Arduino UDP
  - ESP32 Azure IoT Arduino
  - ESP32 BLE Arduino
- Bottom Status Bar:** Shows battery level at 100%, signal strength, and the date/time (8:53 AM, 9/10/2020).

ESP32 provides 5 Modes for Deep Sleep:  
Active, Modem-sleep, Light-sleep, Deep-sleep, Hibernation

| Power mode                     | Active                    | Modem-sleep | Light-sleep | Deep-sleep                   | Hibernation |
|--------------------------------|---------------------------|-------------|-------------|------------------------------|-------------|
| Sleep pattern                  | Association sleep pattern |             |             | ULP sensor-monitored pattern | -           |
| CPU                            | ON                        | ON          | PAUSE       | OFF                          | OFF         |
| Wi-Fi/BT baseband and radio    | ON                        | OFF         | OFF         | OFF                          | OFF         |
| RTC memory and RTC peripherals | ON                        | ON          | ON          | ON                           | OFF         |
| ULP co-processor               | ON                        | ON          | ON          | ON/OFF                       | OFF         |

## Example 8: Software Interrupt



# ESP32: Deep Sleep → External Wakeup

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** TimerWakeUp | Arduino 1.8.13
- Menu Bar:** File Edit Sketch Tools Help
- File Menu:** New Ctrl+N, Open Ctrl+O, Open Recent, Sketchbook, Examples, Close Ctrl+W, Save Ctrl+S, Save As Ctrl+Shift+S, Page Setup Ctrl+Shift+P, Print Ctrl+P, Preferences Ctrl+Comma, Quit Ctrl+Q.
- Code Area:** The code is for "External Wakeup from Deep Sleep". It includes a header file "DeepSleep.h" and a main loop that prints "This is not a good time" and then enters deep sleep. A note at the bottom says "In the next line we always have to wait, so power consumption is low".
- Sketchbook Area:** Shows examples for ESP32 Dev Module like ArduinoOTA, BluetoothSerial, DNSServer, EEPROM, and ESP32.
- Tools Menu:** Shows options for AnalogOut, Camera, ChipID, DeepSleep (selected), ExternalWakeUp, FreeRTOS, GPIO, HallSensor, I2S, ResetReason, SPI, Timer, Touch, and WiFi.
- Board & Port:** Board: Arduino Uno, Port: COM3, Speed: 115200.
- Breadboard Diagram:** A breadboard diagram showing an Arduino Uno connected to a breadboard. Pin 33 (labeled "GPIO 33") is connected to a digital input pin on the breadboard through a pull-down resistor. A red wire connects the Arduino's GND to the breadboard's GND.
- Bottom Taskbar:** Icons for browser, interrupt, Excel, IC54462, Cisco WebEx, Introduction, tutorial, ESP32 Dev, input\_swit..., External..., TimerWake..., 100%, and battery level.

Lecture 8: Interrupt-FSM-PID



[Mar11] Michael Margolis, **Arduino Cookbook**, O'Reilly, 2011.

[OK13] Robert Oshana, Mark Kraeling, **Software Engineering for Embedded Systems**, e-ISBN: 978-0-12-415941-9, Newnes, 2013.

[Whi11] Elecia White, **Making Embedded Systems**, O'Reilly, 2011.

[Kol17] Neil Kolban, Kolban's Book on ESP32, May 2017.

[SS1.4] Rui Santos and Sara Santos, Learn ESP32 with Arduino IDE v1.4.

## Arduino Tutorial

- ✓ <http://www.ladyada.net/learn/arduino/lesson5.html>

## Using interrupts

- ✓ <http://www.uchobby.com/index.php/2007/11/24/arduino-interrupts/>
- ✓ <https://lastminuteengineers.com/handling-esp32-gpio-interrupts-tutorial/>
- ✓ <http://www.arduino.cc/en/Reference/AttachInterrupt>
- ✓ <https://randomnerdtutorials.com/esp32-deep-sleep-arduino-ide-wake-up-sources/>



# **Finite State Machine**

## **FSM**

Asst. Prof. Dr. Thitinan Tantidham

- Logic control is an essential part to develop an intelligence device
- Logic control can be developed by
  - Truth table
    - You only need to know the corresponding input/output relation
    - As there is **no memory**, only simple operations can be achieved
  - Finite State Machine (FSM) or State Machine
    - Decision is based on what it has done i.e. the system has memory (history). Therefore the performance can be more complex.
    - A sequential system contains state stored in memory elements internal to the system.
    - Its behavior depends both on the set of inputs supplied and on the contents of the internal memory, or state of the system.
    - Thus, the sequential system cannot be described with a truth table.

# ESP32: Arduino IDE Programming



- Arduino ESP32 core is built over FreeRTOS. If you look into the `cores/esp32` folder, you will see a file called “`main.cpp`”.
  - Here you can see our Arduino program is just a task of FreeRTOS.
  - When the task is invoked it will invoke “`setup()`” function and then invoke the “`loop()`” in an infinite loop.
- Suppose that we implement an application that have 2 jobs:
  - + job A to scan the input from keyboard.
  - + job B to send a message and wait for response.
  - If job B takes long time to finish, it will block job A.
  - So job A may miss some keyboard pressing event. It is not a good design

```
void loop(){
    jobA_scan_the_keyboard();
    jobB_send_message();
    while(1){
        if( jobB_get_response() ){
            break;
        }
    }
}
```

<http://www.iotsharing.com/2017/05/how-to-apply-finite-state-machine-to-arduino-esp32-avoid-blocking.html>

# Finite State Machines (FSM)



- The purpose of this method is to divide a machine into **a collection of states**.
- At a specific time, **only one state is active** and in each state the machine just executes a specific action.
- After finishing that action, it may transit to another state to execute another action.
- For example:
  - we have a lighting system.
  - It can have 3 states: ON, OFF, IDLE.
  - Its normal state is IDLE and waiting for trigger to change state ON or OFF and then back to IDLE.

<https://arduining.com/2015/09/18/traffic-light-states-machine-with-arduino/>

<http://www.iotsharing.com/2017/05/how-to-apply-finite-state-machine-to-arduino-esp32-avoid-blocking.html>

## Switch/Case

- FSM can be implemented by **switch/case** structure.
- Here jobB is divided into 3 states: **SEND\_MESSAGE**, **WAIT\_FOR\_RESPONSE** and **IDLE**.
- jobB executes a specific action according to its state in every cycle and do not block jobA anymore.
- Each job has a chance to execute itself in every cycle, look like multitasking.

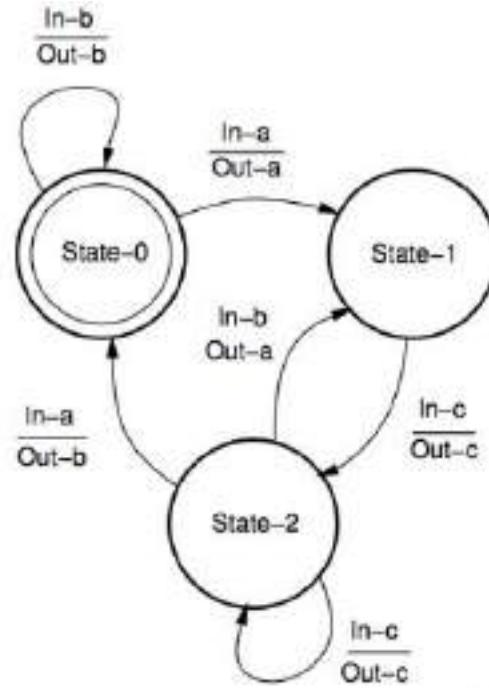
```
void loop(){
    jobA_scan_the_keyboard();
    switch(jobB_state)
    { case SEND_MESSAGE:
        jobB_send_message();
        jobB_state = WAIT_FOR_RESPONSE;
        break;
    case WAIT_FOR_RESPONSE:
        if(jobB_get_response()){
            jobB_state = IDLE;
            break;
        }
    case IDLE:
        if(need_send_message){
            jobB_state = SEND_MESSAGE;
            break;
        }
    case default:
        break;
    }
}
```

<http://www.iotsharing.com/2017/05/how-to-apply-finite-state-machine-to-arduino-esp32-avoid-blocking.html>

## How do we control a plant?

- Automation of the discrete operations (on-off) is largely a matter of a series of carefully timed on-off steps.
- Discrete Signals
  - Control parameters: true or false
  - Actuators: on or off
- Interlocks:
  - Output = function(input)
  - Boolean algebra
- Sequence Networks (Diagrams)
  - Dynamic systems
    - Output =  $f(\text{input}, \text{state})$
    - Next\_state =  $g(\text{input}, \text{state})$
  - FSM, Petri Nets, Grafcet/SFC (Sequential Function Chart), Grafchart

## Diagrams (1/2)

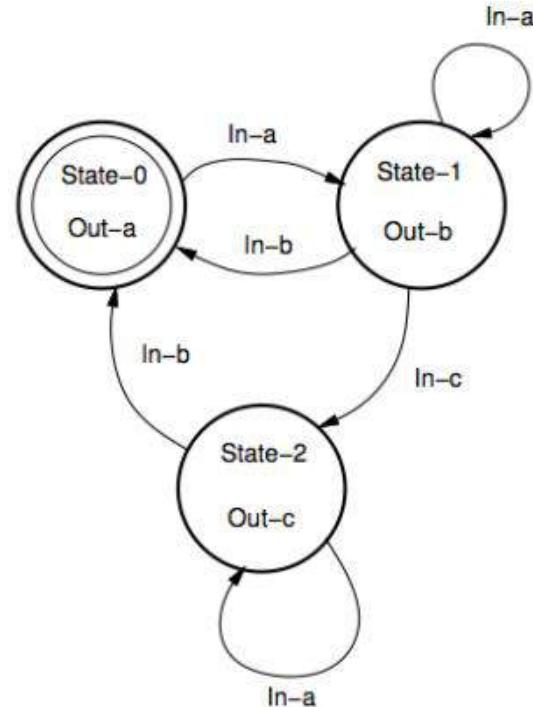


Mealy:

- Output events (actions) are associated with input events

Ref:

<http://www.archive.control.lth.se/media/Education/EngineeringProgram/FRTN20/2016/LectureSlides-02-2016.pdf>



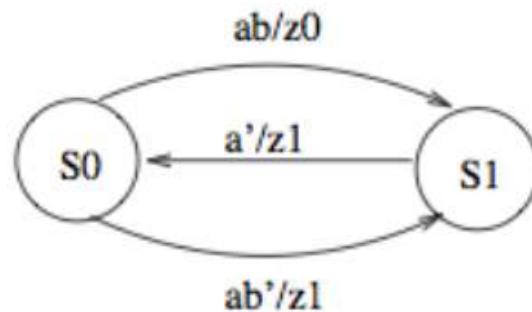
Moore:

- State transitions in response to input events
- Output events (actions) are associated with states.

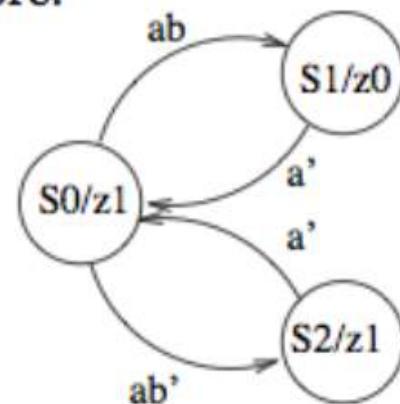


## Diagrams (2/2)

Mealy:



Moore:



Mealy:

- Output events (actions) are associated with input events

Moore:

- State transitions in response to input events
- Output events (actions) are associated with states.

Ref:

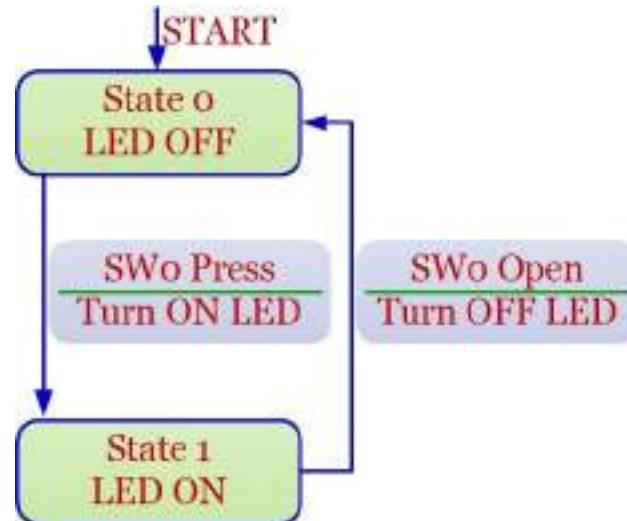
<http://www.archive.control.lth.se/media/Education/EngineeringProgram/FRTN20/2016/LectureSlides-02-2016.pdf>

# FSM Example 0



## Switch Off/On LED

```
#define LED0 15
#define SW0 32
byte State;
bool StateSW0;
void setup() {
    pinMode(LED0,OUTPUT);
    digitalWrite(LED0,LOW);
    pinMode(SW0,INPUT_PULLUP);
    Serial.begin(115200);
    Serial.println("...START...");
    State = 0;
}
void loop() {
    StateSW0 = digitalRead(SW0);
    switch (State)
    { case 0: //LED OFF
        if(StateSW0 == LOW)
        {   digitalWrite(LED0,HIGH);
            State = 1;
        }
        break;
    case 1: //LED ON
        if(StateSW0 == HIGH)
        {
            digitalWrite(LED0,LOW);
            State = 0;
        }
        break;
    }
}
```



Ref:

<https://phatiphatt.wordpress.com/esp32-finite-state-machine/>

<https://arduinoplusplus.wordpress.com/2019/07/06/finite-state-machine-programming-basics-part-1/>



# Finite State Machines

Introduction Video

# Finite State Machine

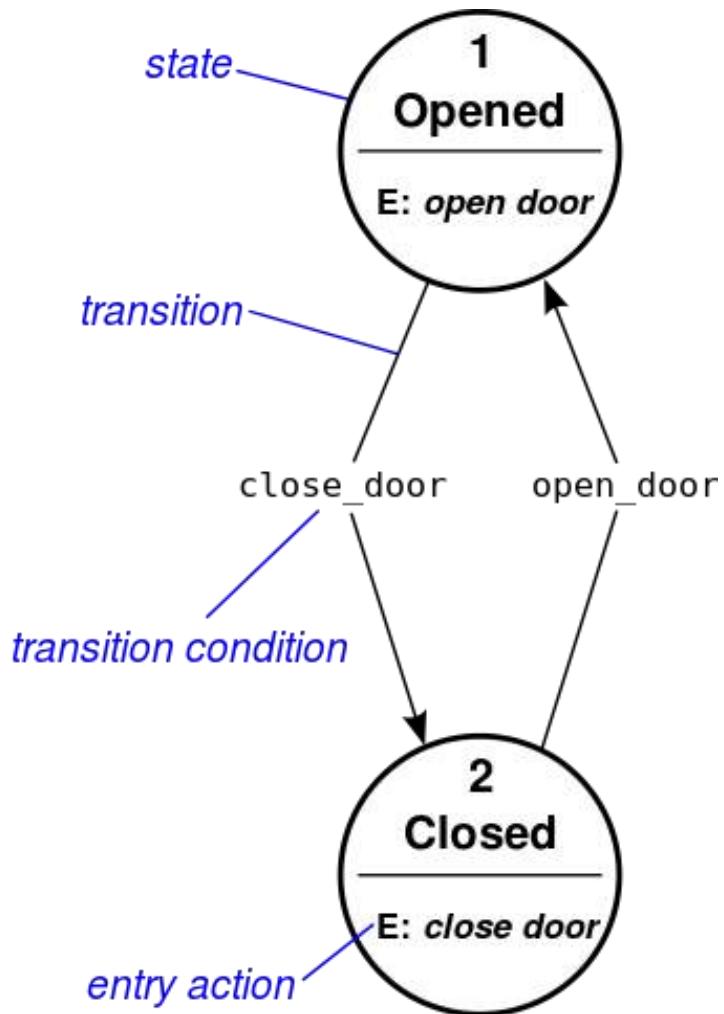


## Open/Close Door

Example of a door

- State
- Transition
- Transition condition
- Entry action

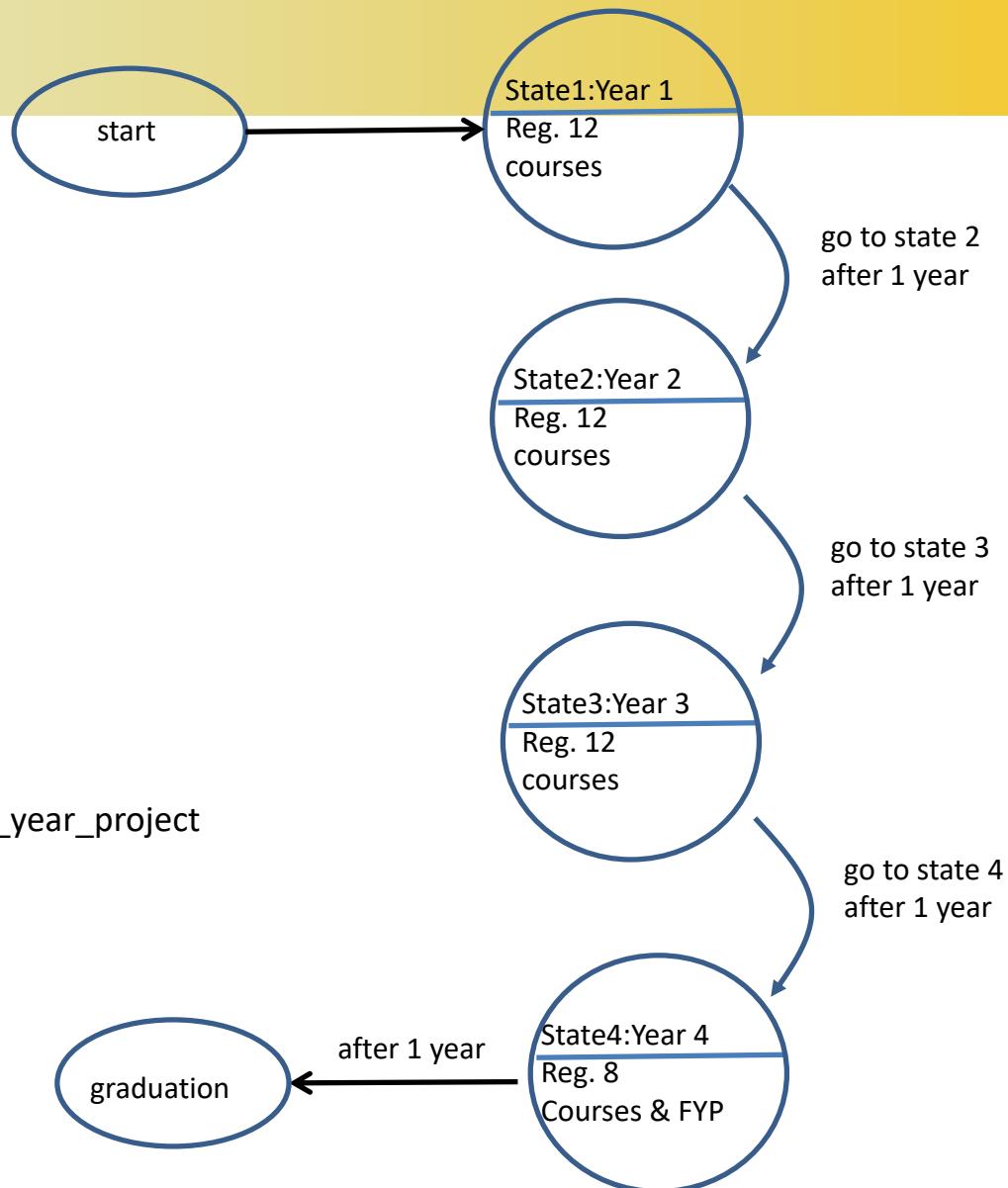
[http://en.wikipedia.org/wiki/State\\_diagram](http://en.wikipedia.org/wiki/State_diagram)



## Simple Study Life

State of a student at CUHK

- Start: go to state 1
- State 1=Year 1:
  - entry action: register 12 courses
  - Transition: go to state 2 after 1 year
- State 2=Year 2:
  - entry action: register 12 courses
  - Transition: go to state 3 after 1 year
- State 3=Year 3:
  - entry action: register 12 courses
  - Transition: go to state 4 after 1 year
- State 4=Year 4:
  - entry action: register 8 courses and FYP Final\_year\_project
  - Transition: go to stop after 1 year
- Stop: Graduation





## Life with Study Hard Condition

State of a student at CUHK

- Start: go to state1
- State 1=Year 1:
  - entry action: register 12 courses
  - Transition: if (study hard) promote to state2 (year2) else go back to state 1
- State 2=Year 2:
  - entry action: register 12 courses
  - Transition: if (study hard) promote to state3 (year3) else go back to state 2
- State 3=Year 3:
  - entry action: register 12 courses
  - Transition: if (study hard) promote to state4 (year4) else go back to state 3
- State 4=Year 4:
  - entry action: register 12 courses
  - Transition: if (study hard) promote to stop(graduation) else back go to state 4
- *Stop: Graduation*



We will have **three** examples here:

a. Simple finite state machine (no sensor).

e.g.: The dancing robot

b. An finite state machine uses 2 sensors.

e.g.: The robot that follows the magnetic strip

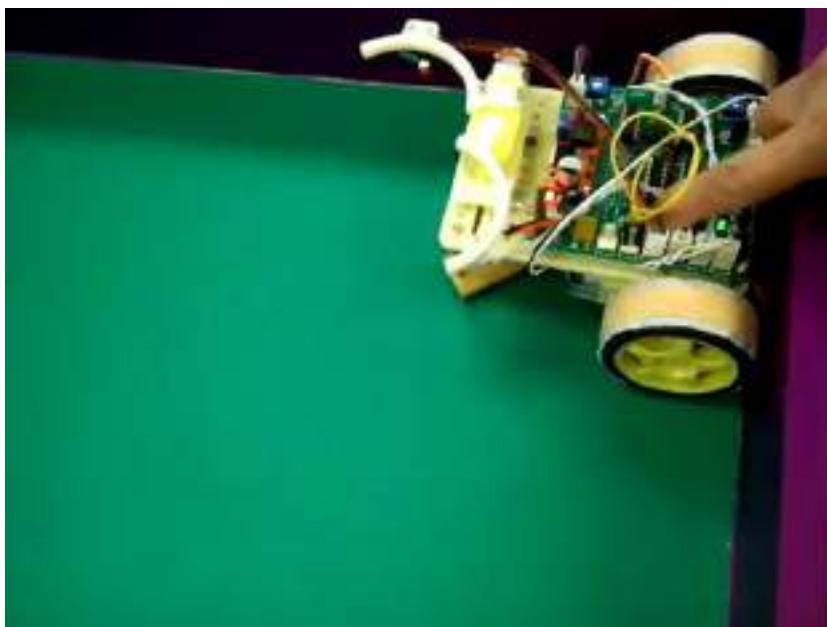
c. An finite state machine uses 3 sensors.

e.g.: The robot that follows the magnetic strip, and stops when it detects a magnet in front of the robot.



## No transition condition

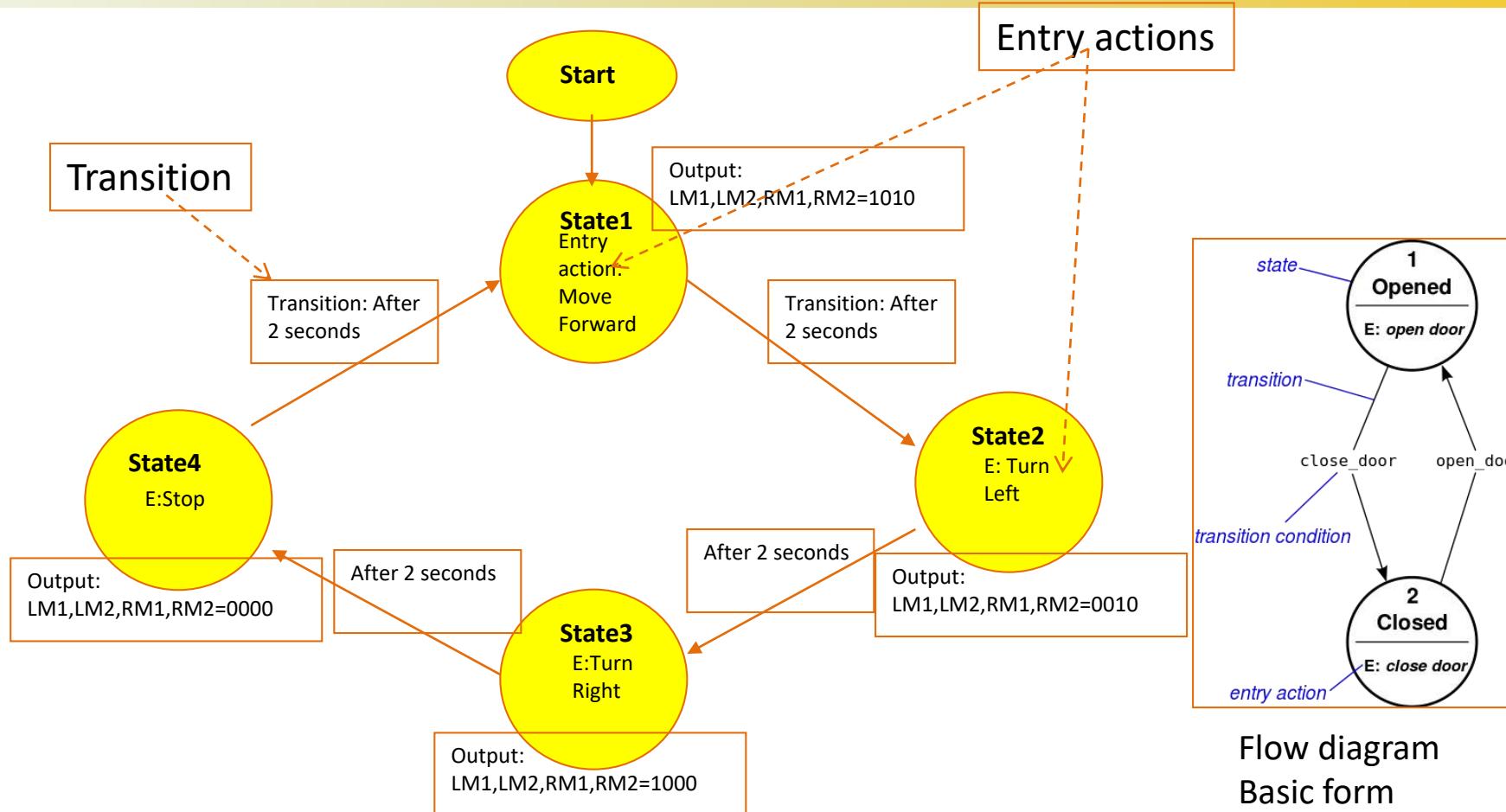
- The robot that dances with a pattern
  - Forward 2 seconds, turn left 2 seconds and turn right 2 seconds, stop and repeat the pattern again
- Video demo: <http://youtu.be/iyakbVyoafI>
- Using timing as a condition to change to another state



# FSM Example 3a: Simple FSM (2/3)



## No sensor input (no transition condition)



Flow diagram  
Basic form

# FSM Example 3a: Simple FSM (3/3)



## Implementation

Part of the sample source code is shown below:

```
switch(state)
{
    case STATE1:
        LM1=1;LM2=0;RM1=1;RM2=0;SPEED=200; //entry action
        DELAY_TIME=2000; // transition :delay 2 seconds
        motors(LM1,LM2,RM1,RM2,SPEED,SPEED,DELAY_TIME);
        state=STATE2; // next state will be state2
        break; //end of the current state
    case STATE2:
        LM1=0;LM2=0;RM1=1;RM2=0;SPEED=200;DELAY_TIME=2000; // delay 2 seconds
        motors(LM1,LM2,RM1,RM2,SPEED,SPEED,DELAY_TIME);
        state=STATE3; //next state will be state3
        break; //end of the current state
    .
    .
    .
}
```

Set motor to run forward with speed=200

Use of DELAY:

DELAY\_TIME=2000

motors(LM1,LM2,RM1,RM2,SPEED,  
SPEED,DELAY\_TIME);

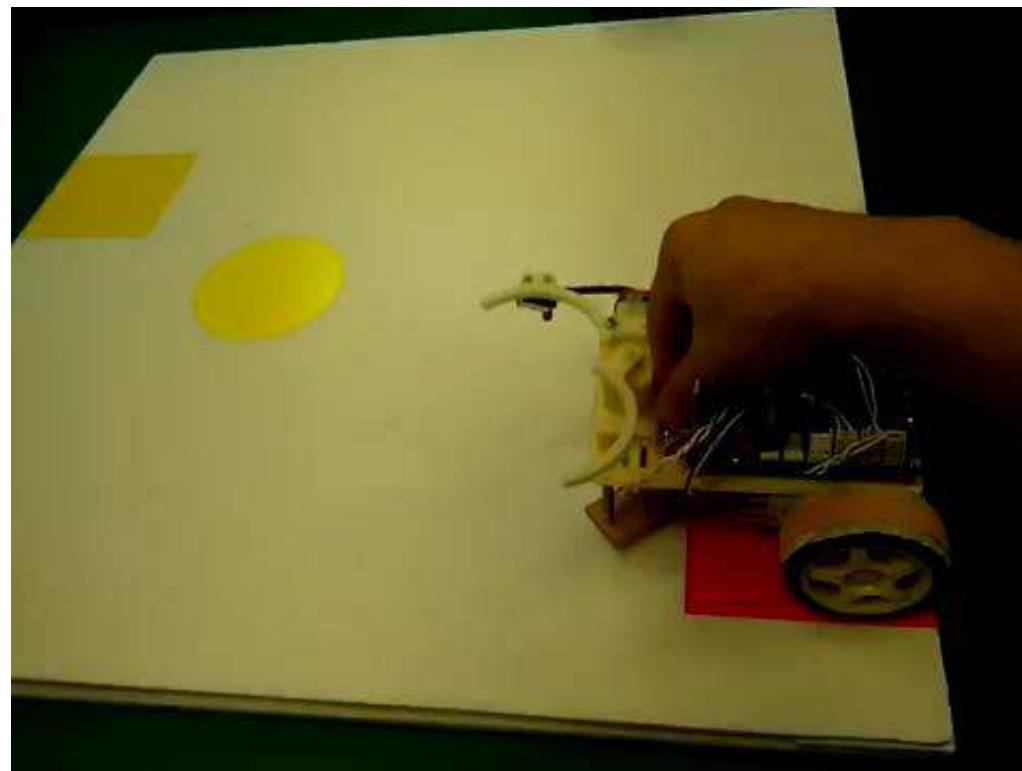
You may need to write  
the function motors( ) for your project

Exercise: explain the meaning of state 2

# FSM Example 3b: FSM with 2 Sensors (1/2)



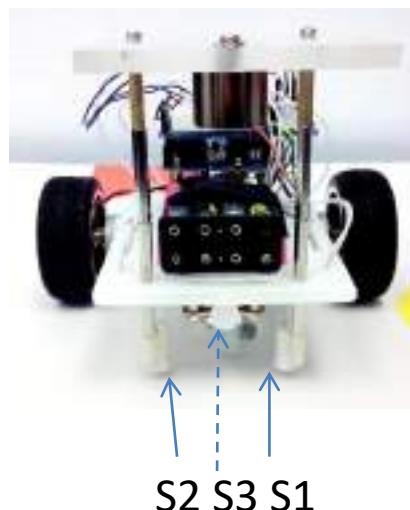
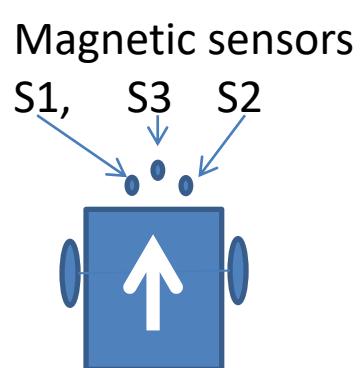
- The robot can follow the magnetic strip
- Video Demo: [http://youtu.be/NWHjWrq\\_VoY](http://youtu.be/NWHjWrq_VoY)



# FSM Example 3b: FSM with 2 Sensors (2/2)



- Sensors: S2, S1 are facing the ground to detect the magnetic strip
- S3 is facing the front, used to detect the target object
  - S3=1 if no object is detected
  - S3=0 if an object is detected

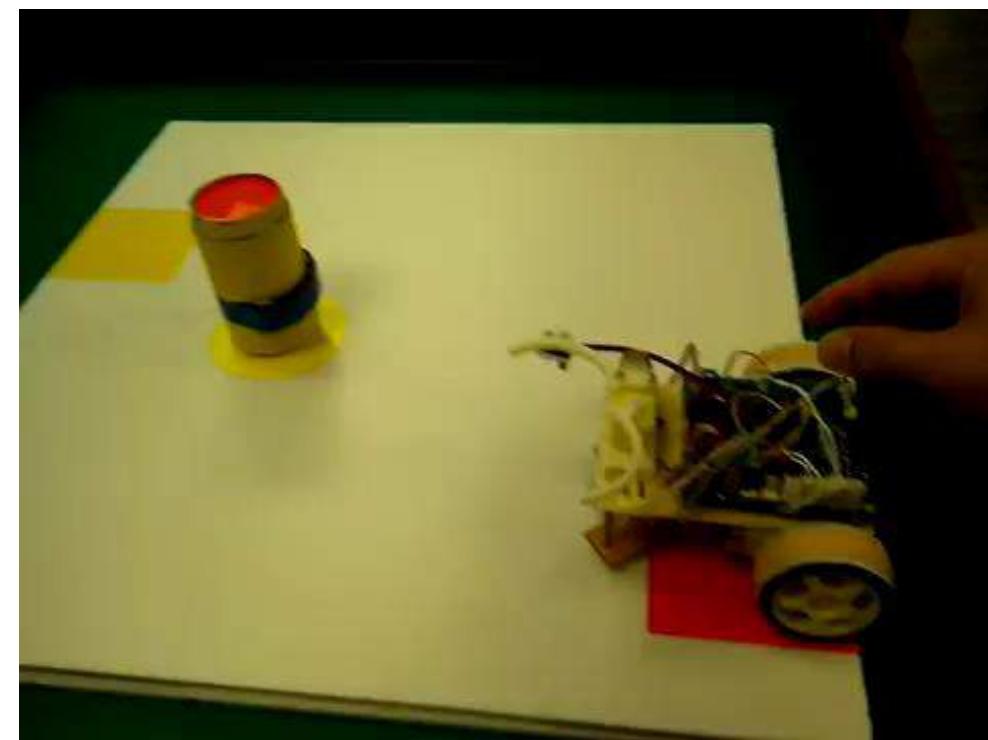
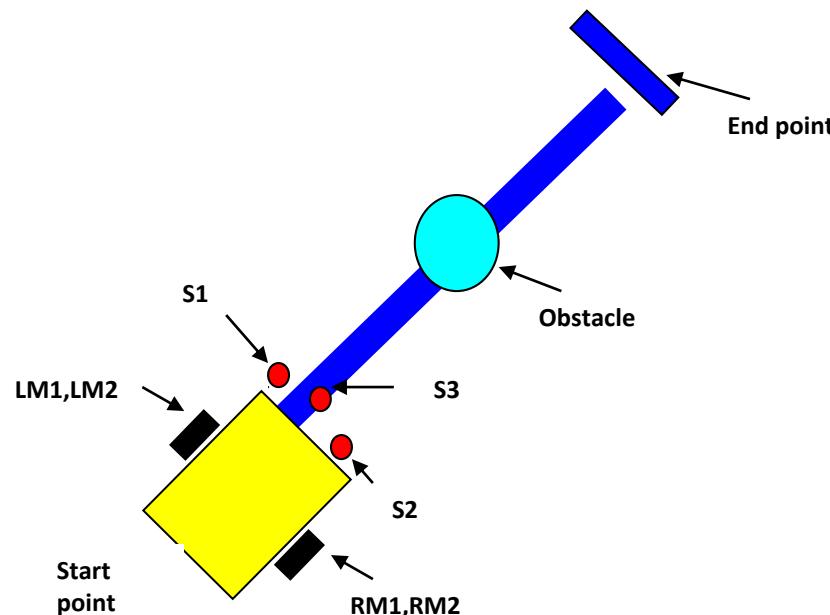


# FSM Example 3c: FSM with 3 Sensors



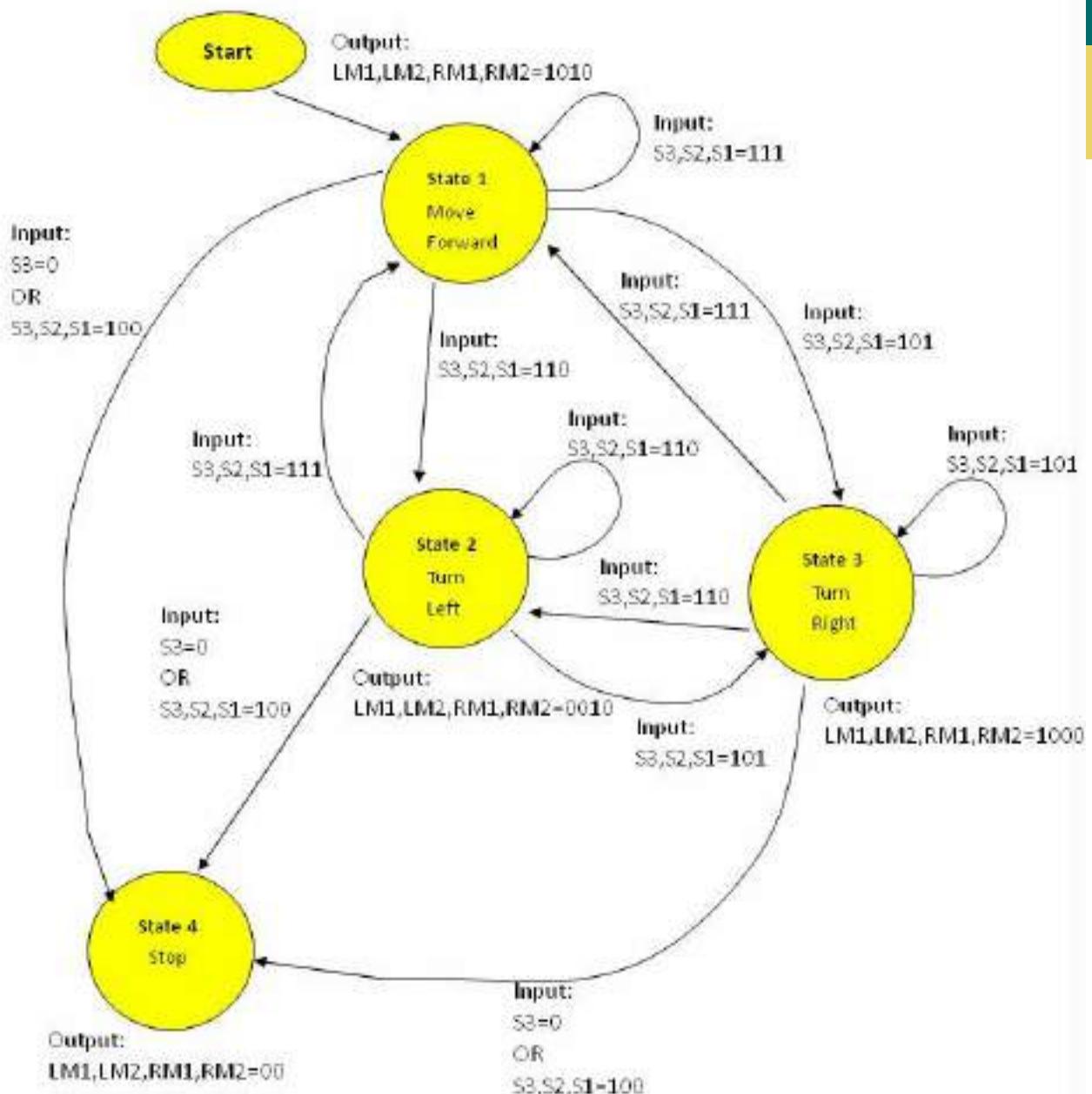
- Video Demo : <http://youtu.be/JEQkuax7IKE>

- The robot finds the CAN using the magnetic strip placed under the testing board and stops

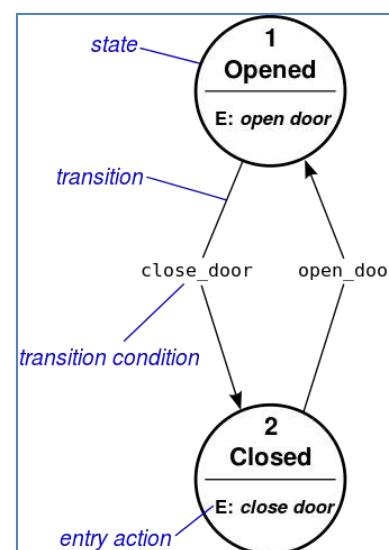




The state diagram of the Smart-car logic can be represented by the state diagram below.



Flow diagram  
Basic form





1/2

The sample source code (program\_segment3) is shown below:

```
switch(state)
{
    case STATE1: // forward for 1 second
        LM1=1;LM2=0;RM1=1;RM2=0;
        SPEED=200;DELAY_TIME=10;
        motors(LM1,LM2,RM1,RM2,SPEED,DELAY_TIME);
        //
        if ( S3()==1 && S2()==1 && S1()==0 ) state=STATE2;
        else if(S3()==1 && S2()==0 && S1()==1) state=STATE3;
        else if((S3==0) || (S3()==1 && S2()==0 && S1()==0)) state=STATE4;
        break;

    case STATE2: //robot turns left
        LM1=0;LM2=0;RM1=1;RM2=0;SPEED=200;DELAY_TIME=10;
        motors(LM1,LM2,RM1,RM2,SPEED,DELAY_TIME);
        //
        if ( S3()==1 && S2()==1 && S1()==1 ) state=STATE1; //back to state 1
        else if(S3()==1 && S2()==0 && S1()==1) state=STATE3;
        else if((S3==0) || (S3()==1 && S2()==0 && S1()==0)) state=STATE4;    break;
}
```

If S3=0, a CAN is detected, next state is state4

Move forward for 1 second

Robot deviated to the right, goto state 2

Robot deviated to the left goto state 3

2/2

```
case STATE3: //robot turns right
    // To be filled by students as an exercise
    .
    .
    .
    .

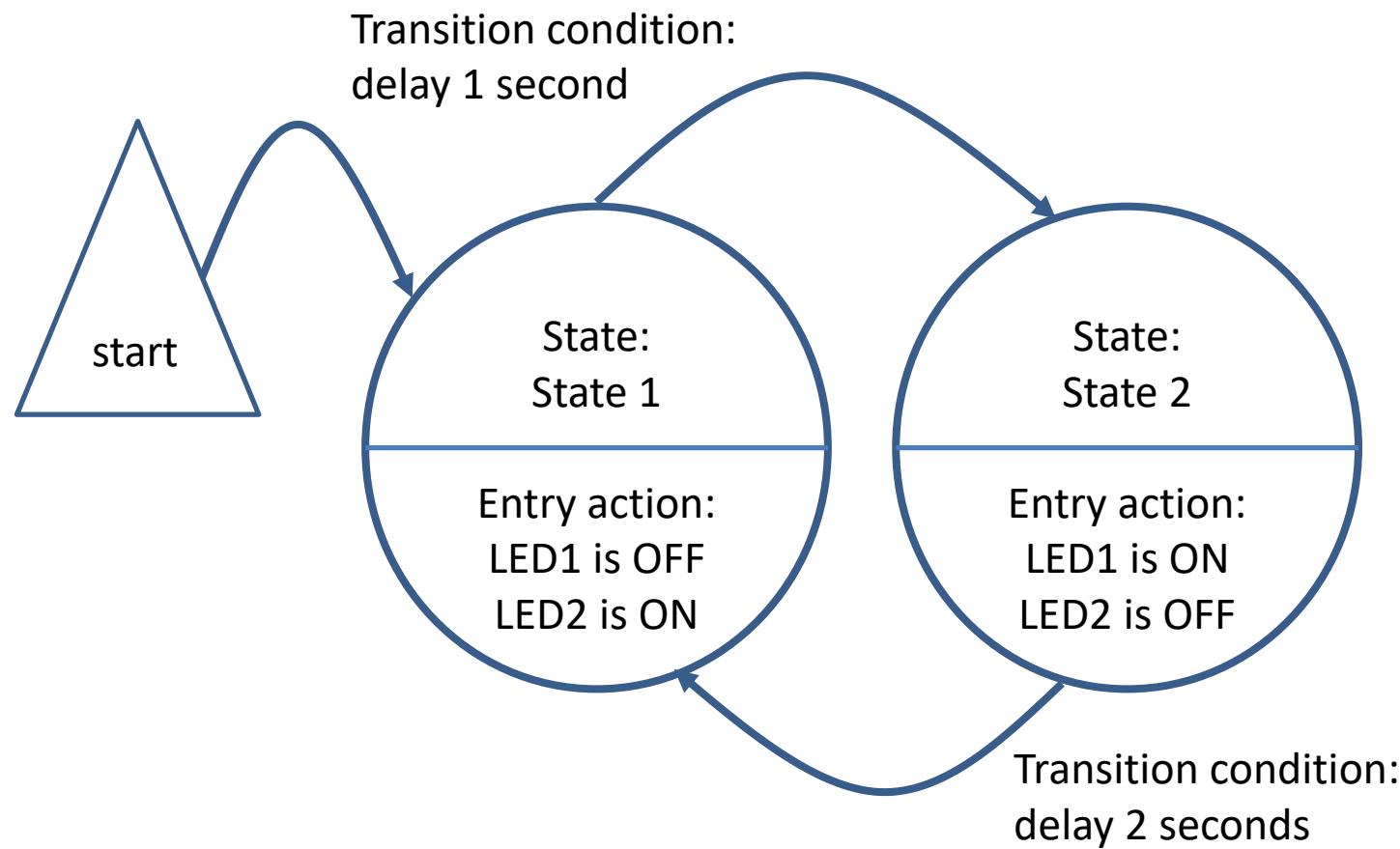
case STATE4: //stop
    // To be filled by students as an exercise
    .
    .

default: //none of above states
    state=STATE4;
    LM1=0;LM2=0;RM1=0;RM2=0;
    SPEED=200;DELAY_TIME=10;
    motors(LM1,LM2,RM1,RM2,SPEED,DELAY_TIME);
break;
}
```



1/2

- Using delay time as transition condition





# FSM Example 4: LEDs: FSM without Input/Sensor

2/2

```
#define STATE1 1
#define STATE2 2
#define STATE_END 100
unsigned char state=1;
//init. to state1

void setup() {
    pinMode (LED1, OUTPUT);
    pinMode (LED2, OUTPUT);
}

void loop() {
    switch(state) {
        case STATE1:
            digitalWrite(LED1, HIGH);
            digitalWrite(LED2, LOW);
            delay(1000);
            state=STATE2;
            break;
        case STATE2:
            digitalWrite(LED1, LOW);
            digitalWrite(LED2, HIGH);
            delay(2000);
            state=STATE1;
            break;
        case STATE_END:
            // turn off the LEDs, this state is not used here
            digitalWrite(LED1, HIGH);
            digitalWrite(LED2, HIGH);
            break;
        default:
            state=STATE_END;
            break;
    }
}
```

# FSM Example 4: LEDs: FSM with Input/Sensor



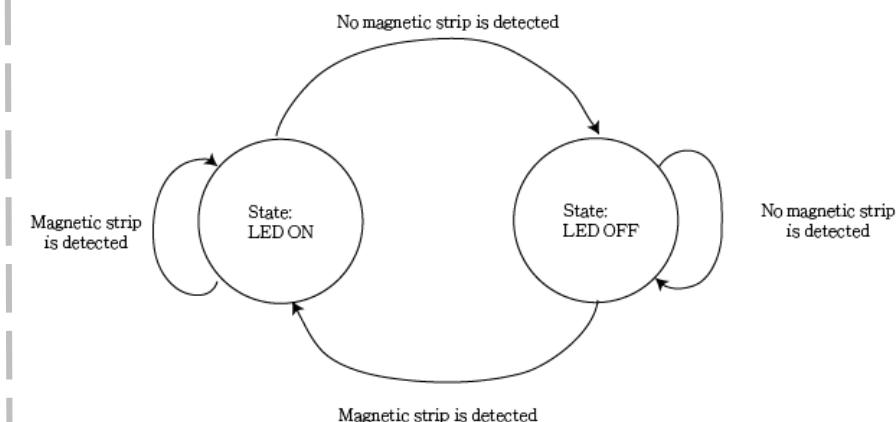
1/2

- When a magnetic strip is detected, the LED is ON
- When there is no magnetic strip, the LED is OFF

State Transition Table

| Input                         | Current State | Next State |
|-------------------------------|---------------|------------|
| Magnetic strip is detected    | ON            | ON         |
| Magnetic strip is detected    | OFF           | ON         |
| No magnetic strip is detected | ON            | OFF        |
| No magnetic strip is detected | OFF           | OFF        |

State Diagram



# FSM Example 4: LEDs: FSM with Input/Sensor



2/2

```
#define STATE1 1
#define STATE2 2
#define STATE_END 100
int magnetic = 7;
int ledPin_S1 = 5;
int ledPin_S2 = 6;
unsigned char state=1;
void setup() {
    pinMode (magnetic, INPUT);
    pinMode (ledPin_S1, OUTPUT);
    pinMode (ledPin_S2, OUTPUT);
}
void loop() {
    switch(state) {
        case STATE1:
            digitalWrite(ledPin_S1, HIGH);
            digitalWrite(ledPin_S2, LOW);
            if (digitalRead(magnetic) == LOW)
                state=STATE2; break;
        case STATE2:
            digitalWrite(ledPin_S1, LOW);
            digitalWrite(ledPin_S2, HIGH);
            if (digitalRead(magnetic) == HIGH)
                state=STATE1; break;
        case STATE_END:
            digitalWrite(ledPin_S1, HIGH);
            digitalWrite(ledPin_S2, HIGH);
            break;
        default:
            state=STATE_END;
            break;  }}
```

## Without FSM

```
void loop()
{
    /* if client was disconnected then try to reconnect again */

    if (!client.connected())
    {
        mqttconnect();
    }
    /* this function will listen for incoming subscribed topic-process-invoke receivedCallback */
}

client.loop();

/* we measure temperature every 3 secs we count until 3 secs reached to avoid blocking
program if using delay()*/

long now = millis();
if (now - lastMsg > 3000) {
    lastMsg = now;
    /* read DHT11/DHT22 sensor and convert to string */
    temperature = dht.readTemperature();
    if (!isnan(temperature))
    { snprintf (msg, 20, "%lf", temperature);
      /* publish the message */
      client.publish(TEMP_TOPIC, msg);
    }
}
```

<http://www.iotsharing.com/2017/05/how-to-apply-finite-state-machine-to-arduino-esp32-avoid-blocking.html>

# FSM Example 5: MQTT & DHT11



## With FSM

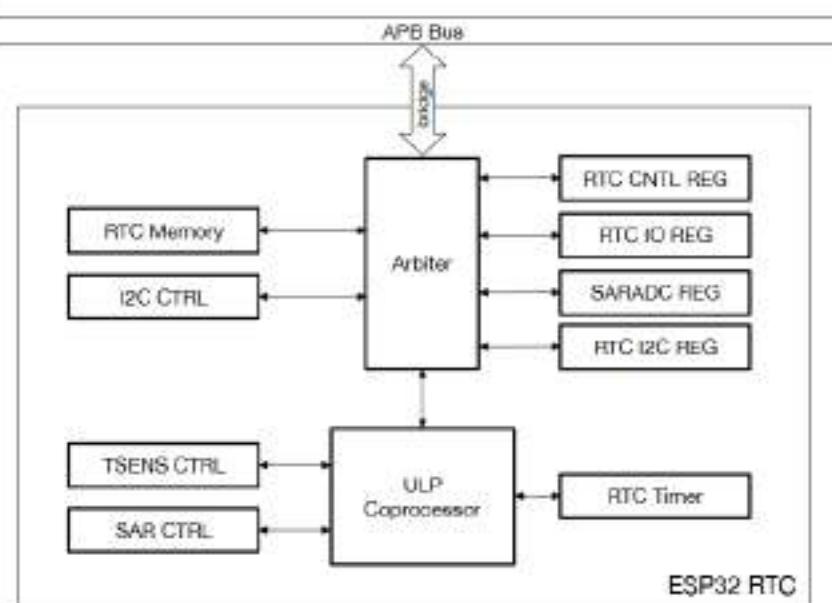
```
/* these are available states of DHT */
typedef enum {
    /* wait until 3 seconds */
    WAIT_FOR_TIMEOUT = 1,
    /* read DHT sensor for temperature */
    START_MEASURE,
    /* publish MQTT topic temperature */
    PUBLISH_MEASURE
} DHT_State;

/* create a variable to hold current state of DHT */
DHT_State state = WAIT_FOR_TIMEOUT;

void loop()
{
    /* if client was disconnected then try to reconnect again */
    if (!client.connected()) {
        mqttconnect();
    }
    /* this function will listen for incomming subscribed topic-process-invoke receivedCallback */
    client.loop();
    /* we measure temperature every 3 secs this code is to avoid blocking program and easy to read
     * maintain when using Finite State Machine */
    switch(state){
        case WAIT_FOR_TIMEOUT:
            long now = millis();
            if (now - lastMsg > 3000) {
                state = START_MEASURE;
            }
            break;
        case START_MEASURE:
            temperature = dht.readTemperature();
            if (!isnan(temperature)) {
                sprintf (msg, 20, "%lf", temperature);
                state = PUBLISH_MEASURE;
            }
            break;
        case PUBLISH_MEASURE:
            /* publish the message */
            client.publish(TEMP_TOPIC, msg);
            lastMsg = now;
            state = WAIT_FOR_TIMEOUT;
            break;
    }
}
```

<http://www.iotsharing.com/2017/05/how-to-apply-finite-state-machine-to-arduino-esp32-avoid-blocking.html>

## FSM with ULP Co-Processor



- The ULP co-processor is an ultra-low-power processor that remains powered on during the Deep-sleep mode of the main SoC.
- Hence, the developer can store in the RTC memory a program for the ULP co-processor to access peripheral devices, internal sensors and RTC registers during deep sleep.
- This is useful for designing applications where the CPU needs to be woken up by an external event, or timer, or a combination of these, while maintaining minimal power consumption.

- The ULP co-processor is a programmable FSM (Finite State Machine) that can work during deep sleep.
- Like general-purpose CPUs, ULP co-processor also has some instructions which can be useful for a relatively complex logic, and **also some special commands for RTC controllers/peripherals**.
- RTC GPIOs can be used to wake up the ESP32 from deep sleep when the ULP co-processor is running. (RTC\_GPIO0, RTC\_GPIOs 3-17)

References:

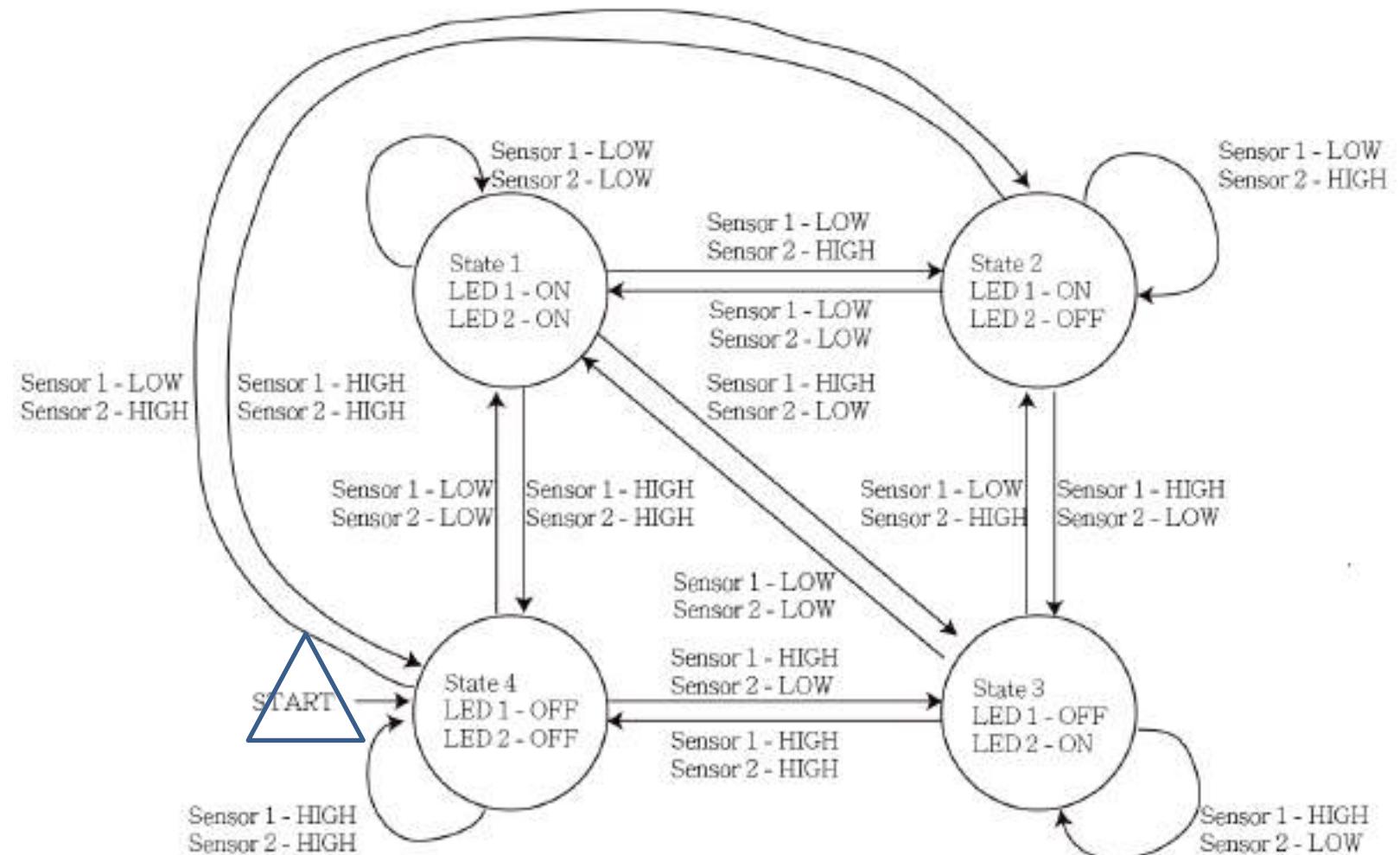
<https://circuits4you.com/2018/12/31/esp32-devkit-esp32-wroom-gpio-pinout/>

[http://www.ee.ic.ac.uk/pcheung/teaching/DE1\\_EE/Labs/esp32\\_technical\\_reference\\_manual\\_en.pdf](http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/Labs/esp32_technical_reference_manual_en.pdf)



- **Write state transition table and program** that controls **two LEDs** through two input signals, the specification is shown in flow diagram in the next slide
  - Use inputs
    - pinA for sensor1, and pinB for sensor2.
    - The values of input signals (sensor 1 and sensor 2) can be either GND (LOW) or 5V (HIGH)
  - Use outputs
    - pinC for LED1 and pinD for LED2
  - Print out the status of two input sensors on console.
- Submit a zip file in Grxx.FSMExercise-StudentID-StudentID on mycourses with
  - pdf report contains FSM diagram, fritzing schematic and source code
  - video file

## FSM Diagram





## Hints

```
//hint
#define STATE1 1
#define STATE2 2
#define STATE3 3
#define STATE4 4
#define STATE_END 100

int sensor1 = 0;
int sensor2 = 1;
int led1 = 5;
int led2 = 6;
unsigned char state=4;

void setup() {
    pinMode (sensor1, INPUT_PULLUP);
    pinMode (sensor2, INPUT_PULLUP);
    pinMode (led1, OUTPUT);
    pinMode (led2, OUTPUT);
}

void loop() {
    switch(state) {
        case STATE1:
            digitalWrite(led1, HIGH);
            digitalWrite(led2, HIGH);
            if ((digitalRead(sensor1) == LOW) &&
                (digitalRead(sensor2) == HIGH)) state=STATE2;
            else if ((digitalRead(sensor1) == HIGH) &&
                      (digitalRead(sensor2) == LOW)) state=STATE3;
            else if ((digitalRead(sensor1) == HIGH) &&
                      (digitalRead(sensor2) == HIGH)) state=STATE4;
            break;
    }
}
```

# References



## Arduino:

- <http://playground.arduino.cc/Code/FiniteStateMachine>
- Slides from Prof. Kin Hong Wong

## FSM:

[http://users.ece.utexas.edu/~valvano/Volume1/E-Book/C10\\_FiniteStateMachines.htm?fbclid=IwAR1UPmrNbXNd8zUhABG7FMLFWm8CdNr2-lukWrfW-MkszEkA5gGNMh7Y1s](http://users.ece.utexas.edu/~valvano/Volume1/E-Book/C10_FiniteStateMachines.htm?fbclid=IwAR1UPmrNbXNd8zUhABG7FMLFWm8CdNr2-lukWrfW-MkszEkA5gGNMh7Y1s)

## FSM Libraries:

Automaton: <https://github.com/tinkerspy/Automaton>

## Function-FSM:

<https://jrvеаlе.com/2019/06/03/library-function-fsm-c/>

github.com/JRVeale/function-fsm\

<https://github.com/jeroendoggen/Arduino-finite-state-machine-library>

## Cases:

<https://medium.com/coinmonks/multitasking-on-the-arduino-with-a-finite-state-machine-and-why-youll-need-it-for-sigfox-d52dafc55d8e>

## Books:

[Mar12] Michael Margolis, **Arduino Cookbook**, O'Reilly, 2012.

[Box13] John Boxall, Arduino Workshop: A Hands-On Introduction with 65 Projects,  
No Starch Press, San Francisco, 2013.

[Mon14] Programming Arduino: Next Steps: Going Further with Sketches, McGrawHill 2014.



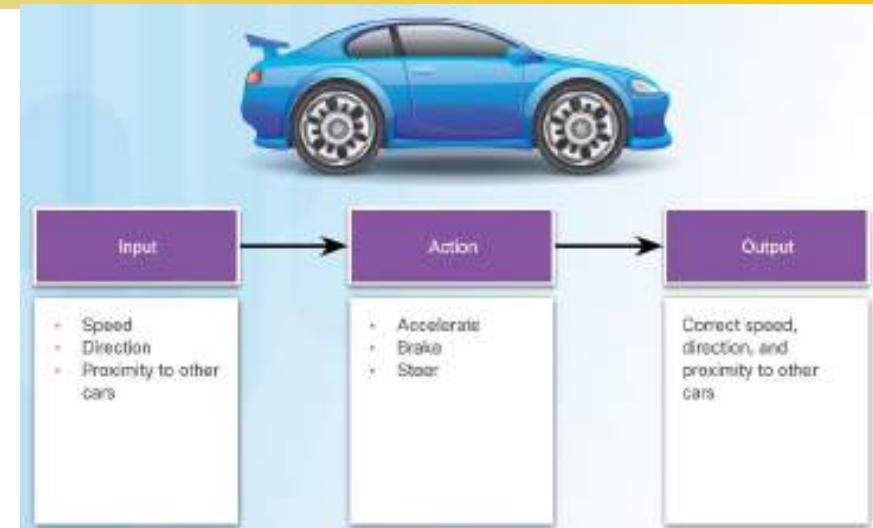
# PID

## Proportional Integral Derivative

# Review: Processes in Controlled Systems



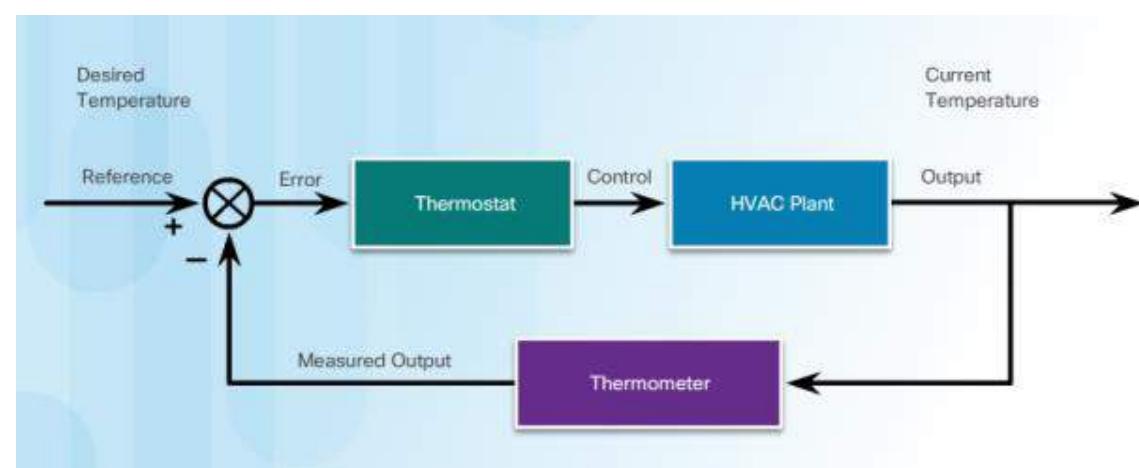
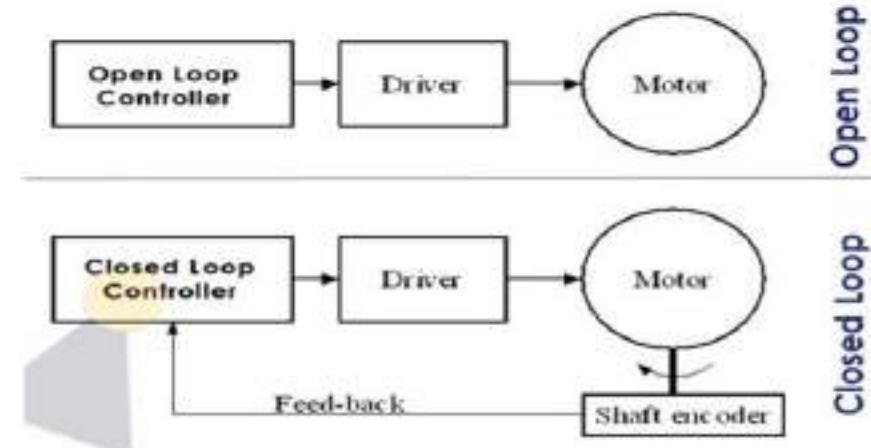
- Processes
  - ✓ A process is **a series of steps or actions** taken to achieve a desired result by the consumer of the process.
- Feedback
  - ✓ Feedback is when **the output of a process affects the input**.
  - ✓ Feedback is often referred to as a feedback loop.
  - ✓ Feedback loops can be positive or negative.
- Control Systems
  - ✓ Includes a controller that **uses inputs and outputs to manage and regulate the behavior of the system in an attempt to achieve a desired state**.
  - ✓ The controlled portion of the system is often called the **plant**.
  - ✓ Choosing the adjustments to apply to a plant to achieve a desired output is called **control theory**.
  - ✓ Control theory is applied to many systems, including driving a car.



# Review: Open-Loop/Closed-Loop Control Systems



- Open-Loop Control Systems
  - ✓ Open-loop control systems do not use feedback.
  - ✓ The plant performs a predetermined action without any verification of the desired results.
  - ✓ Open-loop control systems are often used for simple processes.
- Closed-Loop Control Systems
  - ✓ A closed-loop control system uses feedback to determine whether the collected output is the desired output.
  - ✓ The result is then fed back into a controller to adjust the plant for the next iteration of output, and the process repeats.



# Review: Closed-Loop Controllers



- Types of Closed-Loop Controllers
  - ✓ Proportional controllers (P): based on the difference between the measured output and the desired output.
    - Most systems have many interdependent pieces contributing to and affecting the output.
  - ✓ Integral controllers (PI): use historical data to measure how long the system has deviated from the desired output.
  - ✓ Proportional, Integral and Derivative controllers (PID): include data about how quickly the system is approaching the desired output.
    - PID controller is an efficient way to implement feedback control.
    - The Arduino and Raspberry Pi devices can be used to implement PID controllers.



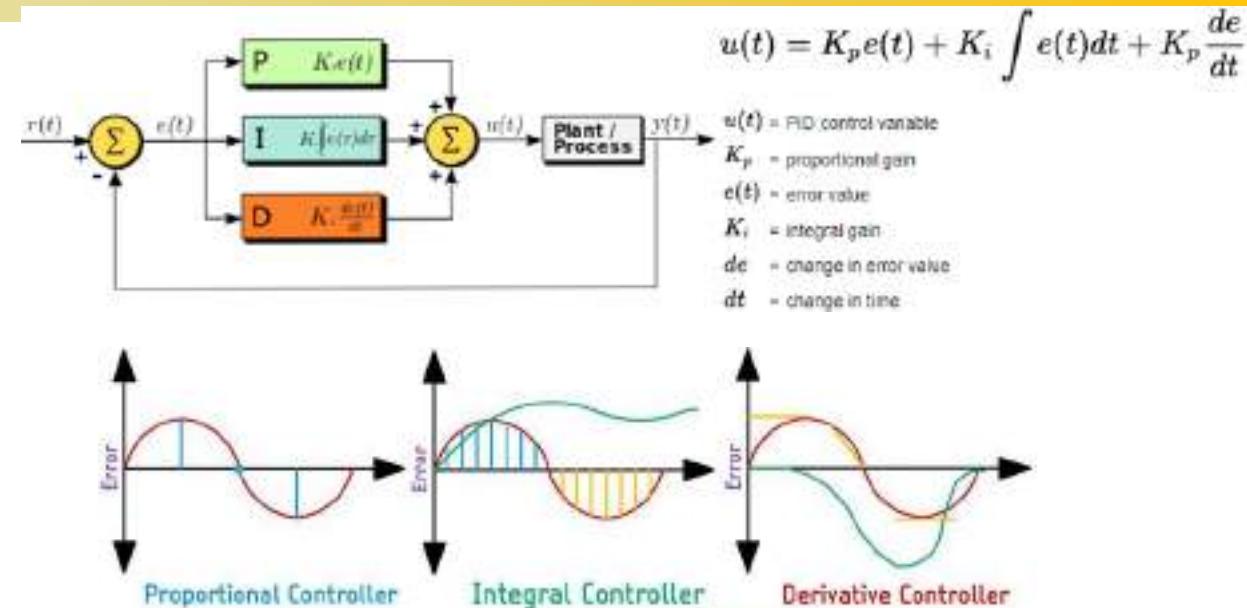
# PID (Proportional Integral Derivative)



- The **proportional** term:
  - It generates a response that is proportional to the error.
  - For example: the error is the angle of inclination of the robot.
- The **integral** term:
  - It generates a response based on the accumulated error.
  - This is essentially the sum of all the errors multiplied by the sampling period.
  - This is a response based on the behavior of the system in past.
- The **derivative** term:
  - It is proportional to the derivative of the error.
  - This is the difference between the current error and the previous error divided by the sampling period.
  - This acts as a predictive term that responds to how the robot might behave in the next sampling loop.
- Multiplying each of these terms by their corresponding constants (i.e, K<sub>p</sub>, K<sub>i</sub> and K<sub>d</sub>) and summing the result, we generate the output which is then sent as command to drive the motor.

For example:

```
motorPower = Kp*(error) + Ki*(errorSum)*sampleTime - Kd*(currentAngle-prevAngle)/sampleTime;
```



Ref:

<https://circuitdigest.com/microcontroller-projects/arduino-based-encoder-motor-using-pid-controller>

# PID Controller





## ESP32

### Project Examples

- ✓ Rancilio Silvia Coffee Machine  
<https://www.instructables.com/PID-Controlled-Thermostat-Using-ESP32-Applied-to-a/>
- ✓ PID based Encoder Motor Controller:  
<https://www.youtube.com/watch?v=Ds2E0NC4JhA>  
<https://josef-adamcik.cz/electronics/self-balancing-stories-ep2-firmware.html>
- ✓ Arduino Self Balancing Robot  
<https://www.instructables.com/Arduino-Self-Balancing-Robot-1/#step6>
- ✓ <https://githubmemory.com/repo/simplefoc/Arduino-FOC-balancer>

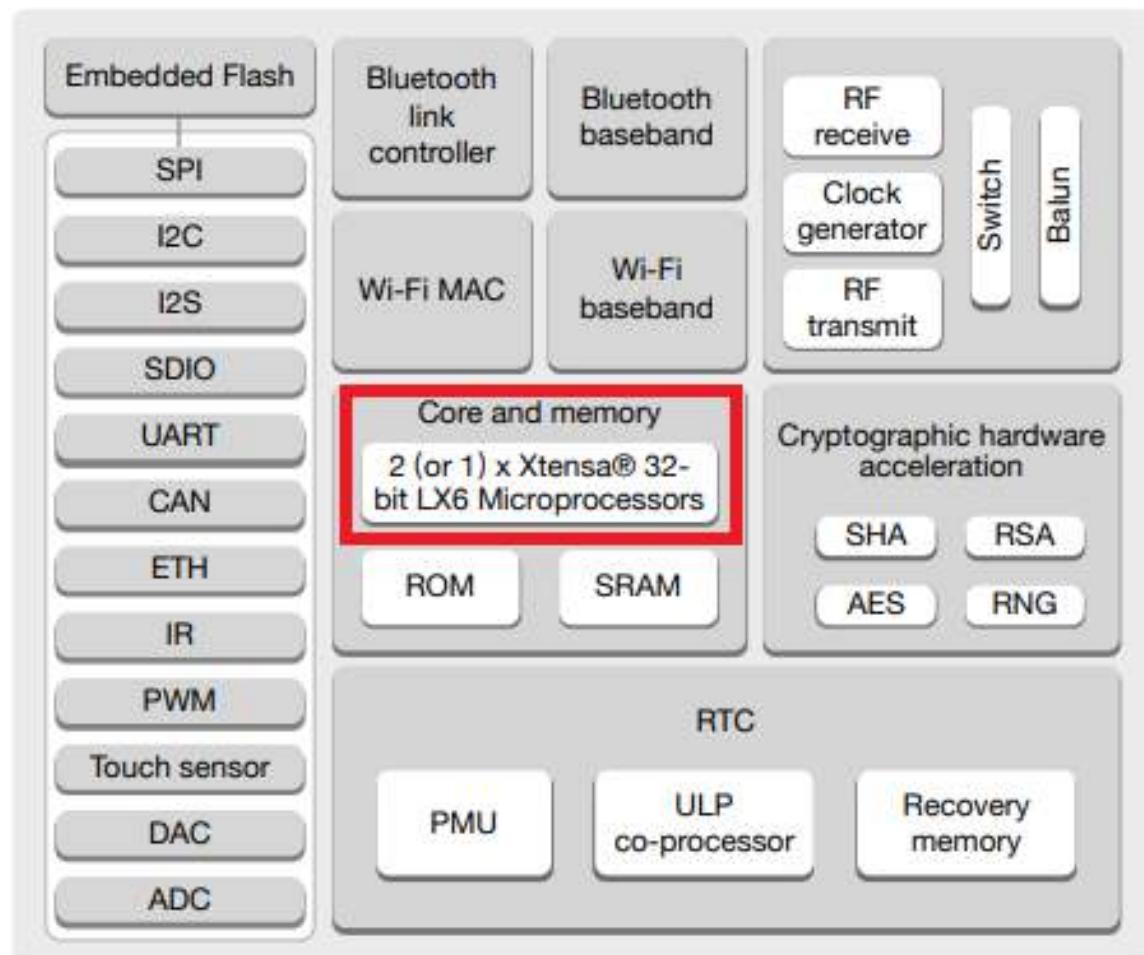
### Libraries

- ✓ QuickPID  
<https://www.arduino.cc/reference/en/libraries/quickpid/>  
<https://github.com/Dlloyddev/QuickPID/wiki/AutoTune>
- ✓ ArcPID (PID.h)  
<https://github.com/ettoreleandrotognoli/ArcPID>  
<https://platformio.org/lib/show/11921/ArcPID>
- ✓ AutoPID  
<https://r-downing.github.io/AutoPID/>
- ✓ SimpleFOClibrary (PIDController.h)  
[https://docs.simplefoc.com/pi\\_controller](https://docs.simplefoc.com/pi_controller)



# Dual Core on ESP32 FREERTOS

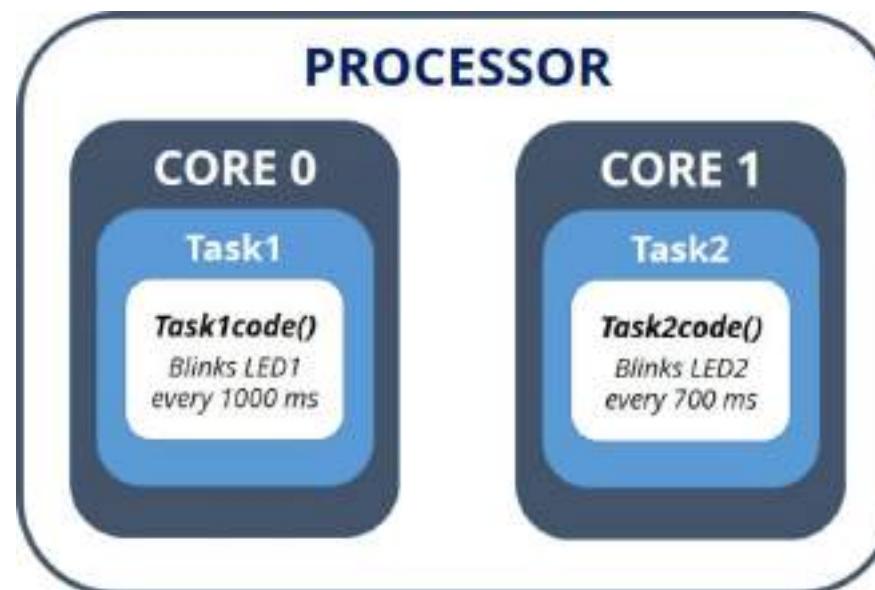
- 2 Xtensa 32-bit LX6 microprocessors: core 0 and core 1.



# FreeRTOS Supported by Arduino IDE



- The Arduino IDE supports FreeRTOS for the ESP32, which is a Real Time Operating system.
- This allows us to handle several tasks in parallel that run independently.



Try: <https://randomnerdtutorials.com/esp32-dual-core-arduino-ide/>



- [VB17] Ovidiu Vermesan and Joel Bacquet, “**Cognitive Hyperconnected Digital Transformation: Internet of Things Intelligence Evolution**”, River Publishers, 2017.
- [BHC+18] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, “**The Industrial Internet of Things (IIoT): An Analysis Framework**”, Computers in Industry 101, October 2018.
- [BS19] Rajkumar Buyya and Satish Narayana Srirama, “**Fog and Edge Computing: Principles and Paradigms**”, Wiley Series on Parallel and Distributed Computing, 2019.
- [RS19] Ammar Rayes and Samer Salam, “**Internet of Things from Hype to Reality: The Road to Digitization**”, 2<sup>nd</sup> Edition, Springer, 2019.
- [LEA20] Perry Lea, “**IoT and Edge Computing for Architects: Implementing edge and IoT systems from sensors to clouds with communication systems, analytics, and security**”, 2<sup>nd</sup> Edition, Packt Publishing, 2020.
- [GG20] Patrick Grossetete and Stephen Goodman, “**Wireless Technologies and Use Cases in Industrial IoT**”, BKKIOT-1775, Cisco Live, January 2020.
- Lecture Course: Wireless and Mobile Networking <https://www.cse.wustl.edu/~jain/cse574-20/index.html>



**ITCS447**

**Lecture 9**

**IoT Communication and Networking**

**TCP/IP Model and Access Networks**

**Standards and Technologies**

**[RS19]**

Asst. Prof. Dr. Thitinan Tantidham



ห้ามมิให้นักศึกษาทำข้า ดัดแปลง หรือใช้ประโยชน์จากการอันมีลิขสิทธิ์ปราการอยู่ในระบบการศึกษาอิเล็กทรอนิกส์ (E-Learning) ของมหาวิทยาลัย ไม่ว่าจะทั้งหมดหรือบางส่วน โดยไม่ได้รับอนุญาตจากมหาวิทยาลัย นอกเหนือจากการศึกษาส่วนบุคคล ทั้งนี้ การทำข้า ดัดแปลง หรือเผยแพร่ต่อสาธารณะชื่องานอันมีลิขสิทธิ์ จะมีโทษปรับตั้งแต่ 20,000 บาท ถึง 200,000 บาท และหากเป็นการกระทำเพื่อการค้า จะมีโทษจำคุกตั้งแต่ 6 เดือน ถึง 10 ปี หรือปรับตั้งแต่ 100,000 บาท ถึง 800,000 บาท หรือทั้งจำทั้งปรับ

Unless allowed by Mahidol University, the User shall not copy, modify, or exploit in part or in whole of the copyrighted materials on the Platform, other than for your own individual study. The copy, modify or communication to public of the copyrighted materials shall be inflicted with a fine from 20,000 Baht up to 200,000 Baht. If the offence is committed with the commercial purpose, the offender shall be inflicted with imprisonment for a term from 6 months up to 10 years or a fine from 100,000 Baht up to 800,000 Baht or both imprisonment and fine.



- Review: History of The Internet & IoT
- Gartner's Hype Cycle
- IoT Communication and Networking
  - OSI and TCP/IP Networking Models
  - Standards and Technologies
  - Network Equipment
  - IoT Protocol Stack
- IoT Networking Considerations and Challenges
- IoT Connectivity & Access Networks

# Review: History of the IoT [Lea20]

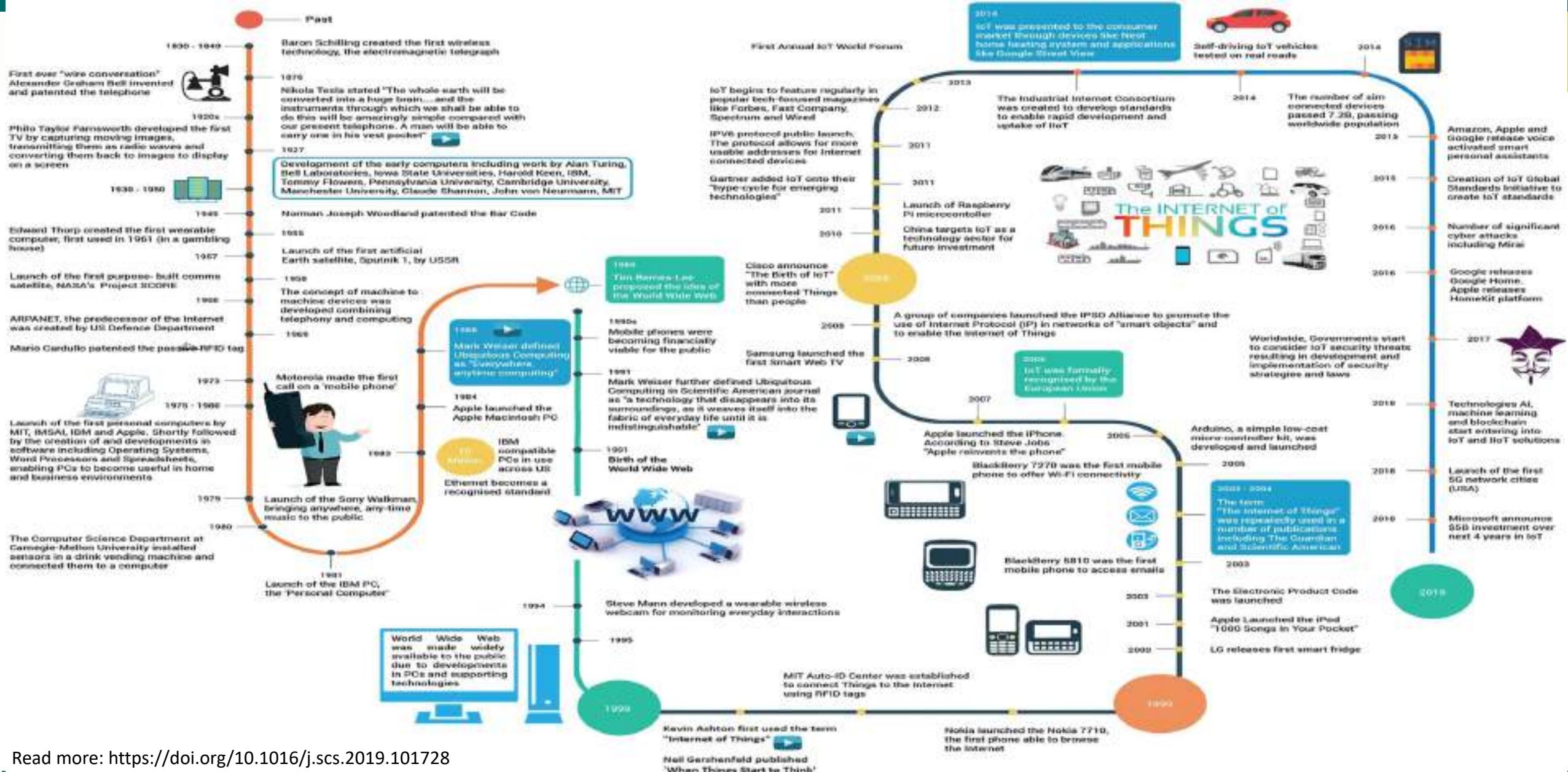


The following timeline shows the slow progress in connecting things to the Internet.

| Year | Device                                                                                                                                                                                                                      | Reference                                                                                                                                                                                 |                                                                                                          |
|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|
| 1969 | <b>The Internet starting with ARPANET (TCP/IP)</b>                                                                                                                                                                          |                                                                                                                                                                                           | Read more: <a href="https://www.postscapes.com/iot-history/">https://www.postscapes.com/iot-history/</a> |
| 1973 | Mario W. Cardullo receives the patent for first RFID tag.                                                                                                                                                                   | US Patent US 3713148 A                                                                                                                                                                    |                                                                                                          |
| 1982 | Carnegie Mellon Internet-connected soda machine.                                                                                                                                                                            | <a href="https://www.cs.cmu.edu/~coke/history_long.txt">https://www.cs.cmu.edu/~coke/history_long.txt</a>                                                                                 |                                                                                                          |
| 1989 | Internet-connected toaster at Interop'89.                                                                                                                                                                                   |                                                                                                                                                                                           | IEEE Consumer Electronics Magazine (Volume: 6, Issue: 1, Jan. 2017)                                      |
| 1991 | HP introduces HP LaserJet IISi: the first Ethernet-connected network printer.                                                                                                                                               | <a href="http://hpmuseum.net/display_item.php?hw=350">http://hpmuseum.net/display_item.php?hw=350</a>                                                                                     |                                                                                                          |
| 1993 | Internet-connected coffee pot at University of Cambridge (the first Internet-connected camera).                                                                                                                             | <a href="https://www.cl.cam.ac.uk/coffee/qsf/coffee.html">https://www.cl.cam.ac.uk/coffee/qsf/coffee.html</a>                                                                             |                                                                                                          |
| 1996 | General Motors OnStar (2001 remote diagnostics).                                                                                                                                                                            | <a href="https://en.wikipedia.org/wiki/OnStar">https://en.wikipedia.org/wiki/OnStar</a>                                                                                                   |                                                                                                          |
| 1998 | Bluetooth Special Interest Group (SIG) formed.                                                                                                                                                                              | <a href="https://www.bluetooth.com/about-us/our-history">https://www.bluetooth.com/about-us/our-history</a>                                                                               |                                                                                                          |
| 1999 | LG Internet Digital DIOS refrigerator.                                                                                                                                                                                      | <a href="https://www.telecompaper.com/news/lg-unveils-internetreadyrefrigerator--221266">https://www.telecompaper.com/news/lg-unveils-internetreadyrefrigerator--221266</a>               |                                                                                                          |
| 2000 | First instances of the Cooltown concept of pervasive computing everywhere: HP Labs, a system of computing and communication technologies that, combined, create a web-connected experience for people, places, and objects. | <a href="https://www.youtube.com/watch?v=U2AkkuIVV-I">https://www.youtube.com/watch?v=U2AkkuIVV-I</a>                                                                                     |                                                                                                          |
| 2001 | First Bluetooth product launched: KDDI Bluetooth-enabled mobile phone.                                                                                                                                                      | <a href="http://edition.cnn.com/2001/BUSINESS/asia/04/17/tokyo.kddibluetooth/index.html">http://edition.cnn.com/2001/BUSINESS/asia/04/17/tokyo.kddibluetooth/index.html</a>               |                                                                                                          |
| 2005 | United Nation's International Telecommunications Union report predicting the rise of IoT for the first time.                                                                                                                | <a href="http://www.itu.int/osg/spu/publications/internetofthings/internetofThings_summary.pdf">http://www.itu.int/osg/spu/publications/internetofthings/internetofThings_summary.pdf</a> |                                                                                                          |
| 2008 | IPSO Alliance formed to promote IP on objects, first IoT-focused alliance.                                                                                                                                                  | <a href="https://www.ipso-alliance.org">https://www.ipso-alliance.org</a>                                                                                                                 |                                                                                                          |
| 2010 | The concept of Smart Lighting formed after success in developing solid-state LED light bulbs.                                                                                                                               | <a href="https://www.bu.edu/smartlighting/files/2010/01/BobK.pdf">https://www.bu.edu/smartlighting/files/2010/01/BobK.pdf</a>                                                             |                                                                                                          |
| 2014 | Apple creates iBeacon protocol for beacons.                                                                                                                                                                                 | <a href="https://support.apple.com/enus/HT202880">https://support.apple.com/enus/HT202880</a>                                                                                             |                                                                                                          |

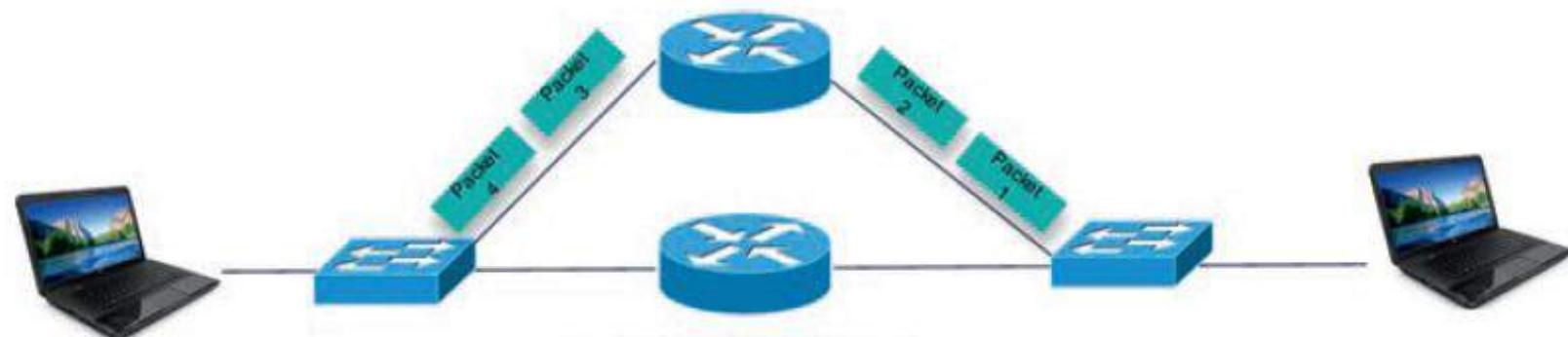


# Review: History of the Internet & IoT [AAH+19]

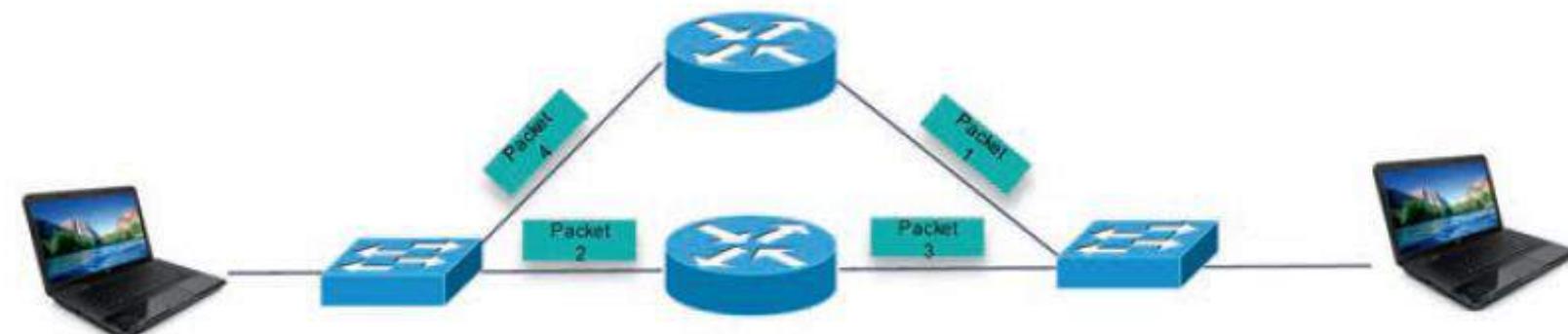


Read more: <https://doi.org/10.1016/j.scs.2019.101728>

# Review: Packet Switching vs Circuit Switching

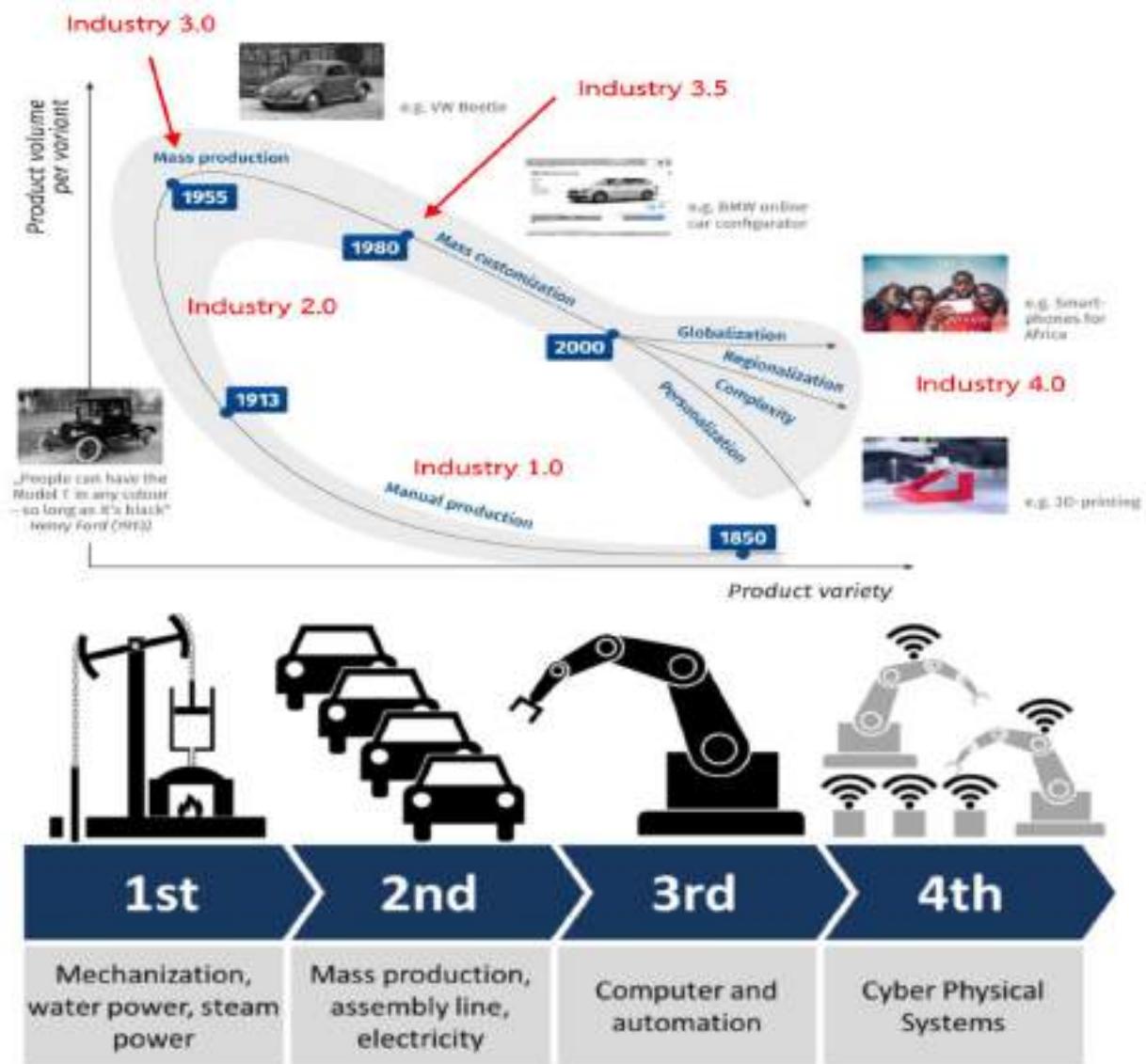
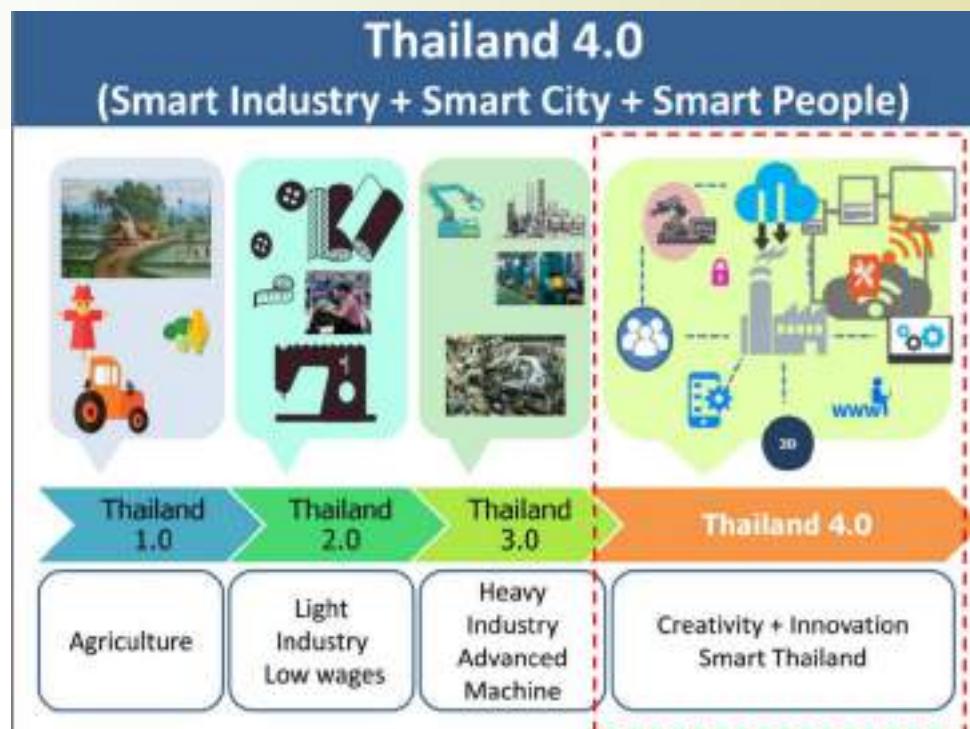


**A. Circuit-Switched**  
**Packets Follows Same Path & Arrive in Order**



**B. Packet-Switched**  
**Packets May Follow Different Paths & Arrive Out-of-Order**

# Review: Thailand 4.0 vs Industry 4.0



<http://www.stta.or.th/images/pdf/Thailand4/Thailand4.pdf>

1784

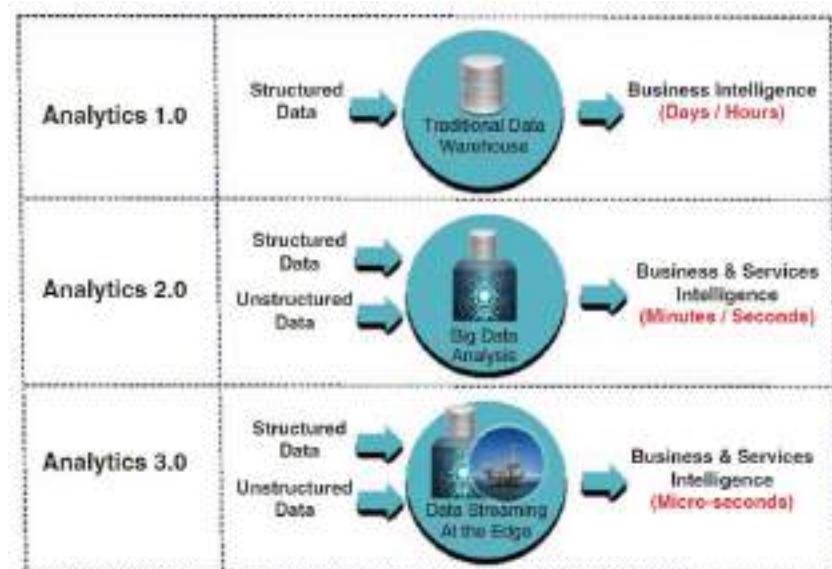
1870

1969

# Review: Analytics 1.0/2.0/3.0



|                        | Analytics 1.0           | Analytics 2.0               | Analytics 3.0               |
|------------------------|-------------------------|-----------------------------|-----------------------------|
| Collected data type    | Structured              | Structured and unstructured | Structured and unstructured |
| Data analysis location | Centralized data center | Centralized data center     | At edge and in data center  |
| Time to analyze data   | Days–hours              | Hours–minutes               | Seconds–microseconds        |
| Data volume            | Small data              | Big data                    | Big data                    |

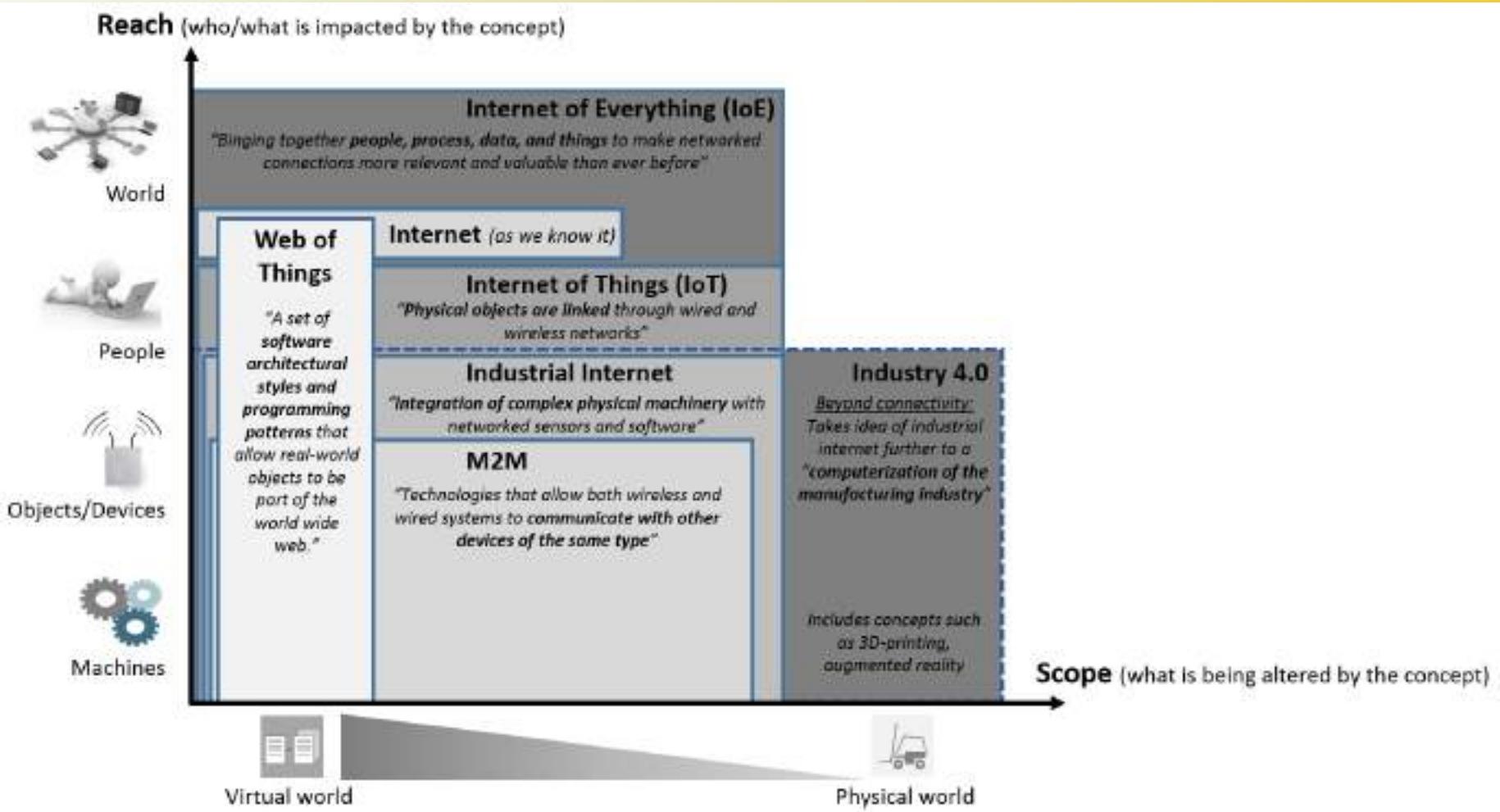


# Review: M2M vs IoT



| M2M                                                                                  | IoT                                                                                        |
|--------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|
| Machines                                                                             | Sensors                                                                                    |
| Hardware-based                                                                       | Software-based                                                                             |
| Vertical applications                                                                | Horizontal applications                                                                    |
| Deployed in a closed system                                                          | Connects to a larger network                                                               |
| Machines communicating with machines                                                 | Machines communicating with machines, humans with machines, machines with humans           |
| Uses non-IP protocol                                                                 | Uses IP protocols                                                                          |
| Can use the cloud, but not required to                                               | Uses the cloud                                                                             |
| Machines use point-to-point communication, usually embedded in hardware              | Devices use IP networks to communicate                                                     |
| Often one-way communication                                                          | Back and forth communication                                                               |
| Main purpose is to monitor and control                                               | Multiple applications; multilevel communications                                           |
| Operates via triggered responses based on an action                                  | Can, but does not have to, operate on triggered responses                                  |
| Limited integration options, devices must have complementary communication standards | Unlimited integration options, but requires software that manages communications/protocols |
| Structured data                                                                      | Structured and unstructured data                                                           |

# Review: M2M vs IoT vs IoE



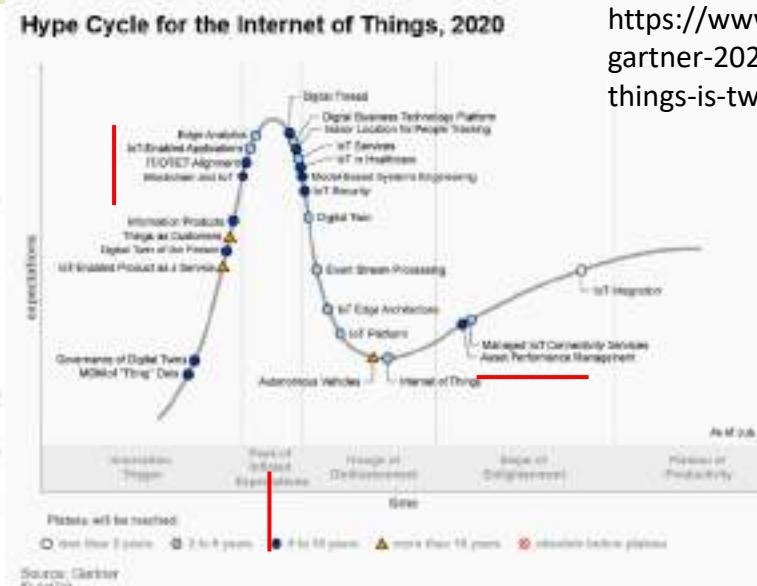
Wikipedia, McKinsey, IoT Analytics

<https://iot-analytics.com/internet-of-things-definition/>



# Gartner's Hype Cycle

## Internet of Things



<https://www.gartner.com/en/newsroom/press-releases/2020-09-09-gartner-2020-hype-cycle-for-supply-chain-strategy-shows-internet-of-things-is-two-to-five-years-away-from-transformational-impact>

<https://www.zdnet.com/article/gartner-releases-its-2021-emerging-tech-hype-cycle-heres-whats-in-and-headed-out/>

## Hype Cycle for Emerging Technologies, 2021



[gartner.com](http://gartner.com)

Source: Gartner  
© 2021 Gartner, Inc. and/or its affiliates. All rights reserved. Gartner and Hype Cycle are registered trademarks of Gartner, Inc. and its affiliates in the U.S. and other countries.

**Gartner**

- Artificial intelligence's impact on generating code, augmenting design and innovation is all 5- to 10-years away.
- Composable is going to be a key buzzword for applications and networks.
- Industry clouds are just beginning on the hype cycle with a plateau reached in 5- to 10-years. That take is interesting given industry clouds are everywhere from multiple vendors.
- Digital humans are being talked about a good bit, but Gartner reckons the technology is more than 10 years away from productivity gains. Quantum-based machine learning is also more than 10 years out.

[https://en.wikipedia.org/wiki/Gartner\\_hype\\_cycle#Five\\_phases](https://en.wikipedia.org/wiki/Gartner_hype_cycle#Five_phases)

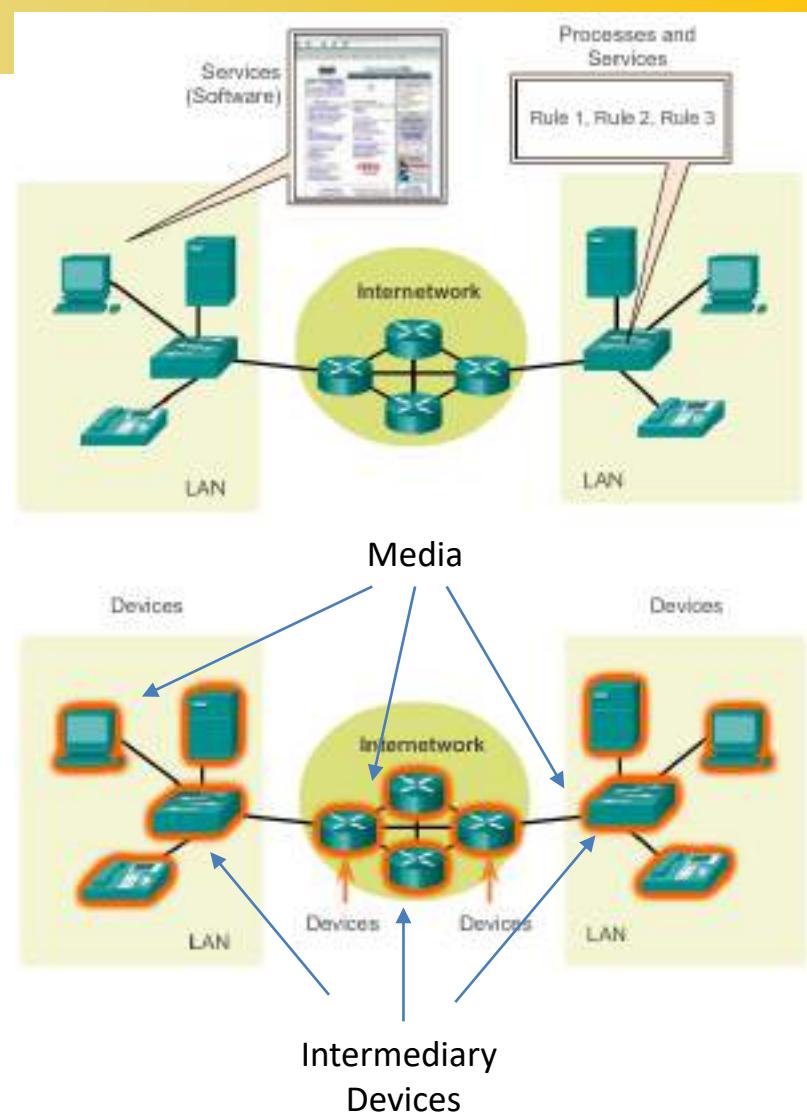
- OSI and TCP/IP Networking Models
- Standards and Technologies
- Network Equipment
- IoT Protocol Stack
  - Network Access and Physical Layer
  - Internet Layer
  - Application Layer



# OSI and TCP/IP Networking Models

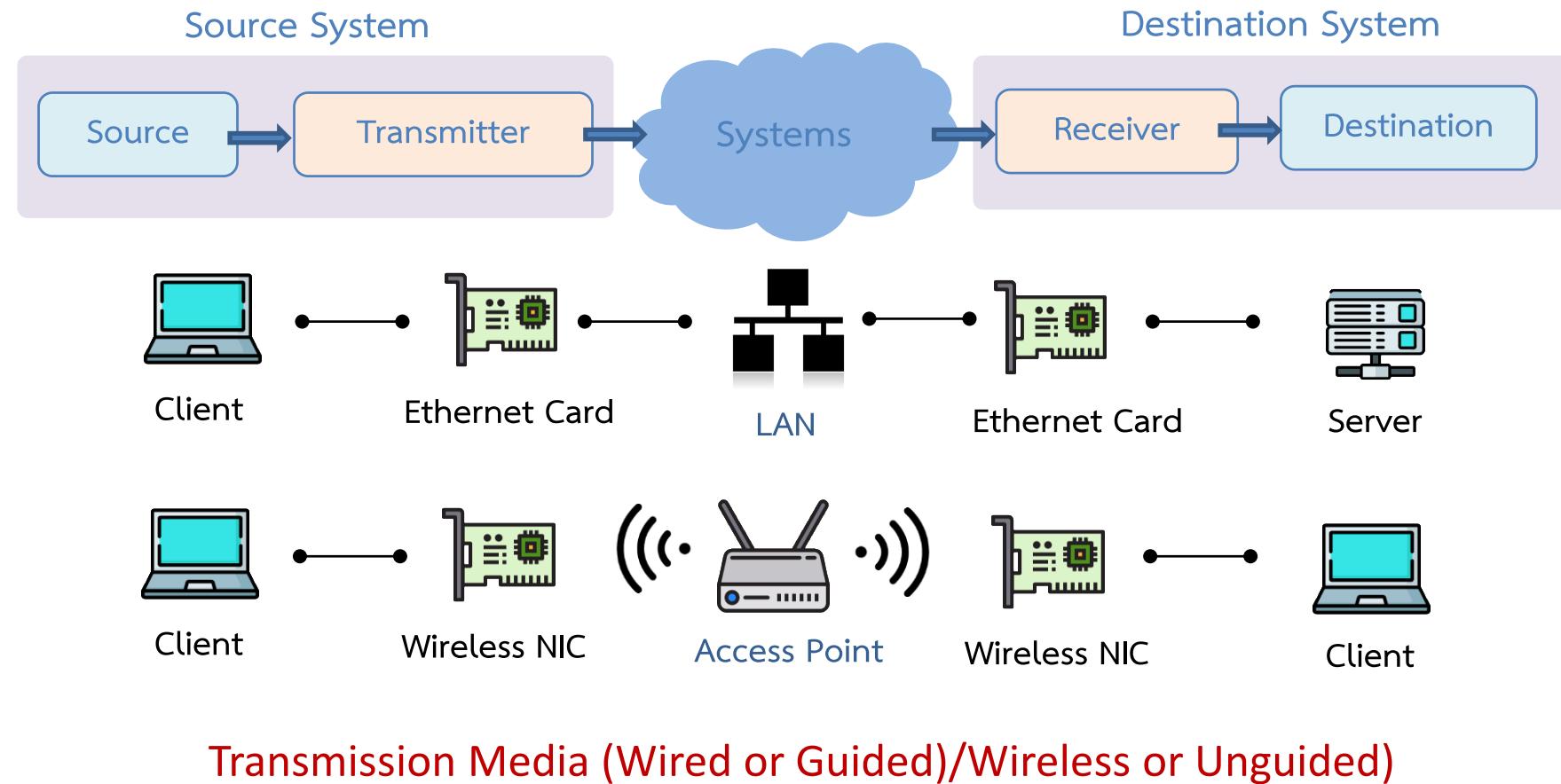
## Network Components (1/2)

- **(End) Devices or Hosts: Sender/Receiver or Source/Destination**
  - Either the source or destination of a message
  - Hosts: Servers -computers provide information to end devices.  
Clients –computers send requests to the servers to retrieve information.
  - IoT devices (e.g. cameras, door locks, doorbells, refrigerators, audio/visual systems, and various sensors)
- **(Intermediary Network) Devices**
  - Connect multiple individual networks to form an internetwork
  - Connect the individual end devices to the network
  - Ensure data flows across the network
  - Provide connectivity
- **(Network) Media**
  - Provide the pathway for data transmission
  - Interconnect devices
  - Wired e.g. copper and fiber optic and Wireless e.g. WiFi and Bluetooth
- **Processes and Services**
  - Services provide information in response to a request. They include network applications e.g. file, email, and web services.
  - Processes provide the functionality that directs and moves the messages through the network e.g. routing and forwarding operations on a router device.





## Network Components (2/2)



NIC: Network Interface Card

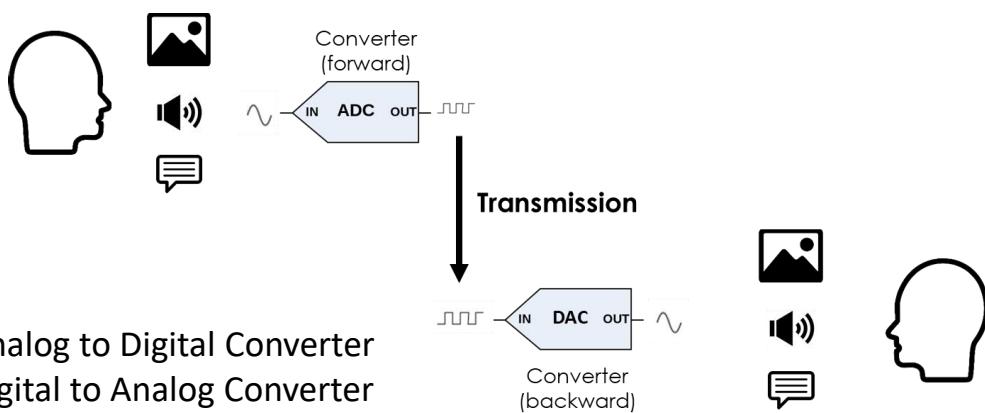
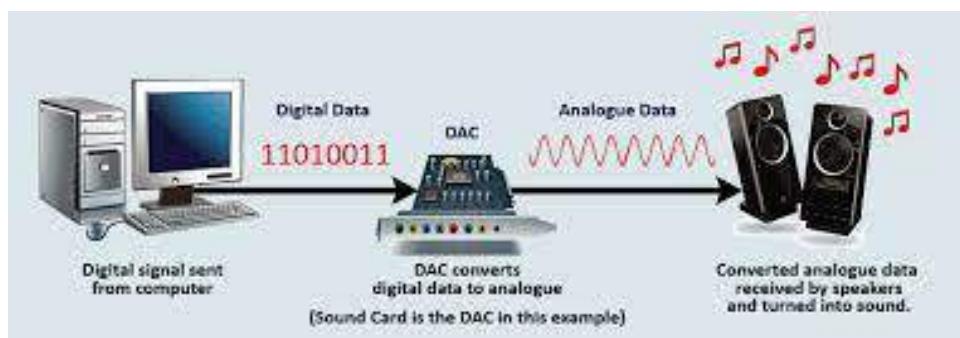
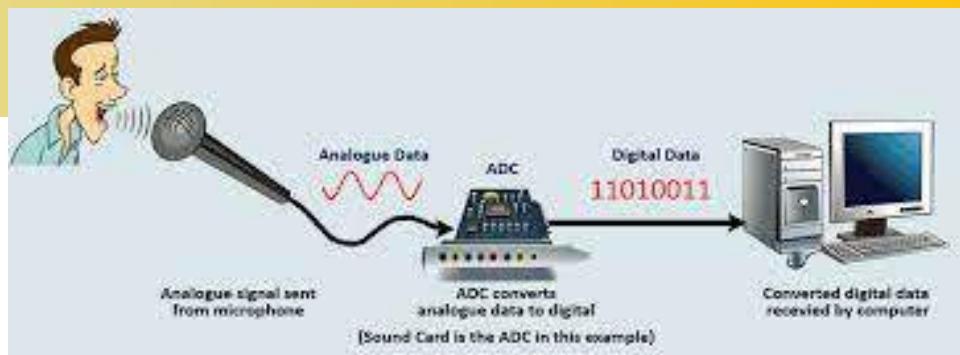
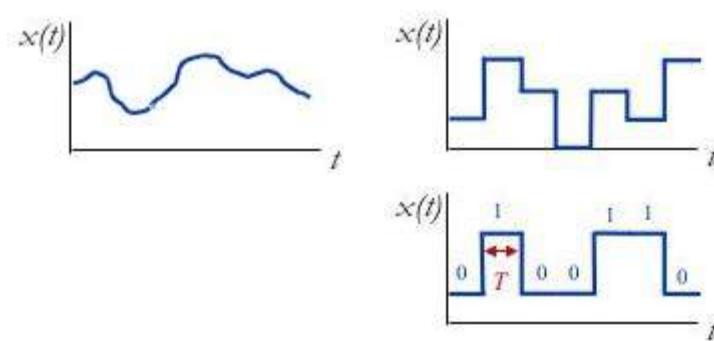
# OSI and TCP/IP Networking Models



## Analog/Digital Data and Signals

Table 2-1 Four combinations of data and signals

| Data    | Signal                | Encoding or Conversion Technique                                               | Common Devices          | Common Systems                                                         |
|---------|-----------------------|--------------------------------------------------------------------------------|-------------------------|------------------------------------------------------------------------|
| Analog  | Analog                | Amplitude modulation<br>Frequency modulation                                   | Radio tuner<br>TV tuner | Telephone<br>AM and FM radio<br>Broadcast TV<br>Cable TV               |
| Digital | (Square-wave) Digital | NRZ-L<br>NRZI<br>Manchester<br>Differential Manchester<br>Bipolar-AMI<br>4B/5B | Digital encoder         | Local area networks<br>Telephone systems                               |
| Digital | (Discrete) Analog     | Amplitude shift keying<br>Frequency shift keying<br>Phase shift keying         | Modem                   | Dial-up Internet access<br>DSL<br>Cable modems<br>Digital Broadcast TV |
| Analog  | Digital               | Pulse code modulation<br>Delta modulation                                      | Codec                   | Telephone systems<br>Music systems                                     |





## Communication Protocol (1/2)

- A **communication protocol** is a **system** of **rules** that **allow two or more entities of a communications system to transmit information** via any kind of variation of a physical quantity.
- The **protocol** defines the **syntax**, **semantics** and **synchronization** of communication and **possible error recovery methods**.
- **Protocols** may be implemented by **hardware**, **software**, or a combination of both.

**Syntax:** rules of communication (e.g. grammar)

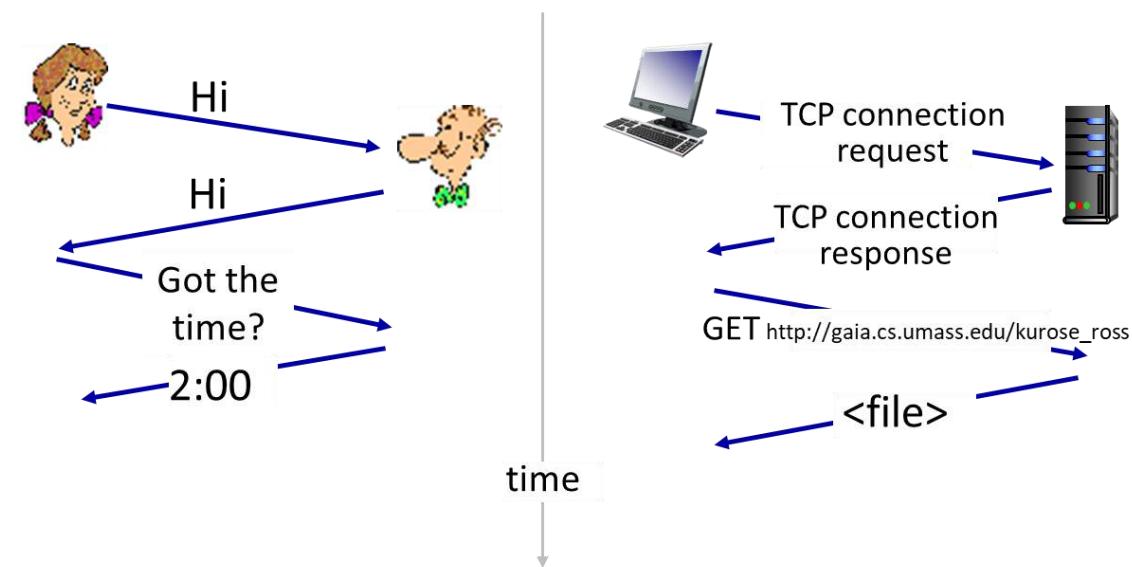
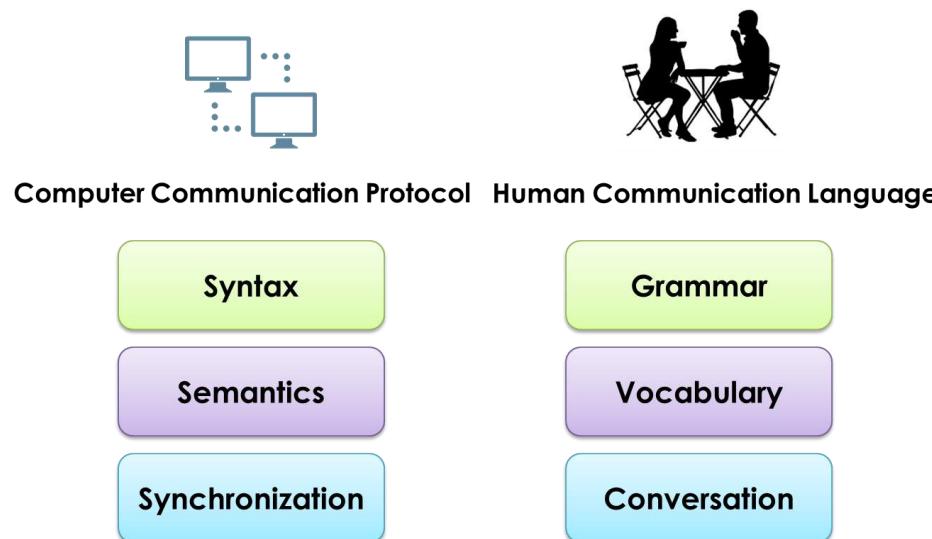
**Semantics:** means of symbols (e.g. vocabulary)

**Synchronization:** timing of conversation (e.g. timing)



## Communication Protocol (2/2)

- A protocol is analogous to “communication language” in human terms.



# OSI and TCP/IP Networking Models



## OSI vs TCP/IP

OSI is introduced by ISO (International Organization for Standardization) 1984.

**TCP/IP**

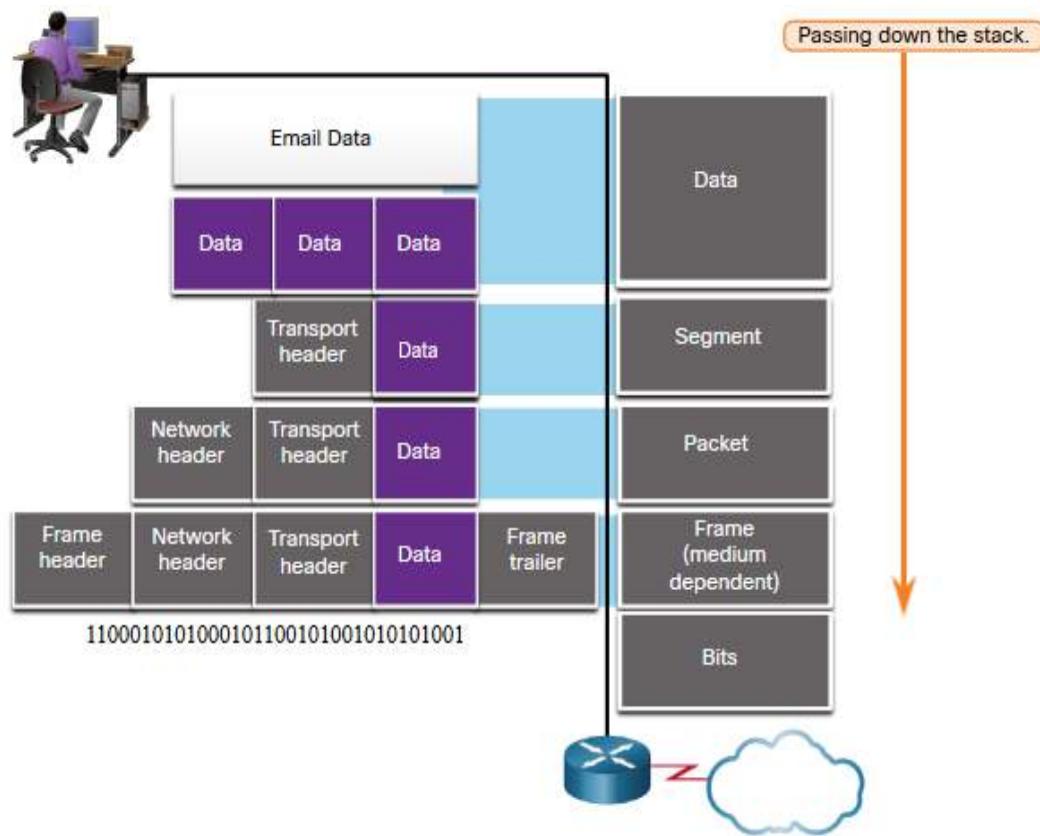
| OSI layer    | Main function                                                          | Examples of main devices                              | Examples of main protocol                                                        |
|--------------|------------------------------------------------------------------------|-------------------------------------------------------|----------------------------------------------------------------------------------|
| Application  | Provides network services to the end host's applications               | Server, laptops, PCs                                  | HTTPS, FTP, Telent, SSH                                                          |
| Presentation | Ensures the data can be understood between two end hosts               | N/A<br><i>Application or Network Operating System</i> | Data encoding, data formatting, and serialization                                |
| Session      | Manages multiple sessions between end hosts                            | N/A<br><i>Application or Network Operating System</i> | Connection management, error recovery<br><i>Authentication e.g. login/logout</i> |
| Transport    | Establishes end-to-end connectivity and ensures reliable data delivery | Firewalls                                             | TCP, UDP                                                                         |
| Network      | Connectivity and path selection based on logical addresses             | Routers, firewalls                                    | IPv4, IPv6                                                                       |
| Data link    | Defines data format for transmission                                   | Switches, APs                                         | IEEE 802.1 (Ethernet), PPP                                                       |
| Physical     | Defines physical media access and properties                           | Fiber optics, category 5 cables, coaxial cables       | IEEE 802.3                                                                       |

OSI (Open Systems Interconnection)

[https://en.wikipedia.org/wiki/OSI\\_model](https://en.wikipedia.org/wiki/OSI_model)



## Data Encapsulation/Decapsulation

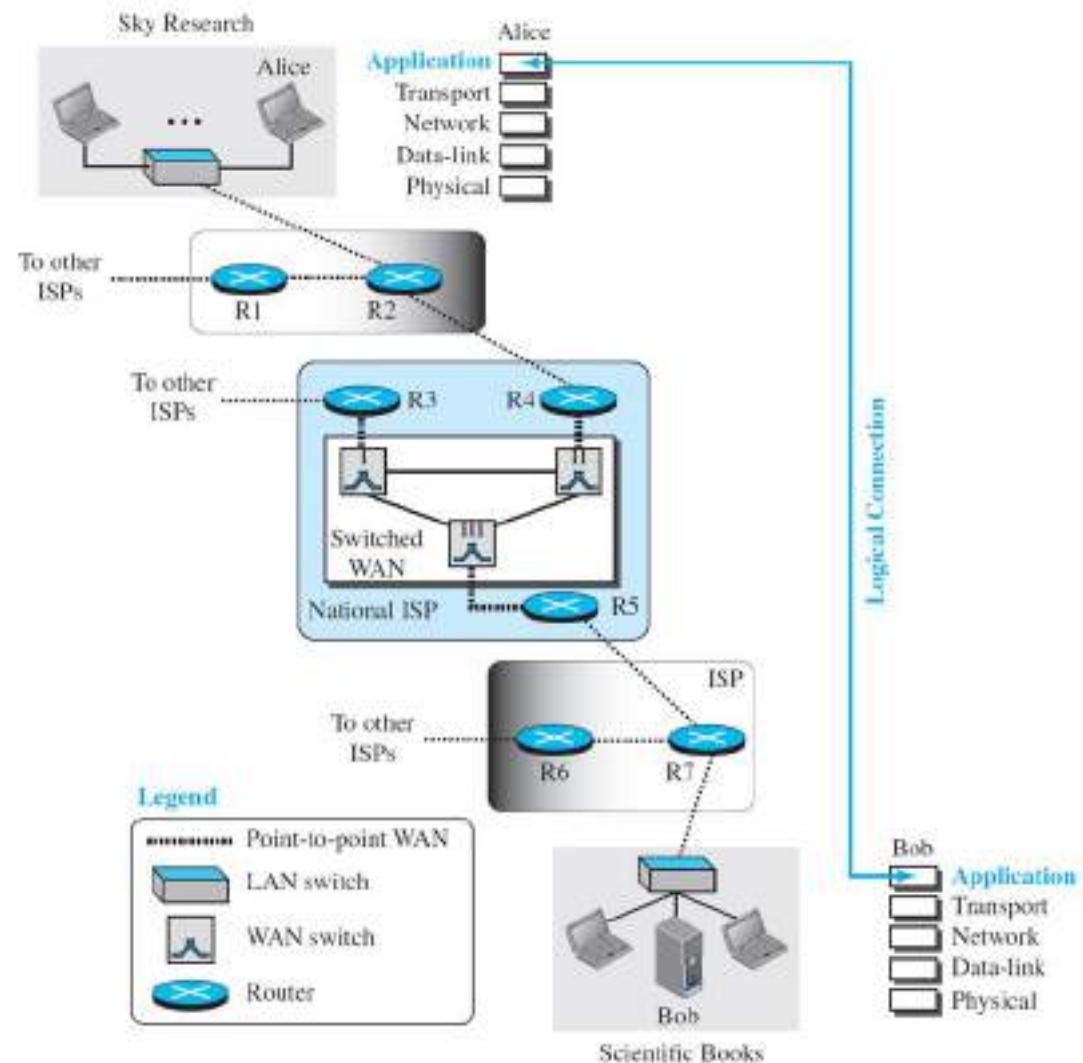


Encapsulation is the process where protocols add their information to the data.

- At each stage of the process, a PDU has a different name to reflect its new functions.
- There is no universal naming convention for PDUs, in this course, the PDUs are named according to the protocols of the TCP/IP suite.
- PDUs passing down the stack are as follows:
  1. Data (Data Stream)
  2. Segment
  3. Packet
  4. Frame
  5. Bits (Bit Stream)

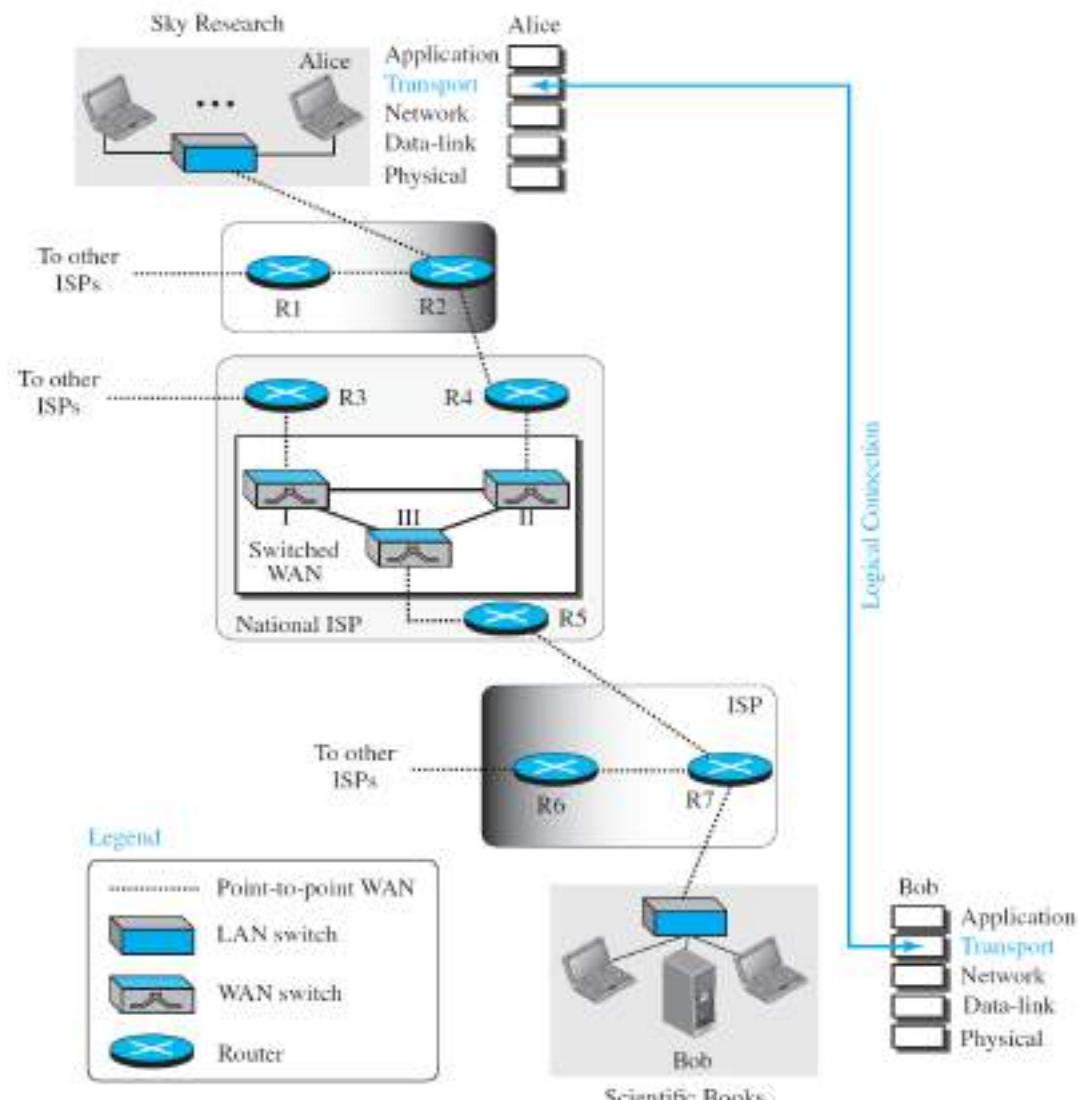


## TCP/IP: Application Layer Communication



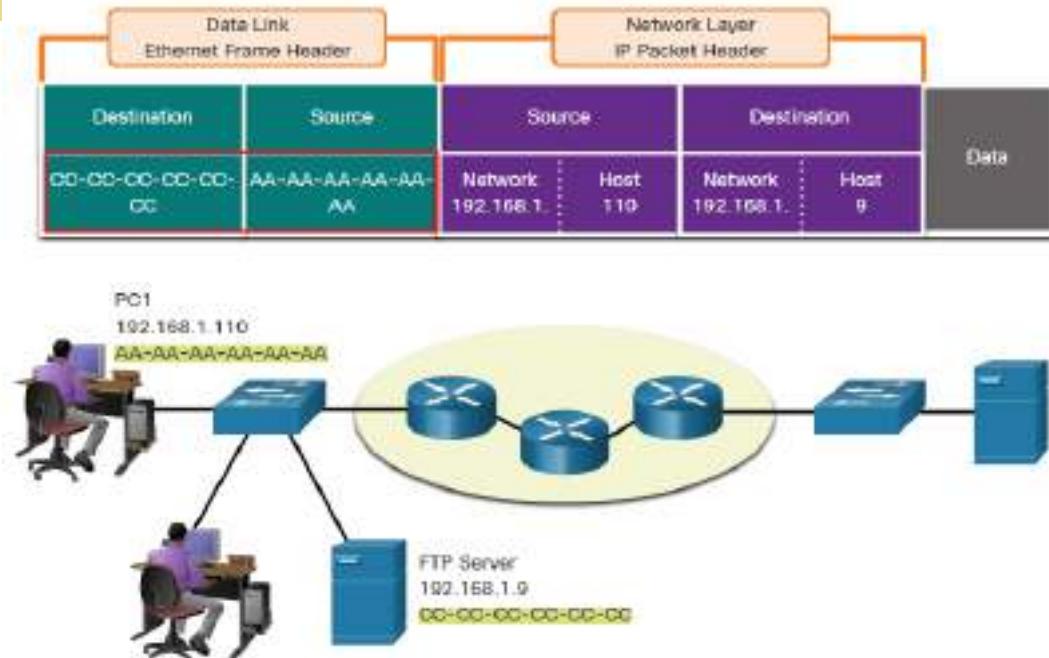
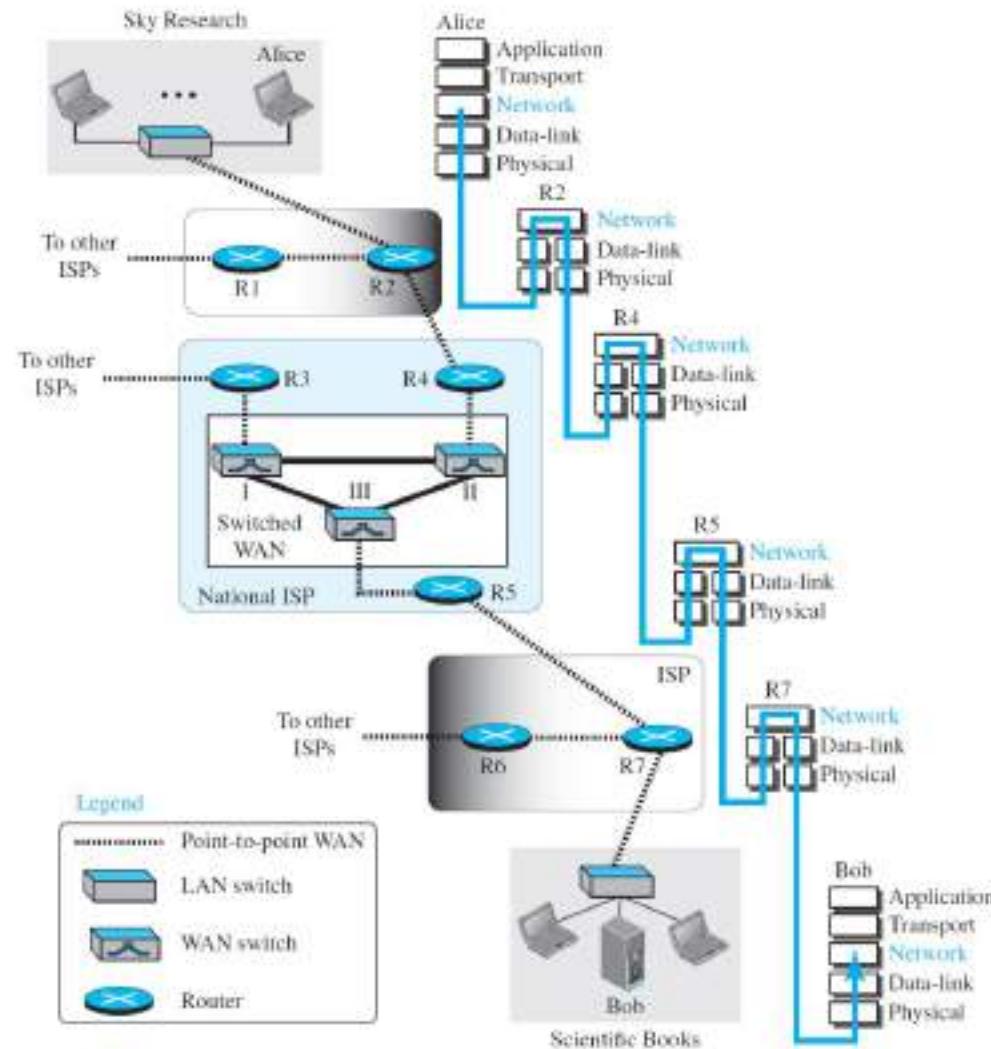


## TCP/IP: Transport Layer Communication





## TCP/IP: Network Layer Communication



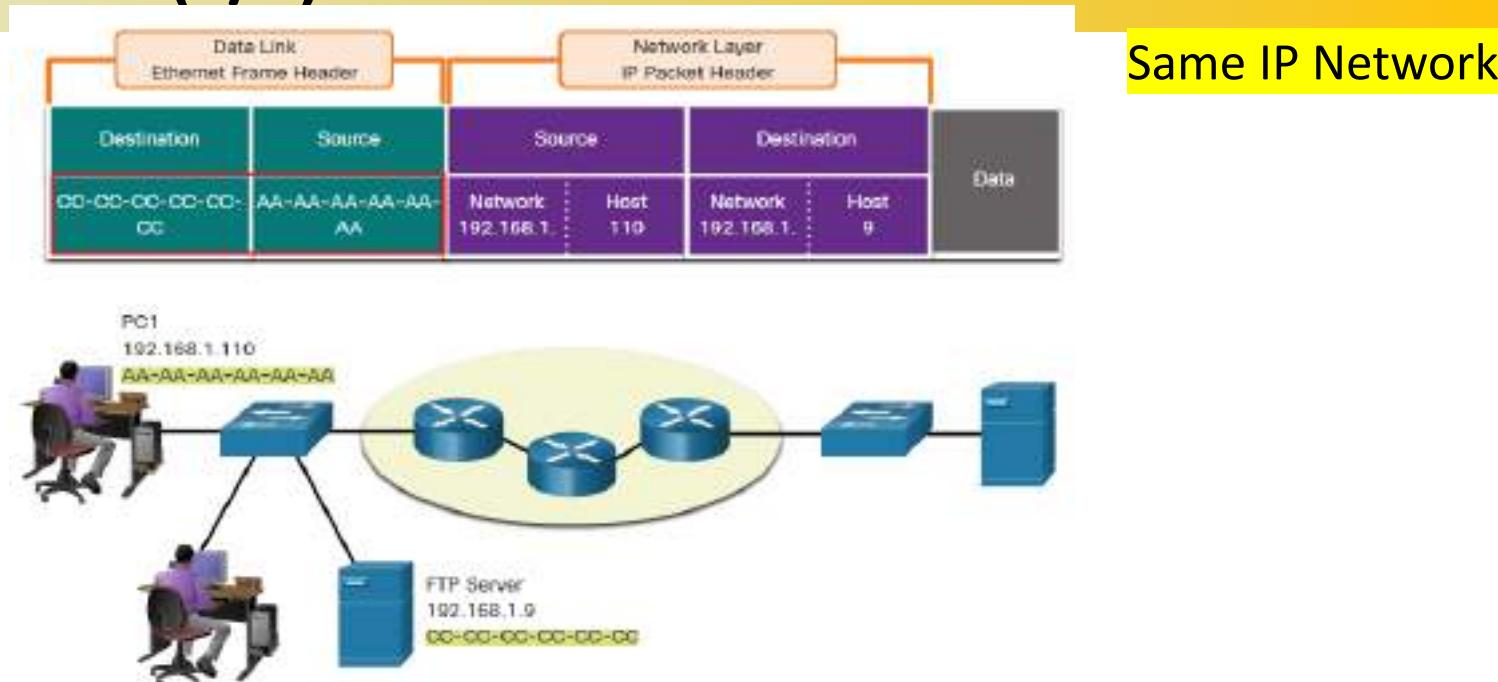
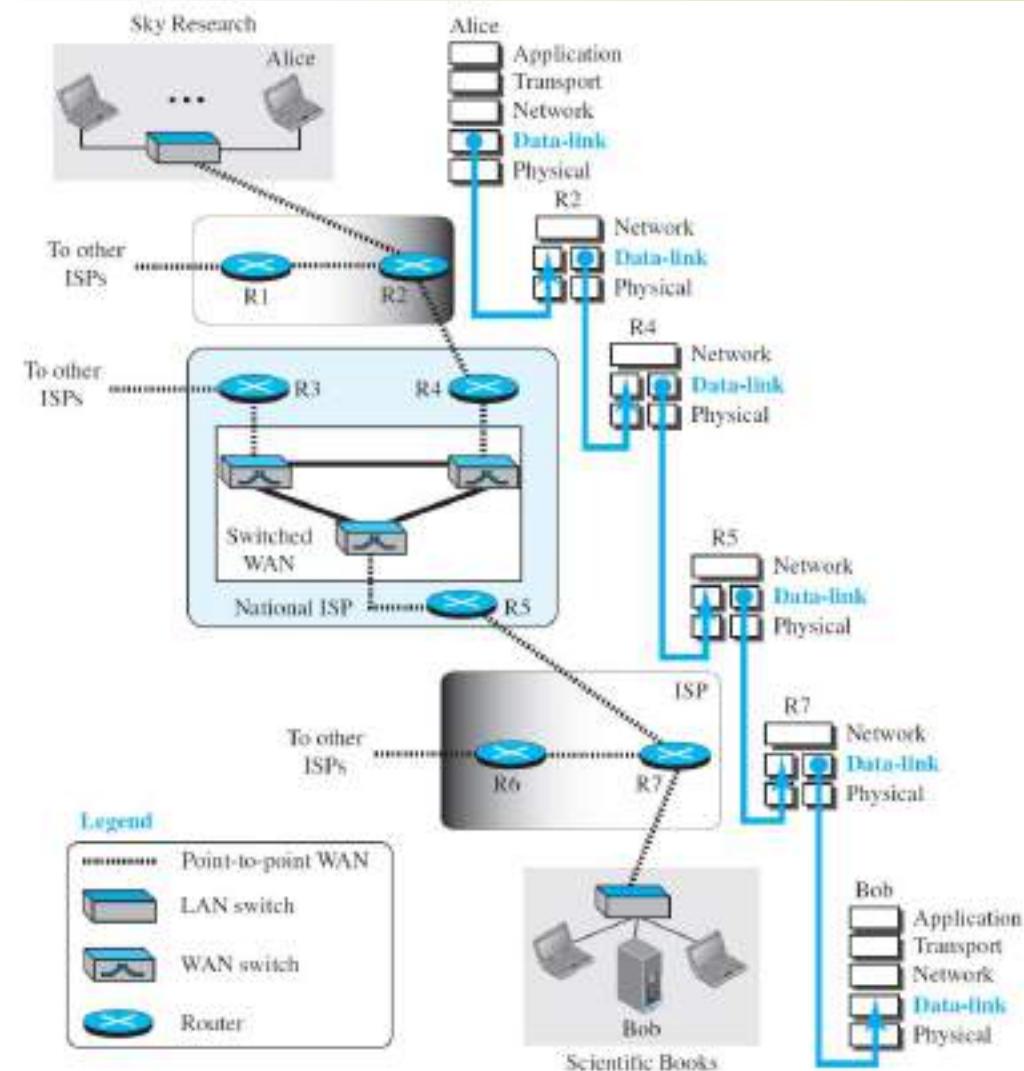
An IP address contains two parts:

- **Network portion (IPv4) or Prefix (IPv6)**
  - The left-most part of the address indicates the network group which the IP address is a member.
  - Each LAN or WAN will have the same network portion.
- **Host portion (IPv4) or Interface ID (IPv6)**
  - The remaining part of the address identifies a specific device within the group.
  - This portion is unique for each device on the network.



# OSI and TCP/IP Networking Models

## TCP/IP: Datalink Layer Communication (1/2)



When devices are on the same Ethernet network the data link frame will use the actual **MAC address** of the **destination NIC**.

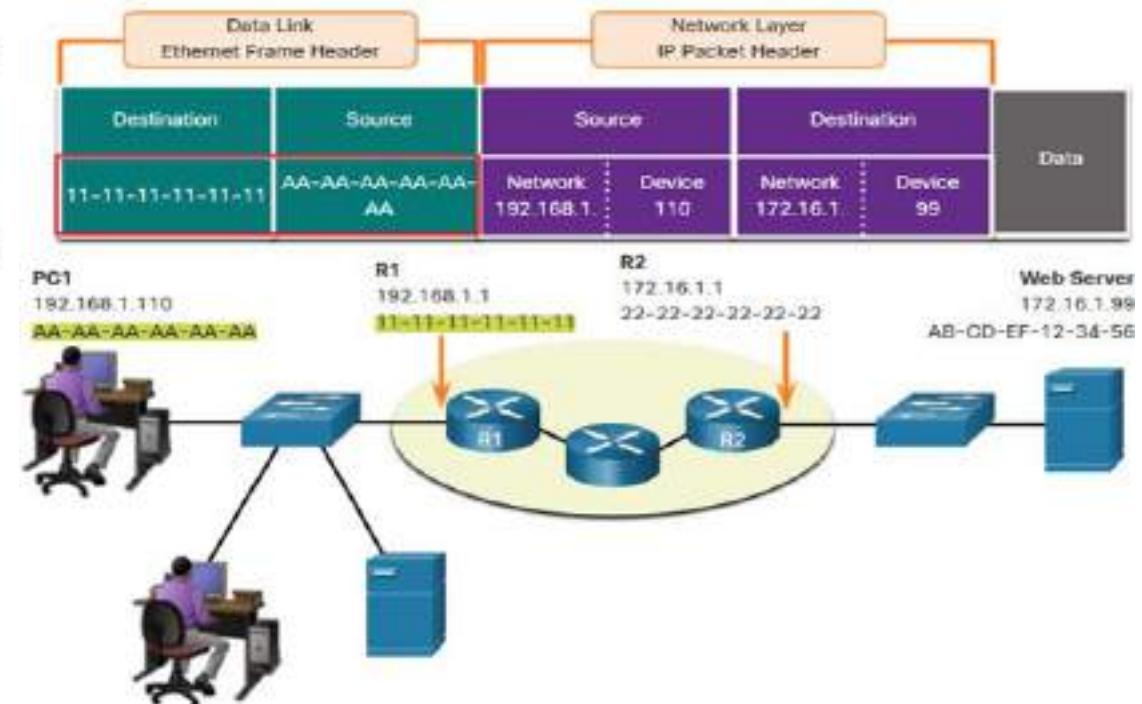
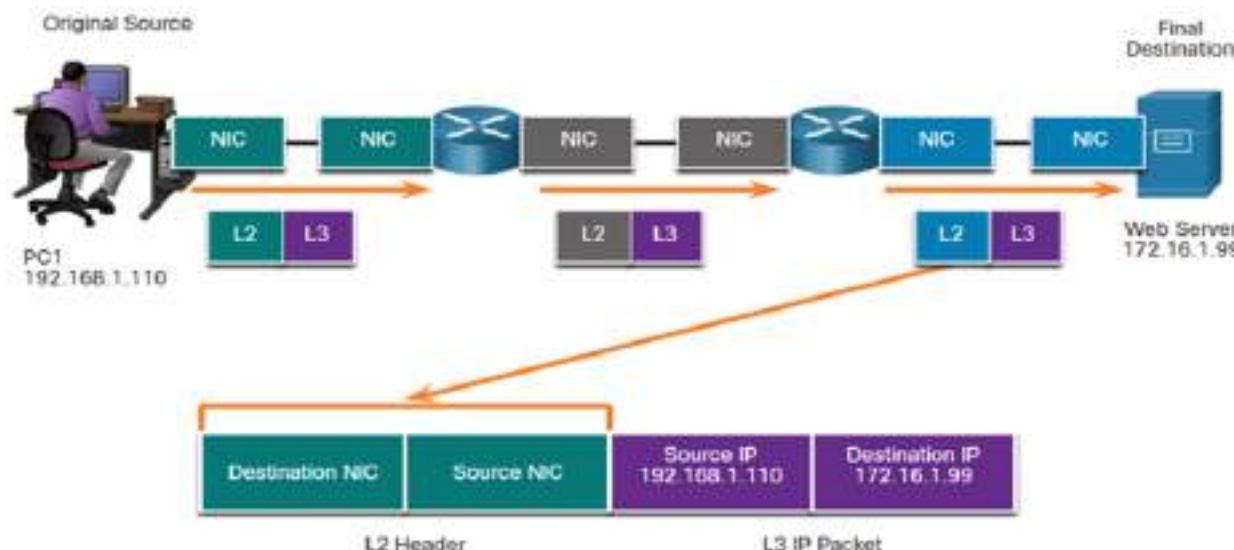
**MAC addresses are physically embedded into the Ethernet NIC and are local addressing.**

- The Source MAC address will be that of the originator on the link.
- The Destination MAC address will always be on the same link as the source, even if the ultimate destination is remote.



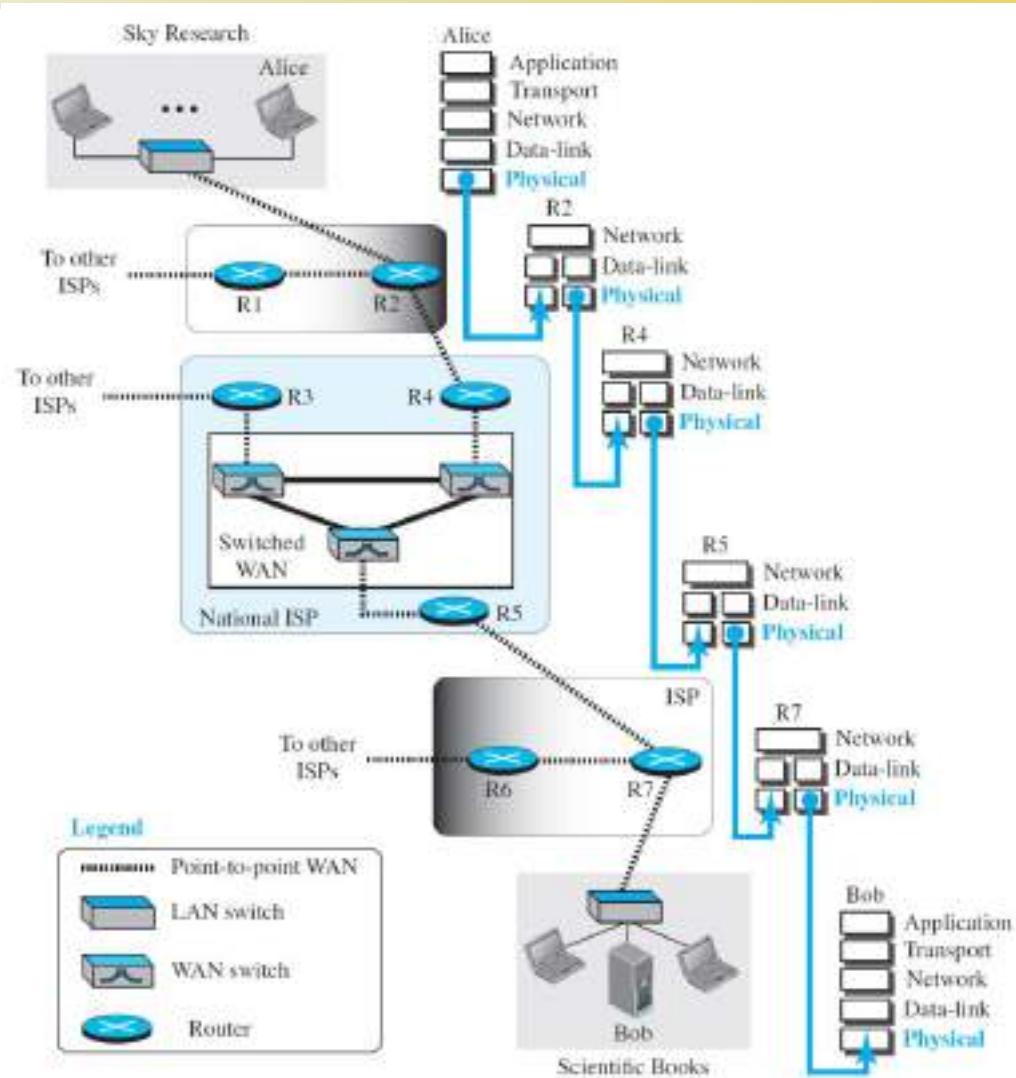
## TCP/IP: Datalink Layer Communication (2/2)

Different IP Networks





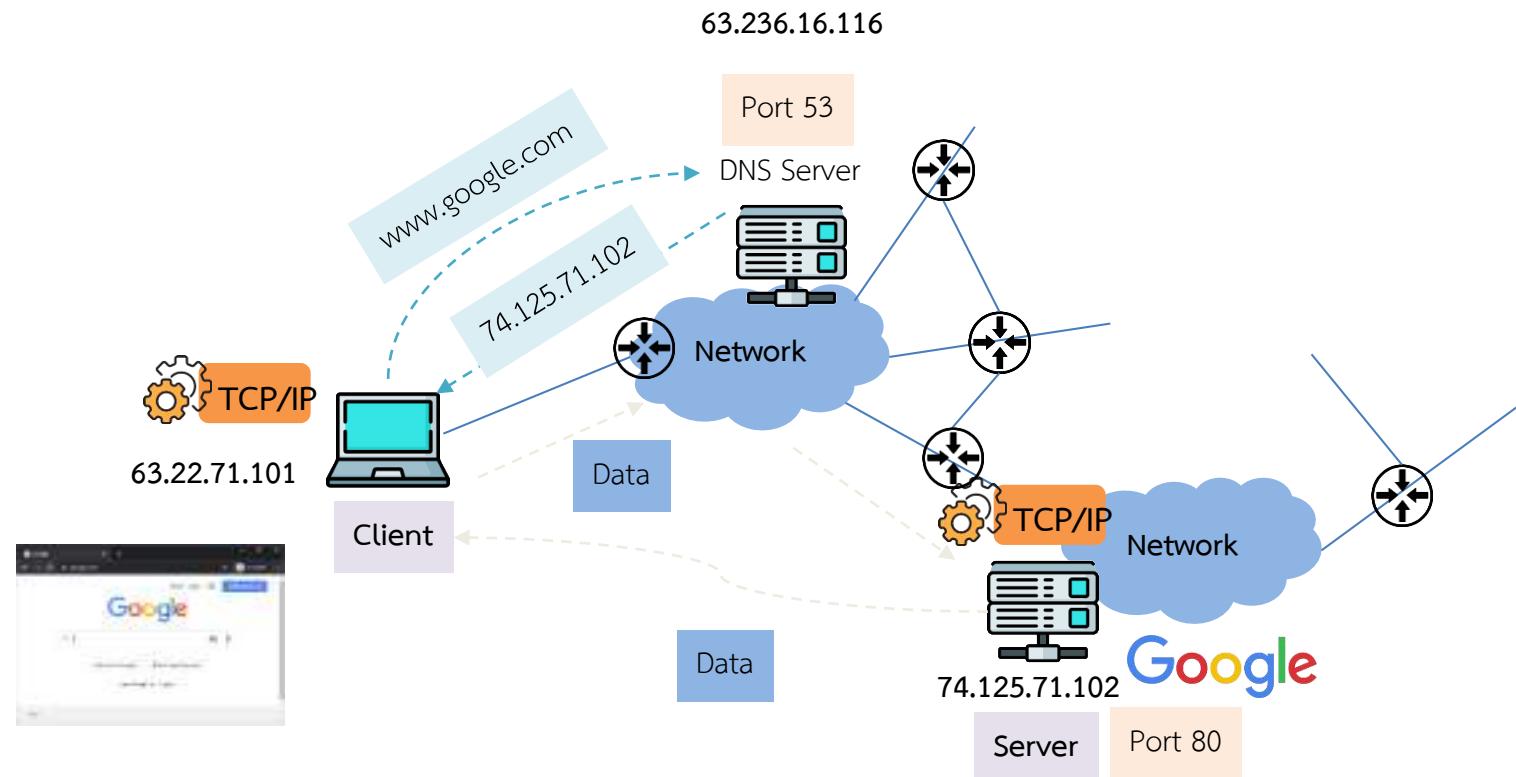
## TCP/IP: Physical Layer Communication





## Domain Name Service (DNS) (1/2)

- To convert from domain name to IP address

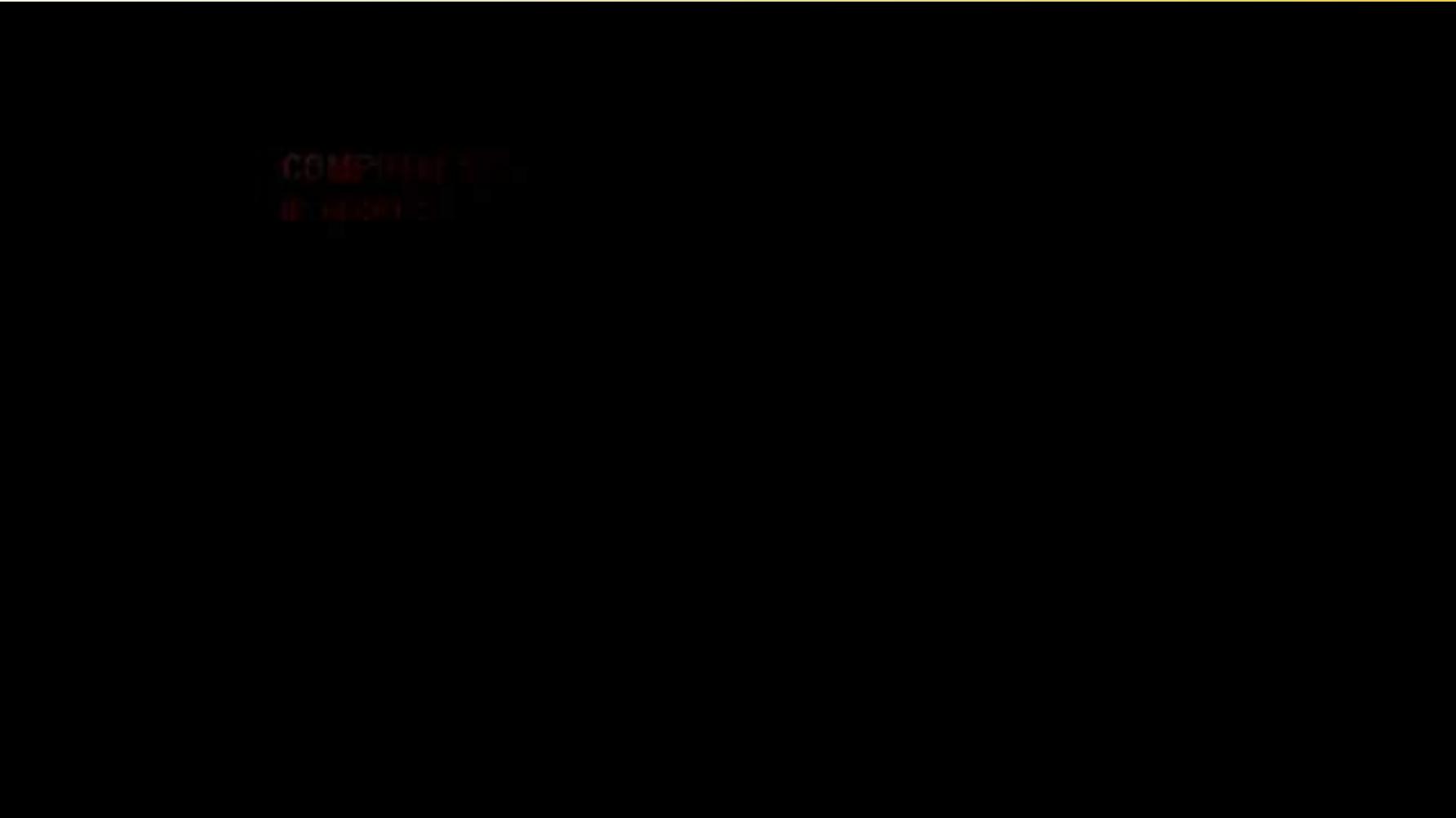


How DNS Works:

[https://www.youtube.com/watch?v=2ZUxoi7YNgs&ab\\_channel=CIRANEWS](https://www.youtube.com/watch?v=2ZUxoi7YNgs&ab_channel=CIRANEWS)



## Domain Name Service (DNS) (2/2)





## IP Packet Flows (WARRIORS OF THE NET)

This clip is for non-commercial use only

<https://www.youtube.com/watch?v=RhvKm0RdUY0>



## Architecture of IoT: Physical Design

### #04 Architecture of IoT: Physical Design (General Blocks)

<https://www.youtube.com/watch?v=8D7OeoWZEK4>



## TCP/IP Protocol Stack in IoT



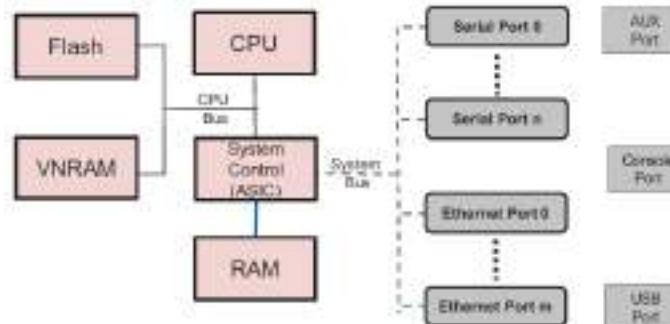
<https://www.youtube.com/watch?v=phkdwmvTq8Q>



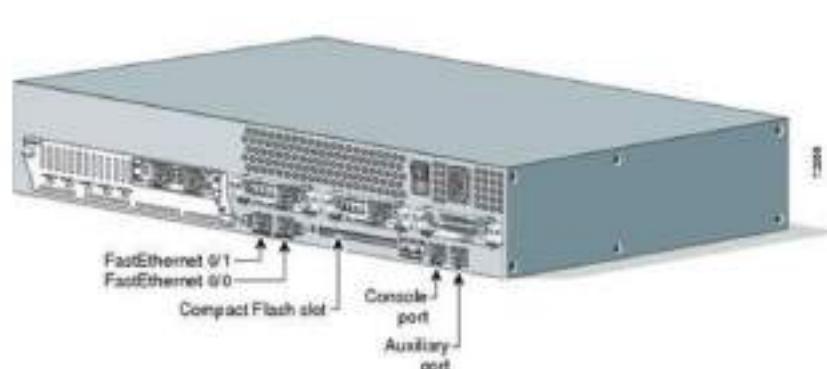
## Network Equipment – Router vs IoT Gateway [CISCO]

- An internet of things (IoT) gateway is a physical device or software program that serves as the connection point between the cloud and controllers, sensors and intelligent devices.
- All data moving between IoT devices and the cloud passes through an IoT gateway, which can be either a dedicated hardware appliance or software program. An IoT gateway might also be referred to as an intelligent gateway or a control tier.

IoT Gateway



| Router component | Main function                                                                                                             | Volatile/nonvolatile |
|------------------|---------------------------------------------------------------------------------------------------------------------------|----------------------|
| CPU              | Executes operating system commands: initialization, routing, and switching functions                                      | Nonvolatile          |
| RAM              | Stores the instruction and data that CPU needs to execute (considered the working area of memory storage used by the CPU) | Volatile             |
| ROM              | Contains code for basic functions to start and maintain the router                                                        | Nonvolatile          |
| Flash            | Permanently stores the operating system (e.g., where a router finds and boots its IOS image)                              | Nonvolatile          |
| NVRAM            | Stores the "startup config" file, holds configuration register settings                                                   | Nonvolatile          |
| Interfaces/ports | Routes are accessed and connected to the external world via the interfaces                                                | N/A                  |



<https://www.techtarget.com/iotagenda/definition/IoT-gateway>

<https://www.cisco.com/c/en/us/products/collateral/wireless/aironet-1815-series-access-points/datasheet-c78-738243.html>

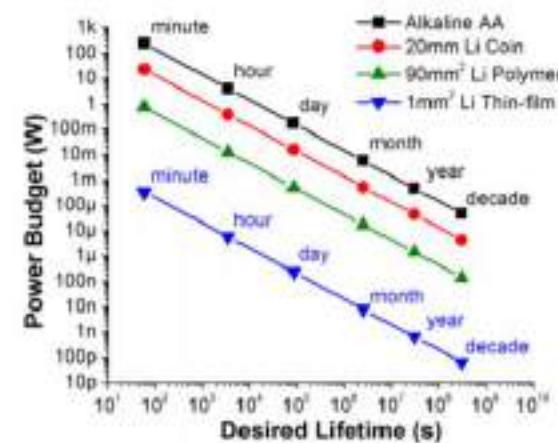


## A Taxonomy of Constrained Devices [IETF-RFC 7228]

- Maximum code complexity (ROM/Flash).
- Size of run-time state and buffers (RAM).
- Amount of computation feasible in a specific period of time (“processing power”).
- Available power resources.
- Management of user interface and accessibility in deployment (ability to set security keys, update software, etc.)

| Specifications           | Class 0                                                    | Class 1                                                                                                                                                     | Class 2                                            |
|--------------------------|------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------|
| RAM                      | << 10kB                                                    | ≈ 10kB                                                                                                                                                      | ≈ 50kB                                             |
| Flash                    | << 100kB                                                   | ≈ 100kB                                                                                                                                                     | ≈ 250kB                                            |
| RTOS support             | Devices do not support RTOS                                | RTOS could be implemented in these devices                                                                                                                  | RTOS can be operated                               |
| Communication protocols  | No protocol stack embedded, use gateways for communication | Communicate via lightweight protocols such as CoAP (Constrained Application Protocol). They can communicate with other devices without the help of gateways | Communication protocols such as HTTP are supported |
| Security Vulnerabilities | Data compromise will result in a basic threat              | Data compromise will result in medium threat                                                                                                                | Data compromise will result in medium/high threat  |

| Name    | Data size | Code size |
|---------|-----------|-----------|
| Class 0 | <<10 KB   | <<100 KB  |
| Class 1 | ~10 KB    | ~100 KB   |
| Class 2 | ~50 KB    | ~250 KB   |

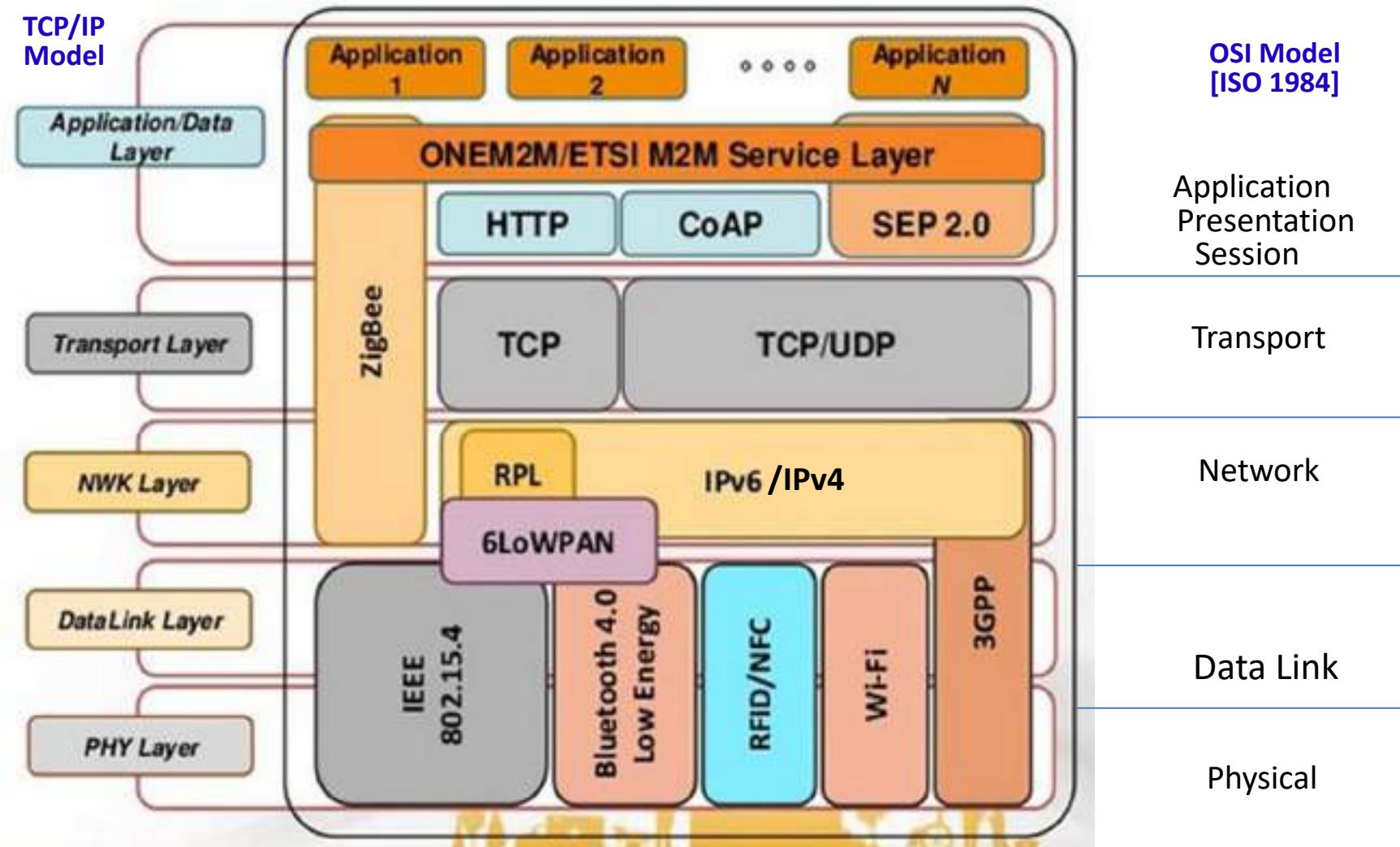


<https://tools.ietf.org/html/rfc7228>

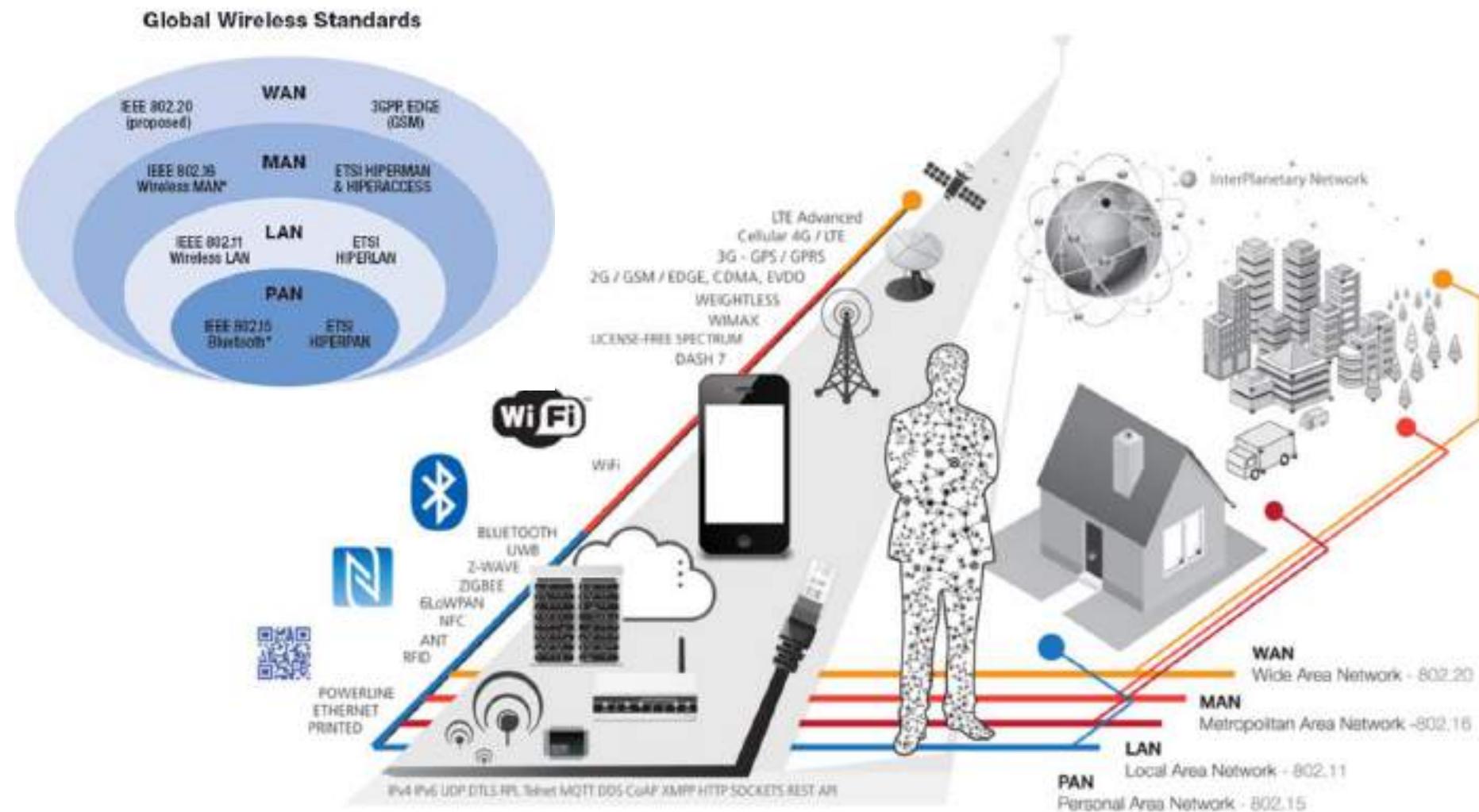
<https://arpi.unipi.it/retrieve/handle/11568/939859/360671/low-end-preprint.pdf>

<https://www.cisco.com/c/en/us/products/collateral/wireless/aironet-1815-series-access-points/datasheet-c78-738243.html>

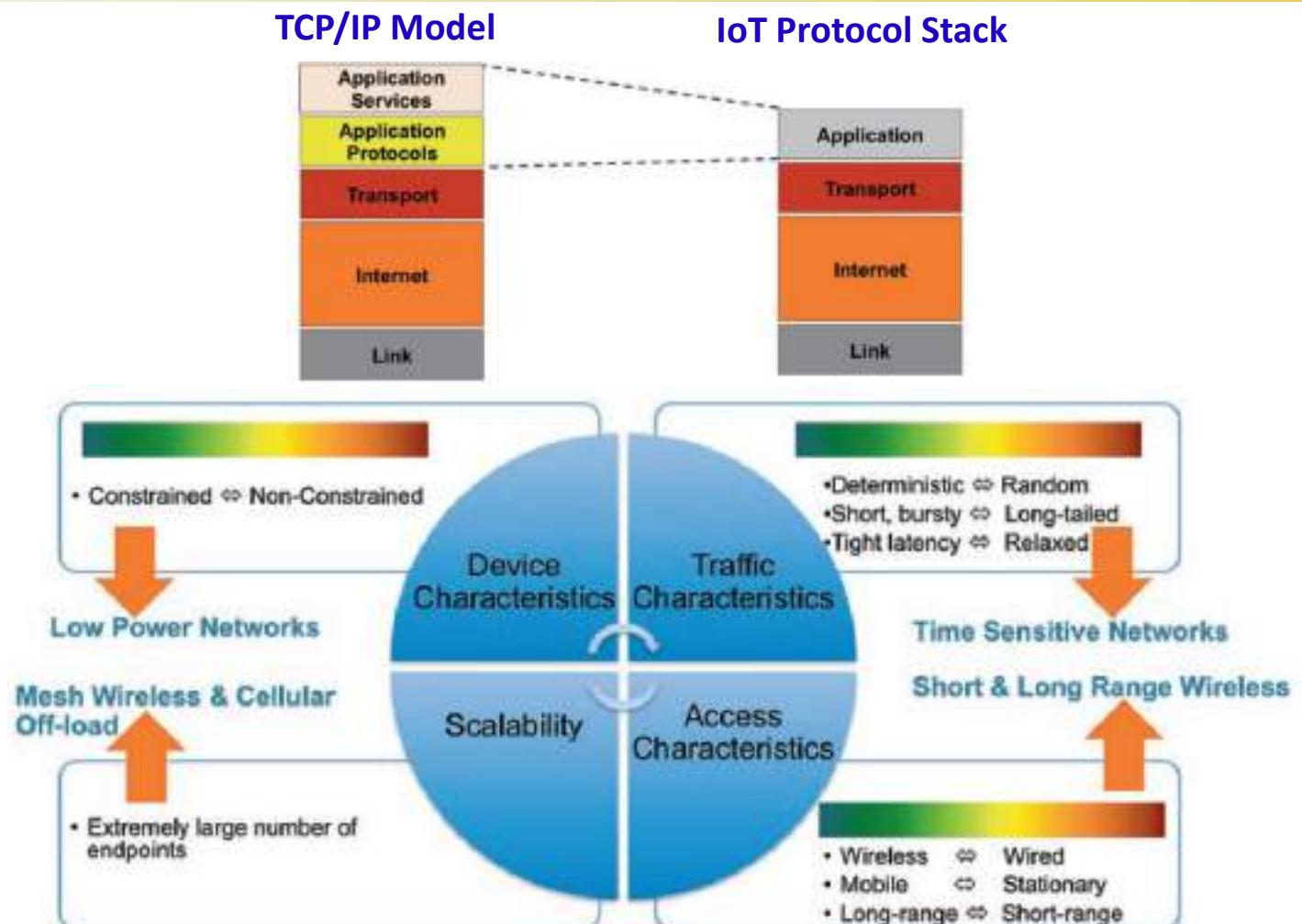
## TCP/IP vs OSI (Partial List)



## Wireless Connectivity and Network Size

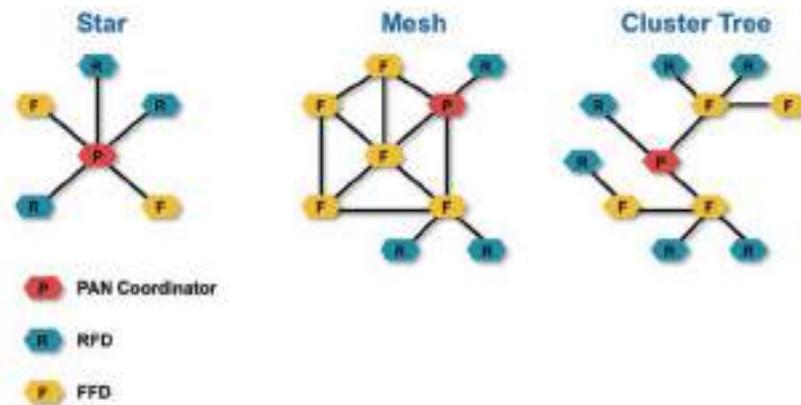


# IoT Protocol Stack: Link Layer





## Personal Area Network (IEEE 802.15.4)



| Protocol      | Range       | Data rate       | Topology | Application                        | Power consumption |
|---------------|-------------|-----------------|----------|------------------------------------|-------------------|
| IEEE 802.15.4 | Up to 1 km  | 1Mbps to 10Kbps | Mesh     | Personal area network/home network | Very low          |
| LoRaWAN       | Up to 20 km | Up to 50Kbps    | Star     | Wide area network                  | Low               |
| IEEE 802.11ah | Up to 1 km  | >100Kbps        | Star     | Metropolitan block                 | Medium            |

### Two types of devices in an 802.15.4

#### FFD: Full-Function Device

Allows to communicate with other device in the network. It may relay messages.

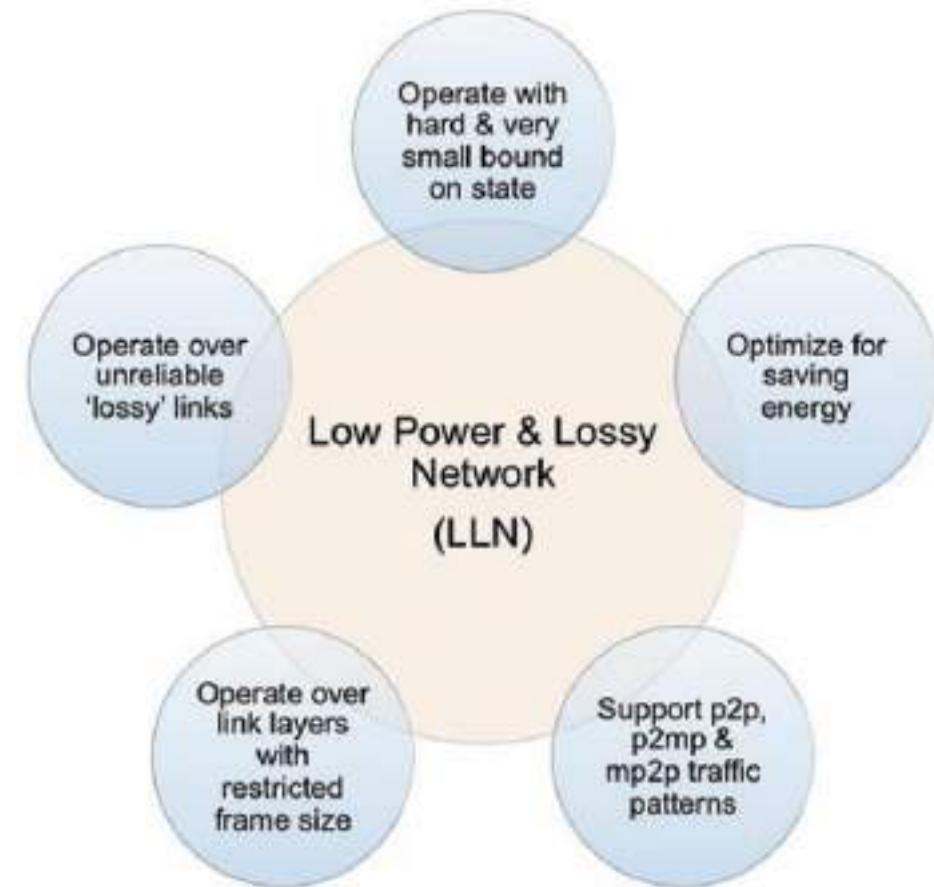
#### RFD: Reduced-Function Device

Extremely simple devices are to be embedded into the things.

# IoT Protocol Stack: Internet Layer



- Many IoT deployments constitute what is referred to as low-power and lossy networks (LLNs).
- The LLNs comprise of a large number (several thousand) of constrained embedded devices with limited power, memory, and processing resources, with a variety of link layer technologies, such as IEEE802.15.4, Bluetooth, Wi-Fi, or Power-Line Communication (PLC) links.

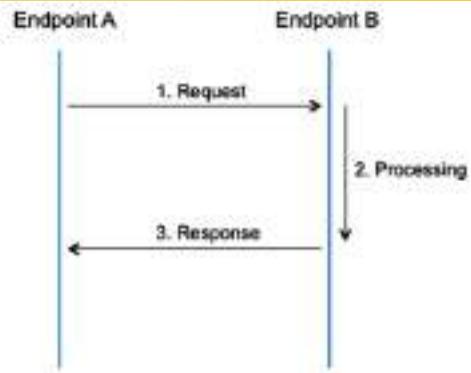




## 2 Communication Paradigms

- **Request/Response Paradigm**

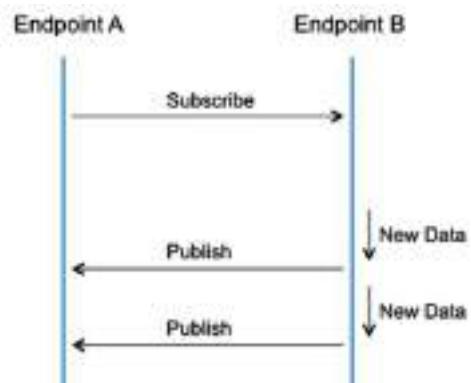
- The deployment follows a client-server architecture.
- The deployment requires interactive communication: both endpoints have information to send to the other side.
- The receipt of information needs to be fully acknowledged (e.g., for reliability).



- **Publish/Subscribe Paradigm**

**Not all IoT devices can be interactive (like sensors or actuators)**

- Loose coupling between the communicating endpoints, especially when compared with the client-server model.
- Better scalability by leveraging parallelism and the multicast capabilities of the underlying transport network.



See next Lecture: IoT Application Protocols



## IoT Application Protocols

| Protocol   | Functions                                                                                                                      | Primary use                                                | Transport      | Format | SDO         |
|------------|--------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------|----------------|--------|-------------|
| CoAP       | REST resource manipulation via CRUD<br>Resource tagging with attributes<br>Resource discovery through RD                       | LLNs                                                       | UDP            | Binary | IETF        |
| XMPP       | Manage presence<br>Session establishment<br>Data transfer (text or binary)                                                     | Instant messaging                                          | TCP<br>HTTP    | XML    | IETF<br>XSF |
| MQTT       | Lightweight pub/sub messaging<br>Message queuing for future subscribers                                                        | Enterprise telemetry                                       | TCP            | Binary | OASIS       |
| AMQP       | Message orientation, queuing and pub/sub<br>Data transfer with delivery guarantees (at least once, at most once, exactly once) | Financial services                                         | TCP            | Binary | OASIS       |
| SIP        | Manage presence<br>Session establishment<br>Data transfer (voice, video, text)                                                 | IP telephony                                               | TCP, UDP, SCTP | XML    | IETF        |
| IEEE 1888  | Read/write data into URI<br>Handling time-series data                                                                          | Energy and facility management                             | SOAP/<br>HTTP  | XML    | IEEE        |
| DDS (RTPS) | Pub/sub messaging with well-defined data types<br>Data discovery<br>Elaborate QoS                                              | Real-time distributed systems (military, industrial, etc.) | UDP            | Binary | OMG         |

Learn more next Lecture: IoT Application Protocols



## Technology Gaps on Application Service Layer

### Search and discovery capabilities:

- Mechanisms by which devices as well as applications can **automatically discover** each other as well as discover middleware/common services nodes.
- **Mechanisms by which applications can search for devices with specific attributes** (e.g., sensors of particular type) or context (e.g., within a specific distance from a location).
- Mechanisms by which applications can search for data based on attributes (e.g., semantic annotations) or context (e.g., spatial or temporal).

### Data encoding, interpretation, and modeling

- Mechanisms that render IoT data understandable to applications without a priori knowledge of the data or the devices that produced it.
- **Mechanisms that enable application interaction at a high level of abstraction by means of physical/virtual entity modeling.**
- Mechanisms that enable data management services to host the semantic description of IoT data that is being handled.
- **Framework for defining formal domain-specific semantic models or ontologies, including but not limited to defining an upper-level ontology for IoT.**

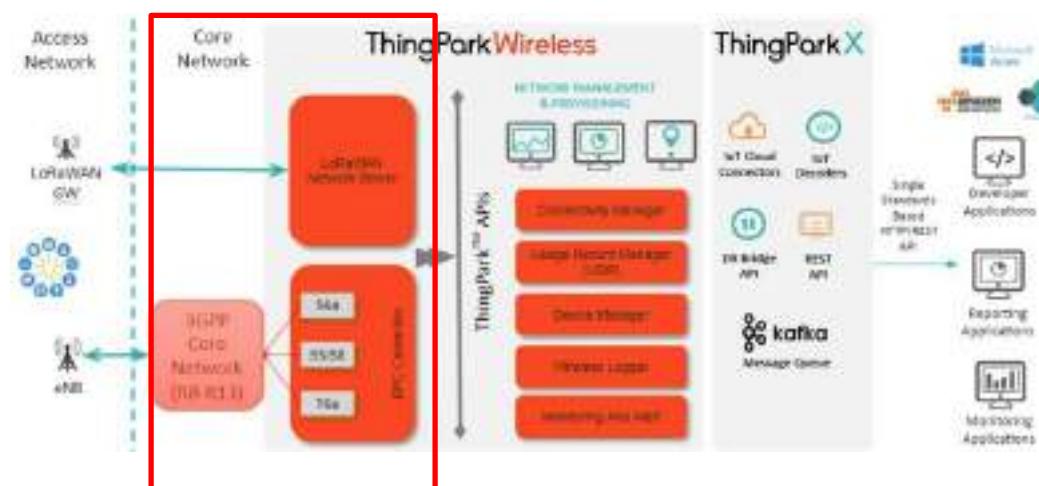
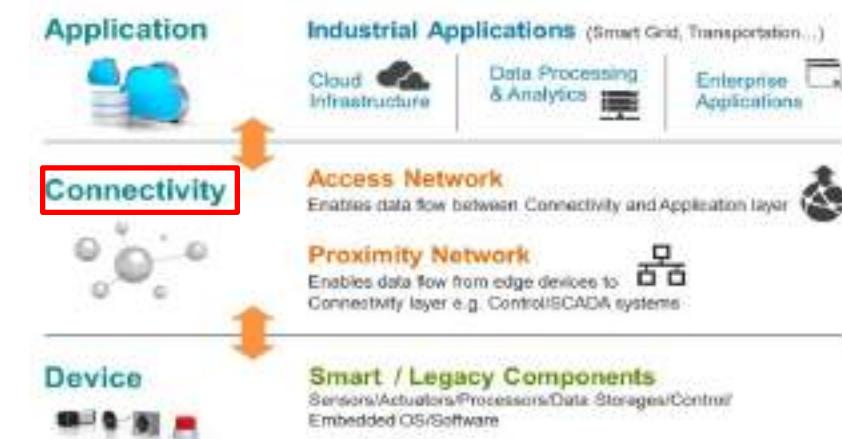
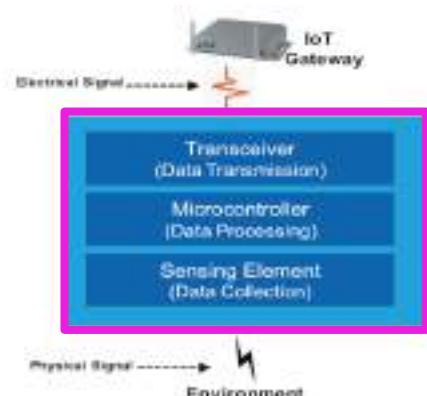
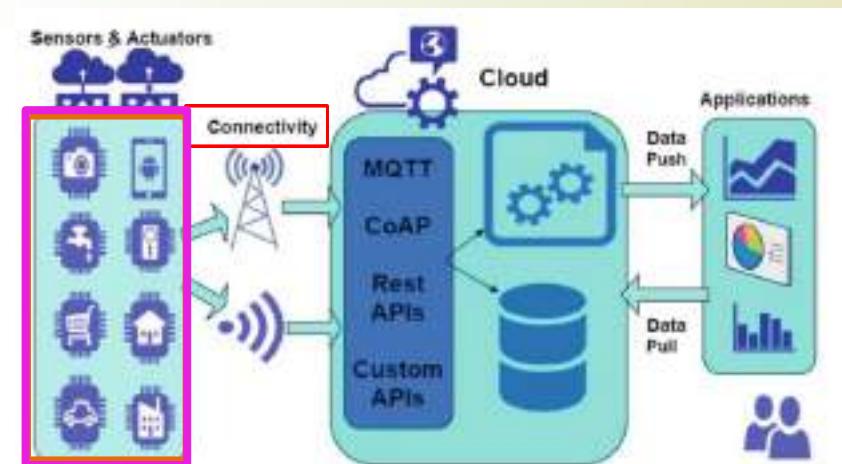
# IoT Networking Considerations and Challenges



- Scalability: Millions of devices
  - IPv6/IPv4 (DHCP/NAT) addresses
  - tons of data
  - Legacy support: non-IP, specialized devices, multiple vertical solutions
- Range: PAN/LAN/MAN/WAN
- Bandwidth
  - The volume of data each device gathers and transmits
  - The number of devices deployed
  - Whether data is being sent as a constant stream or intermittent bursts, and if any peak periods are notable.
  - Intermittent connectivity: either planned or unexpected link disruptions
- Constrained Devices (lost cost): lossy networks, low bandwidth, small batteries, ...
- Need for Real Time: real time streaming data analysis, proactive action, ..
- Different IoT reference architectures: ETSI, ITU, IEEE, NIST, ISO, oneM2M Alliance, ISA, ...
- Security: Private/Public Data travels through public networks
  - Authentication, Encryption, Port Protection

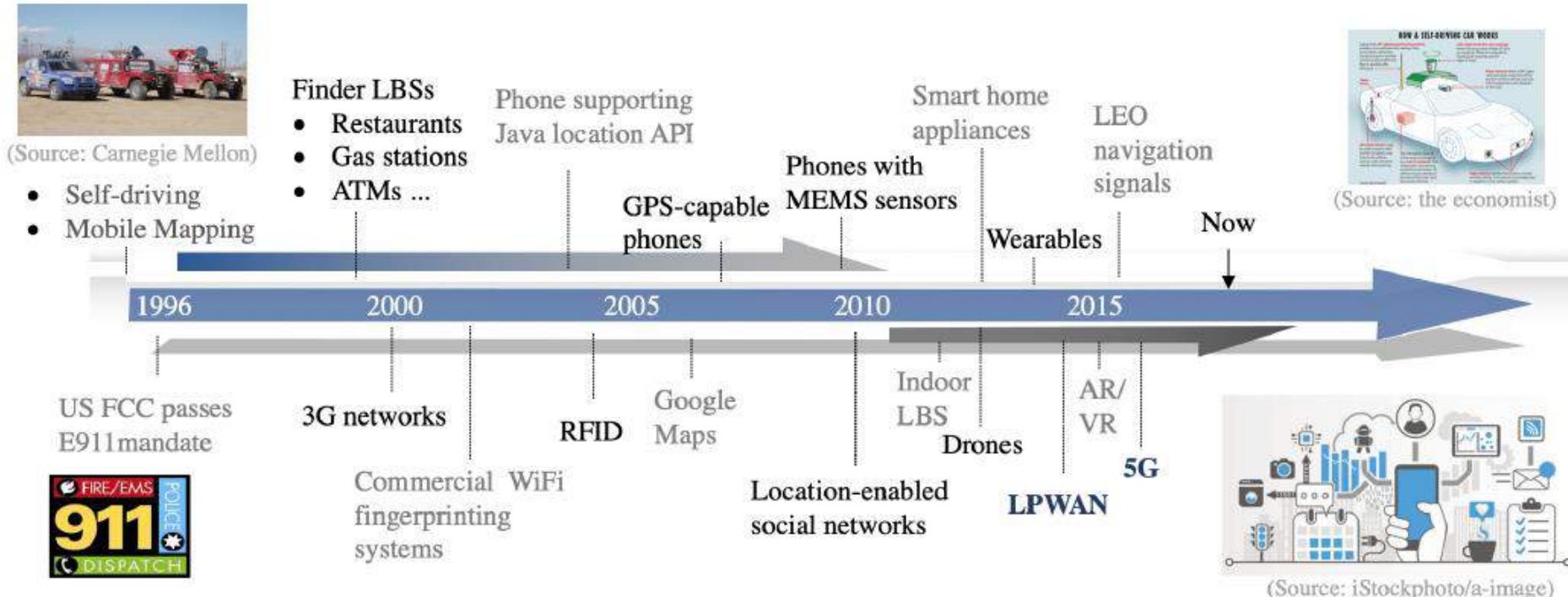
Read more: <https://developer.ibm.com/technologies/iot/articles/iot-ip101-connectivity-network-protocols/>

# IoT Connectivity vs Access Networks





## Location-based Applications



# IoT Connectivity vs Access Networks



## Next-Generation Wireless Technologies

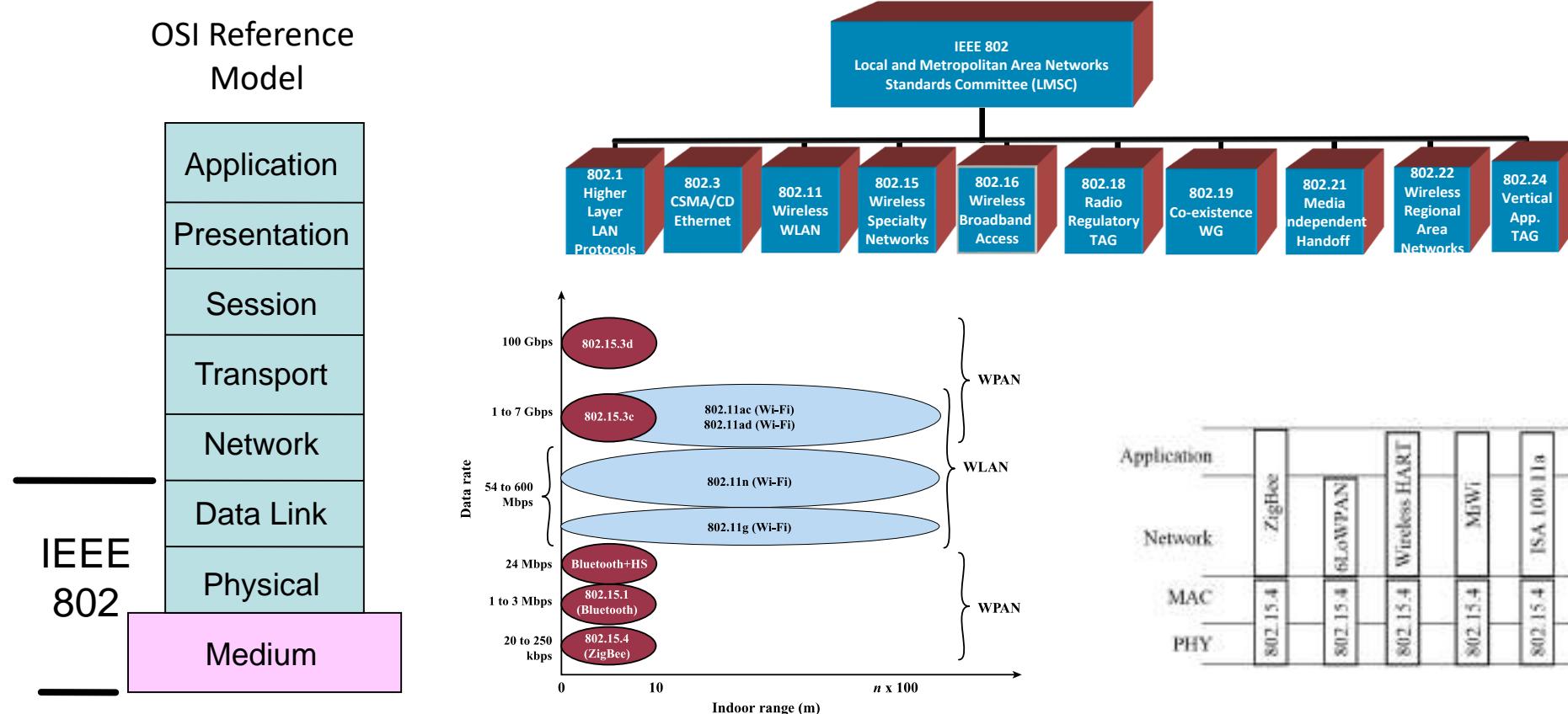
|  | Frequency                     | Range  | Data rate  | Energy consumption | End nodes per gateway | Availability                           |
|--|-------------------------------|--------|------------|--------------------|-----------------------|----------------------------------------|
|  | 125 kHz – 960 MHz             | 100 m  | 640 kBit/s | Very low           | 1                     | Dedicated infrastructure               |
|  | 13.56 MHz                     | 10 cm  | 430 kBit/s | Low                | 1                     | Every smartphone                       |
|  | 2.4 GHz                       | 350 m  | 1 MBit/s   | Very low           | Unlimited             | Every smartphone & some gateways       |
|  | 1, 2.4, 5 & 6 GHz             | 50 m   | 10 GBit/s  | Medium             | 250                   | Every smartphone & new infrastructure  |
|  | 410 - 5900 MHz                | 10 km  | 1 GBit/s   | Low                | 600                   | Public deployment                      |
|  | 2.5 GHz - 49 GHz              | 500 m  | 50 GBit/s  | Very high          | 1 million             | Limited public deployment              |
|  | 500 - 5900 MHz                | 100 km | 375 kBit/s | Low                | Not available         | All cellular networks country-specific |
|  | 698 - 2170 MHz                | 40 km  | 250 kBit/s | Low                | ca. 20,000            | All cellular networks country-specific |
|  | 490 MHz<br>868 MHz<br>915 MHz | 50 km  | 300 Bit/s  | Very low           | 1 million             | 157 countries                          |
|  | 433 MHz<br>915 MHz            | 50 km  | 100 Bit/s  | Very low           | 1 million             | 70 countries                           |
|  | 400 MHz                       | Global | 144Bit/s   | Medium             | Not applicable        | Global                                 |

Some specifications may vary by countries or future releases.



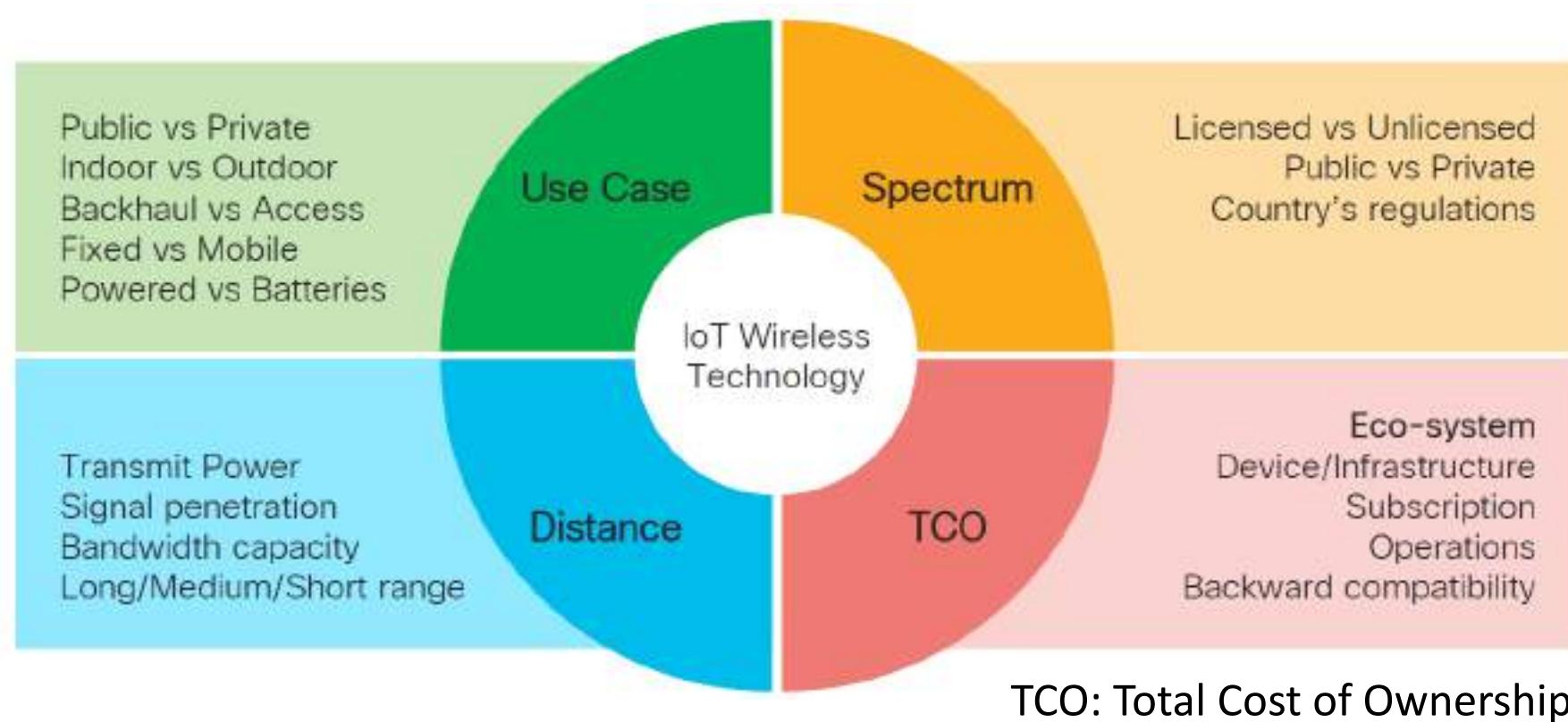
# IoT Connectivity vs Access Networks

## IEEE 802





## Industrial IoT Wireless Selection Criteria

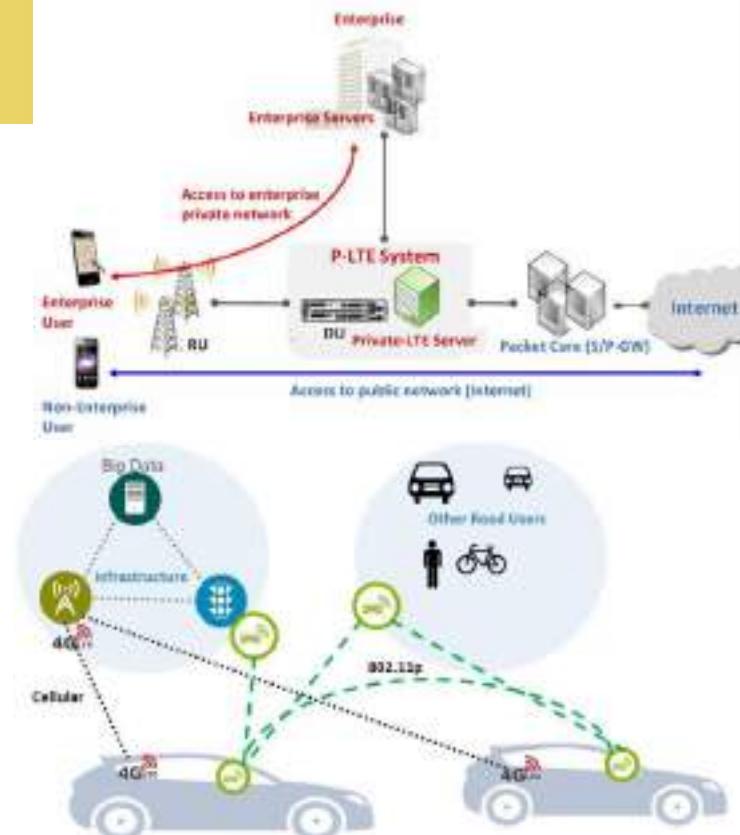


# IoT Connectivity vs Access Networks



## Use Cases for IoT Wireless

| From bits/sec to gigabits/sec                       |                                                                                                                                                                      |                                                                                            |
|-----------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|
| Industries                                          | Use Cases                                                                                                                                                            | Wireless Technologies<br>Access (A) or Backhaul (B)                                        |
| Manufacturing,<br>Warehouse,<br>Distribution Center | Industrial automation, industrial security, plant efficiency, workforce enablement                                                                                   | LoRaWAN (A), Wi-Fi (A/B), 4G (B), 5G (B)                                                   |
| Transportation                                      | Passenger experience, data operations, operational efficiency, safety and compliance, traffic operations, roadway safety, sustainable mobility, sensor modernization | LoRaWAN (A), Wi-Fi (A/B), DSRC (A)<br>4G (B), 5G (A/B)                                     |
| Cities                                              | Cities operations, public safety and security, citizen services, economic sustainability                                                                             | LoRaWAN (A), Resilient Mesh (A), Wi-Fi (A/B), 4G (B), 5G (B)                               |
| Mining                                              | Field operations, industrial security, workforce enablement                                                                                                          | LoRaWAN, (A) WirelessHart (A),<br>ISA100.11a (A), Wi-Fi (A/B), 4G (B), p-LTE (A/B), 5G (B) |
| Oil & Gas                                           | Plant and field operations, industrial security, workforce enablement                                                                                                | LoRaWAN, (A) WirelessHart (A),<br>ISA100.11a (A), Wi-Fi (A/B), 4G (B), p-LTE (A/B), 5G (B) |
| Utilities                                           | Connected substations, distribution grid management, workforce enablement, grid safety, production plants                                                            | LoRaWAN (A), Resilient Mesh (A), Wi-Fi (A/B), 4G (B), P-LTE (B), 5G (B)                    |



P-LTE: Private 4G  
LTE: Long Term Evolution

[https://www.cisco.com/c/m/en\\_us/solutions/industries/portfolio-explorer.html](https://www.cisco.com/c/m/en_us/solutions/industries/portfolio-explorer.html)

[https://en.wikipedia.org/wiki/List\\_of\\_mobile\\_phone\\_generations](https://en.wikipedia.org/wiki/List_of_mobile_phone_generations)

[https://en.wikipedia.org/wiki/IEEE\\_802.11](https://en.wikipedia.org/wiki/IEEE_802.11)

<https://www.zdnet.com/article/what-is-v2x-communication-creating-connectivity-for-the-autonomous-car-era/>

<https://www.fcc.gov/media/radio/public-and-broadcasting>

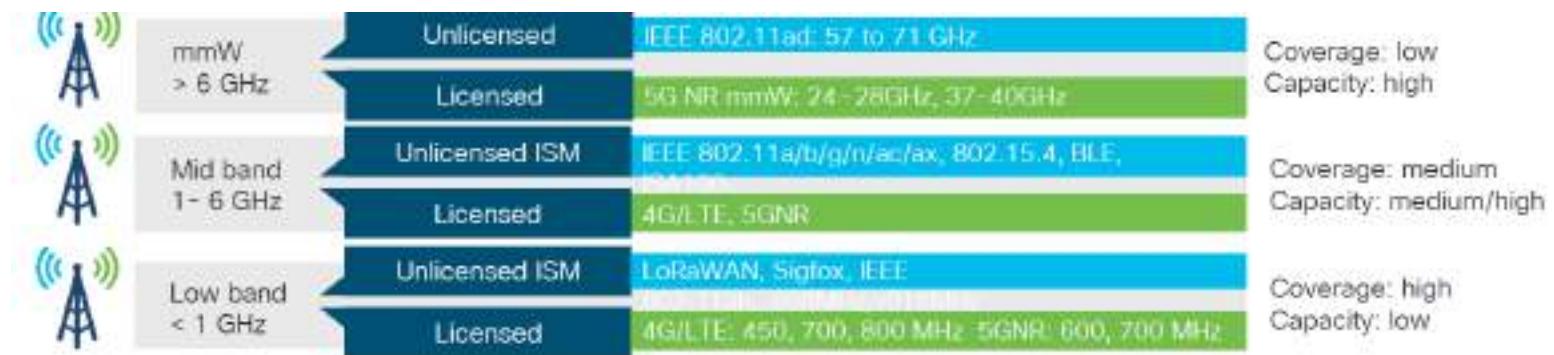


## Spectrum is a Scarce Resource

Managed by World organizations/Countries strongly regulated: Transmit power, duty cycle...

Spectrum types in IoT Wireless

- **Unlicensed:** also refer as **ISM (Industrial, Scientific and Medical) bands**, generally **free** of charge, public, and private infrastructures, but regulated.
  - Shared between technologies; co existence definition in specifications
- **Licensed:** dedicated to **SP (public services)** or **industries (private), not free**, may be reallocated.
  - Introducing Shared licensed model.

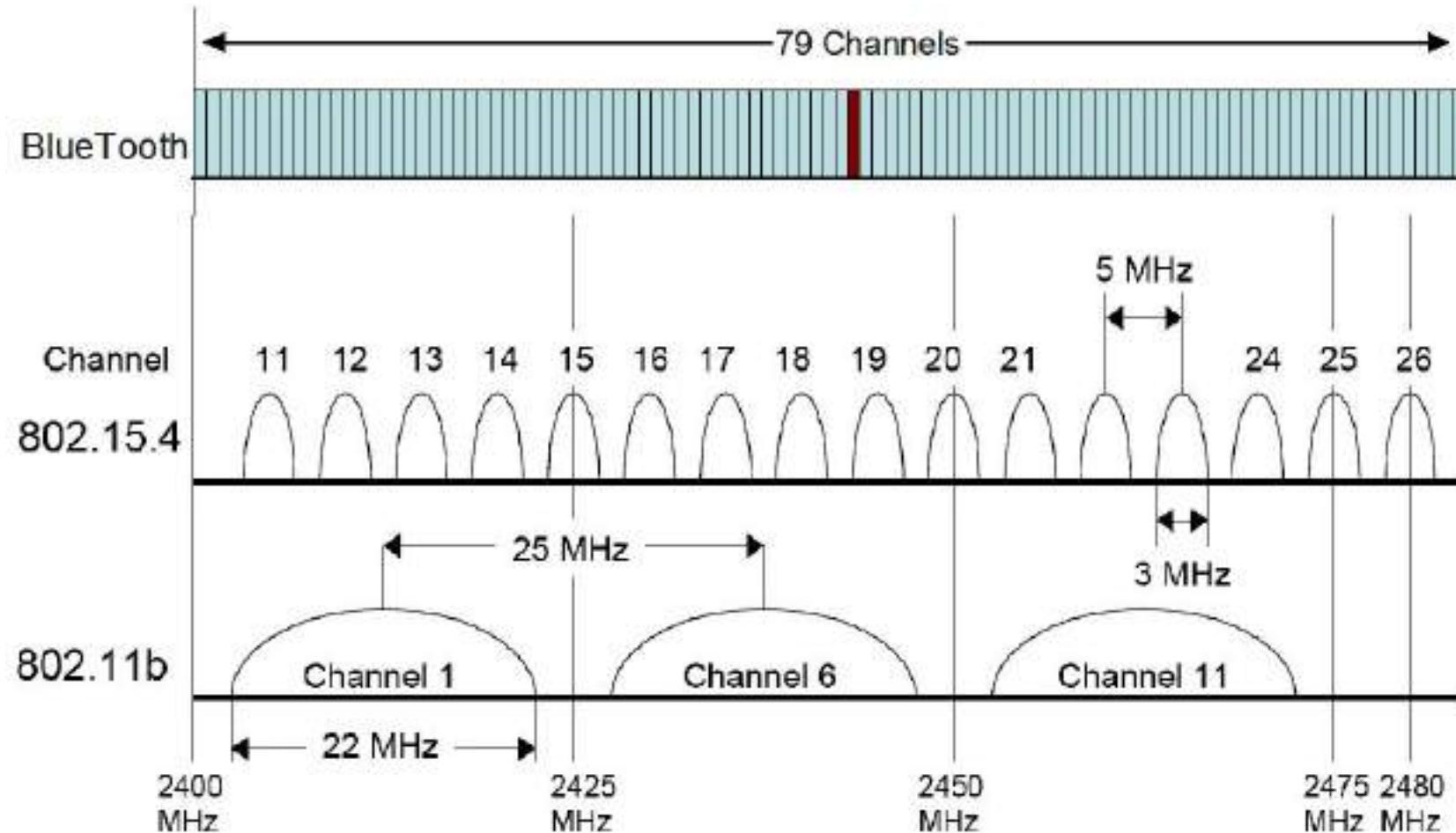


[https://en.wikipedia.org/wiki/ISM\\_band](https://en.wikipedia.org/wiki/ISM_band)

# IoT Connectivity vs Access Networks



## Example: ISM Band



# IoT Connectivity vs Access Networks



## Shared Licensed: Radio-Frequency Spectrum



THE NATIONAL BROADCASTING AND  
TELECOMMUNICATIONS COMMISSION  
(NBTC) www.nbtc.go.th

**VLF**  
3 kHz  
100 km

9 kHz

14 kHz

RADIO NAVIGATION

FIXED

STANDARD FREQUENCY AND TIME SIGNAL

MARITIME MOBILE

70 kHz

160 kHz

190 kHz

**LF**  
300 kHz  
1 km

**MF**  
300 kHz  
1 km

526.5 kHz

1605.5 kHz

1800 kHz

**MF**  
3 MHz  
100 m

**HF**  
3 MHz  
100 m

1000 kHz

1000 kHz

1000 kHz

**HF**  
30 MHz  
10 m

**VHF**  
30 MHz  
10 m

74.8 MHz

87 MHz

108 MHz

117.975 MHz

137 MHz

**VHF**  
300 MHz  
1 m

148 MHz

174 MHz

230 MHz

**UHF**  
300 kHz  
1 m

510 MHz

890 MHz

960 MHz

**UHF**  
3 GHz  
100 m

1164 MHz

**SHF**  
30 GHz  
100 mm

2200 MHz

2700 MHz

**SHF**  
300 GHz  
100 mm

**SHF**  
3 GHz  
100 mm

4.2 GHz

5.925 GHz

6.7 GHz

10.7 GHz

11.7 GHz

**EHF**  
300 GHz  
1 mm

41 GHz

105 GHz

275 GHz

**EHF**  
300 GHz  
10 mm

**EHF**  
30 GHz  
10 mm

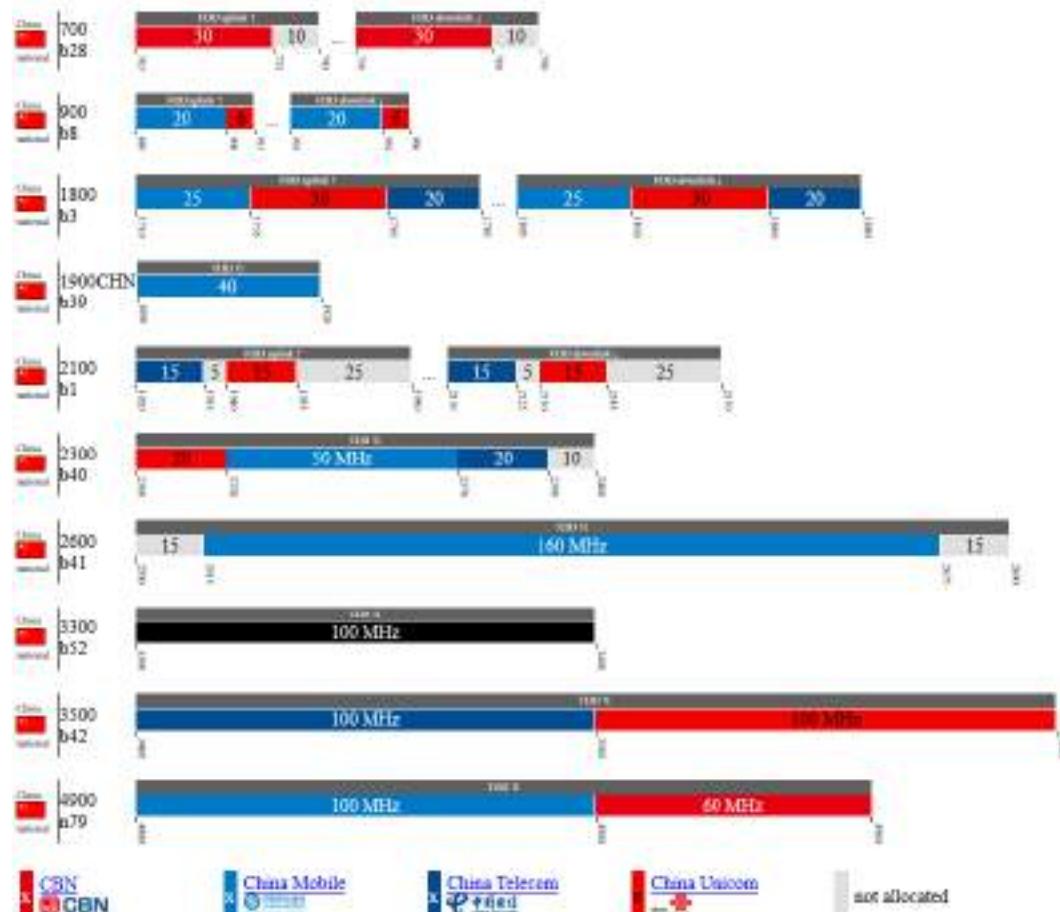
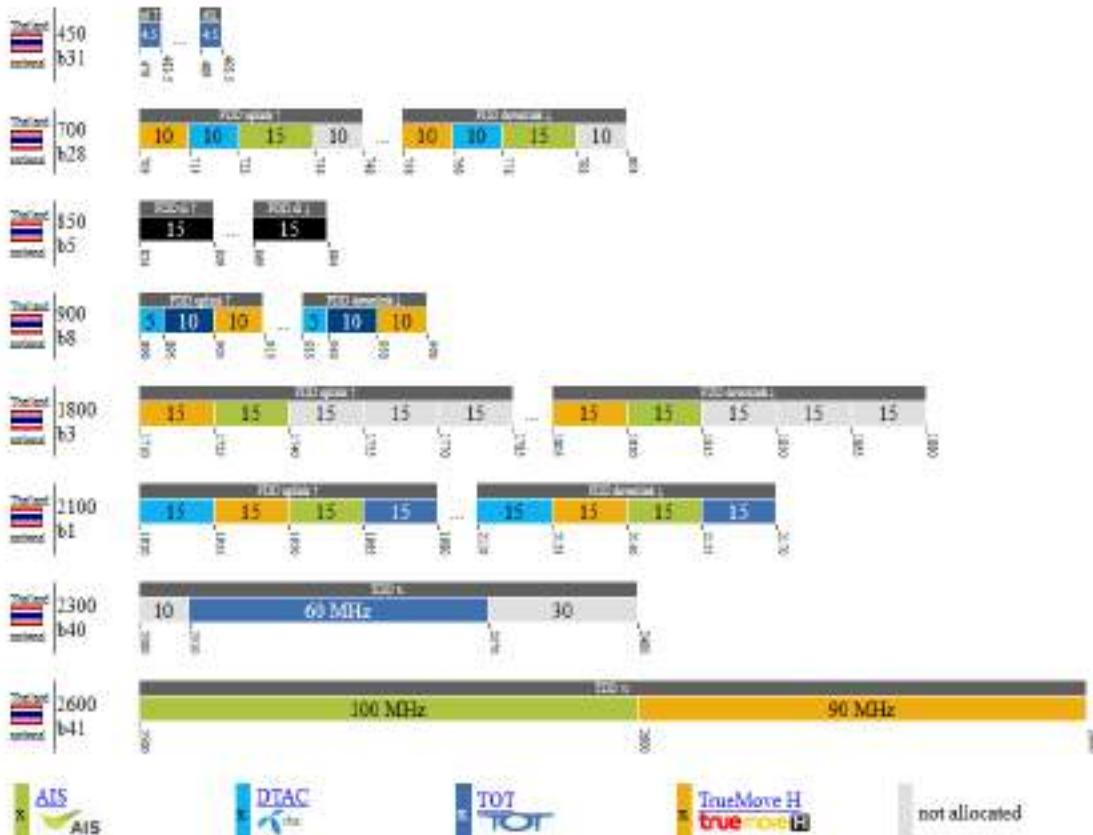
300 GHz - 1000 GHz (Not allocated)

<https://www.youtube.com/watch?v=VzD-L24vg4U>

# IoT Connectivity vs Access Networks



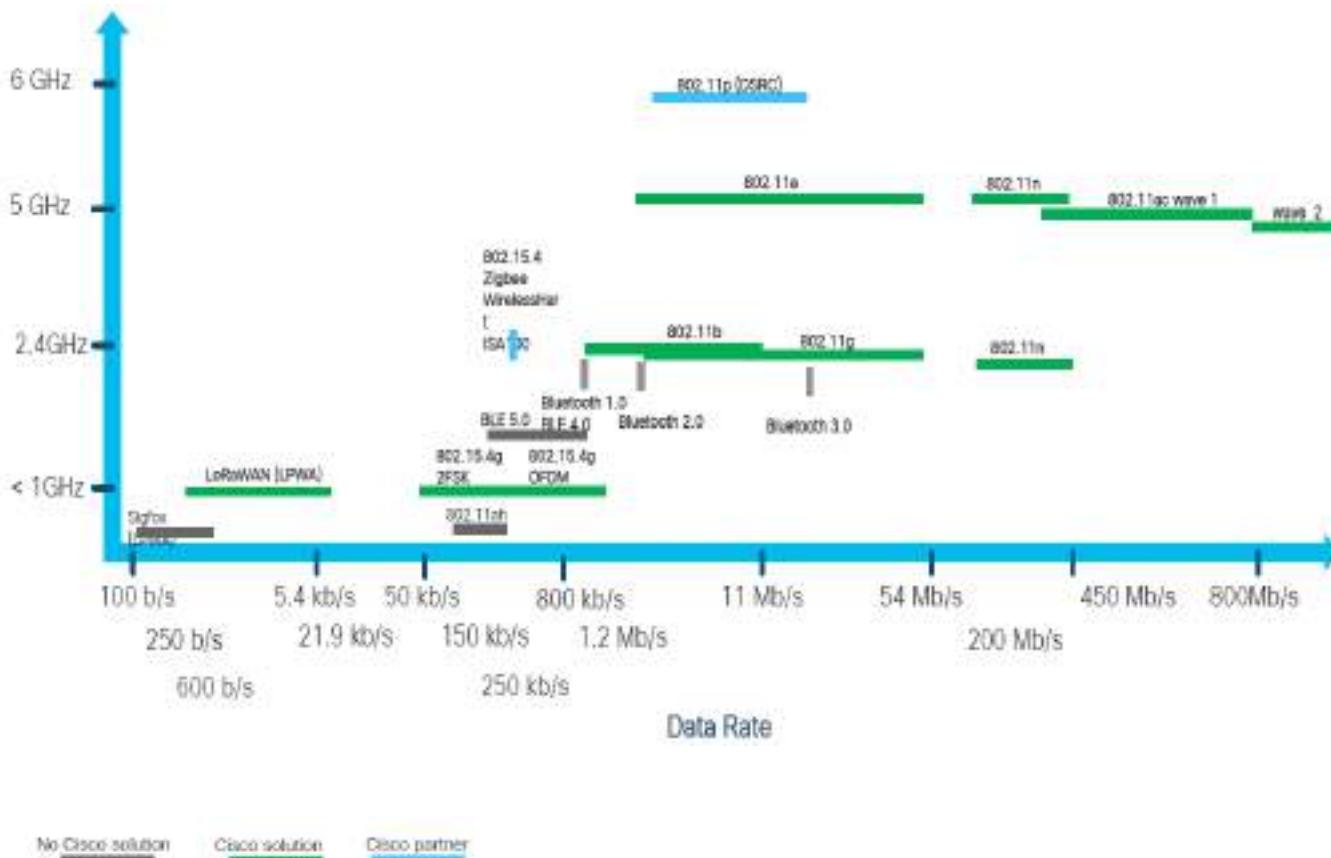
# Shared Licensed: Global Mobile Frequencies



<https://www.spectrummonitoring.com/frequencies.php>



## Unlicensed Bands: IoT Wireless Technologies e.g. CISCO



- In addition to spectral auctions, spectrum can be set aside in specific frequency bands that are **free to use** with a license according to a **specific set of etiquette rules**.
- The purpose of these unlicensed **bands** is to encourage **innovation and low-cost implementation**.
- Many extremely successful wireless systems operate in unlicensed bands, including **wireless LANs, Bluetooth, and cordless phones**.
- A major difficulty is that they can be killed by their own success.
  - If many unlicensed devices in the same band are used in close proximity, they generate much **interference to each other**, which can make the band unusable.

# IoT Connectivity vs Access Networks



## Licensed/Unlicensed Bands in IoT 1/2

| Technology | Standard                   | Distance | Data rate                      | Freq. Band                                                   | Licensed | Indoor/ outdoor | Mobility            | Topology                                | Latency |
|------------|----------------------------|----------|--------------------------------|--------------------------------------------------------------|----------|-----------------|---------------------|-----------------------------------------|---------|
| 3G         | IMT-2000                   | 35 km    | 384 kb/s                       | 1.25 MHz/<br>5 MHz                                           | Both     | Indoor /Outdoor | 90 km/h             | -                                       | 25ms    |
| 4G         | Single unified LTE + WiMax | 100 km   | 3Gb/s,<br>1.5Gb/s              | 1.8-2.6 GHz                                                  | Both     | Indoor /Outdoor | 110 km/h            | -                                       | 15ms    |
| LTE        | 3GPP Rel 8                 | 75 km    | 30 CMbs/s(DL),<br>75 Mb/s(UL)  | 2.5 GHz,<br>5 GHz, 10 GHz                                    | Both     | Indoor /Outdoor | 350km/h             | Star                                    | 15ms    |
| LTE-A      | 3GPP Release 10, 11, 12    | 200 km   | 1Gb(DL),<br>500Mb/s(U<br>L)    | 2.5 GHz,<br>5 GHz, 10 GHz,<br>15 GHz, 20 GHz                 | Both     | Indoor /Outdoor | 350km/h-<br>500km/h | Point to point                          | 10ms    |
| LTE-A Pro  | 3GPP Release 13 and beyond | 150km    | higher data rate beyond 3 Gbps | Licensed (400 MHz to 3.8 GHz) and unlicensed (5GHz) spectrum | Both     | Indoor /Outdoor | 500km/h             | End-to-End Network Slicing for Multiple | 2ms     |
| 5G         | Single unified, 4G + WW/WW | 300km    | 10-50 Gb/s                     | 1.8-2.6 GHz-<br>30-300 GHz, UP 60GHz                         | Both     | Indoor /Outdoor | 500 km/h            | End-to-End Network Slicing for Multiple | 1ms     |
| LoRa       | Proprietary                | 22 Km    | 290 bps-50 kbps                | 868 MHz and 900 ed MHz                                       | Unlicens | Indoor/O utdoor | —                   | Star                                    | —       |

| Technology                  | Standard | Distance  | Data rate                                 | Freq. Band                                                     | License  | Indoor/ outdoor | Mobility | Topology                 | Latency |
|-----------------------------|----------|-----------|-------------------------------------------|----------------------------------------------------------------|----------|-----------------|----------|--------------------------|---------|
| IEEE 802.16e (WiMax)        | 3GPP     | 50 km     | 15 Mbps<br>90 Mbps                        | 2.4 GHz and<br>2.4835 GHz or between<br>5.75 GHz and 5.850 GHz | Licensed | Indoor/ Outdoor | -        | Point-to-multipoint mesh | 30ms    |
| IEEE 802.16m (Mobile WiMax) | 3GPP     | 30-100 km | 120 Mbit/s up to 5 downlink/<br>60 Mbit/s | 100MHz                                                         | Both     | Indoor /Outdoor | 120km/h  | Point-to-multipoint mesh | 27.5    |

# IoT Connectivity vs Access Networks



## Licensed/Unlicensed Bands in IoT 2/2

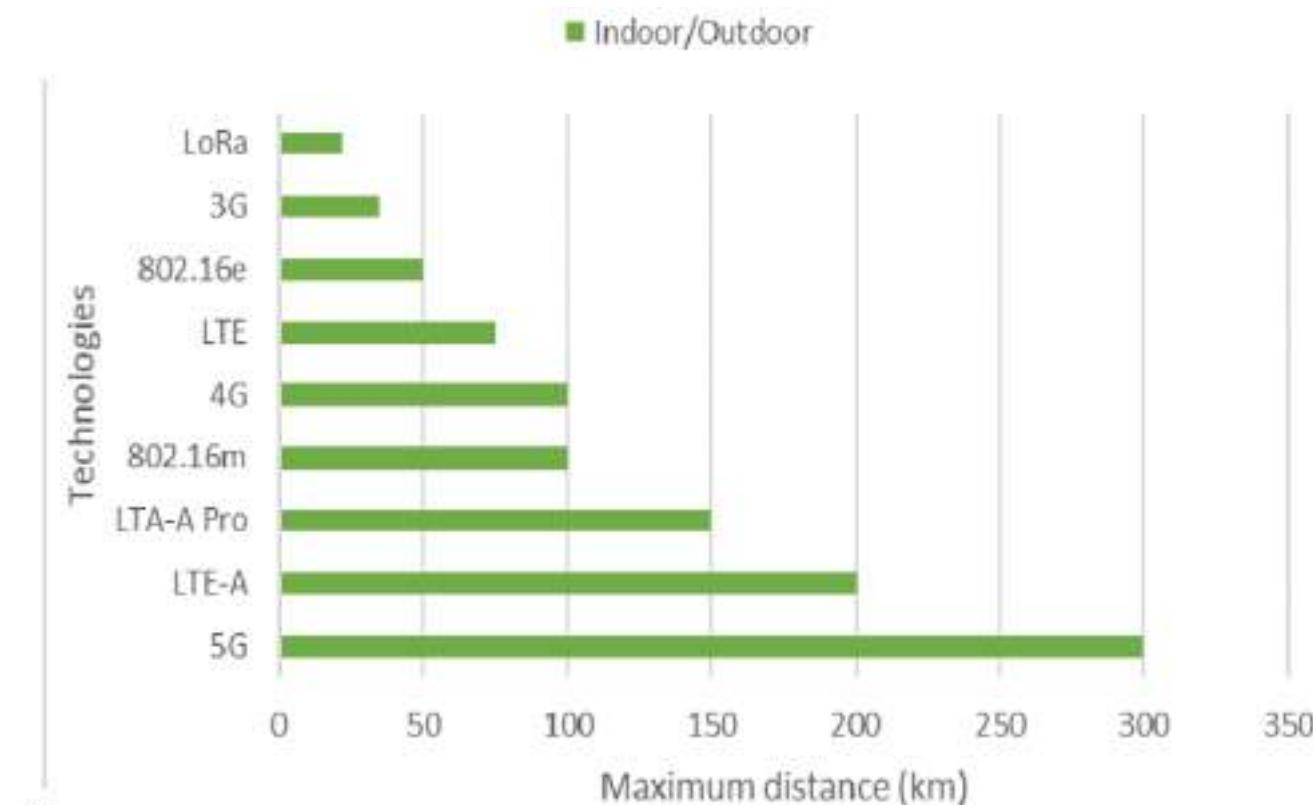
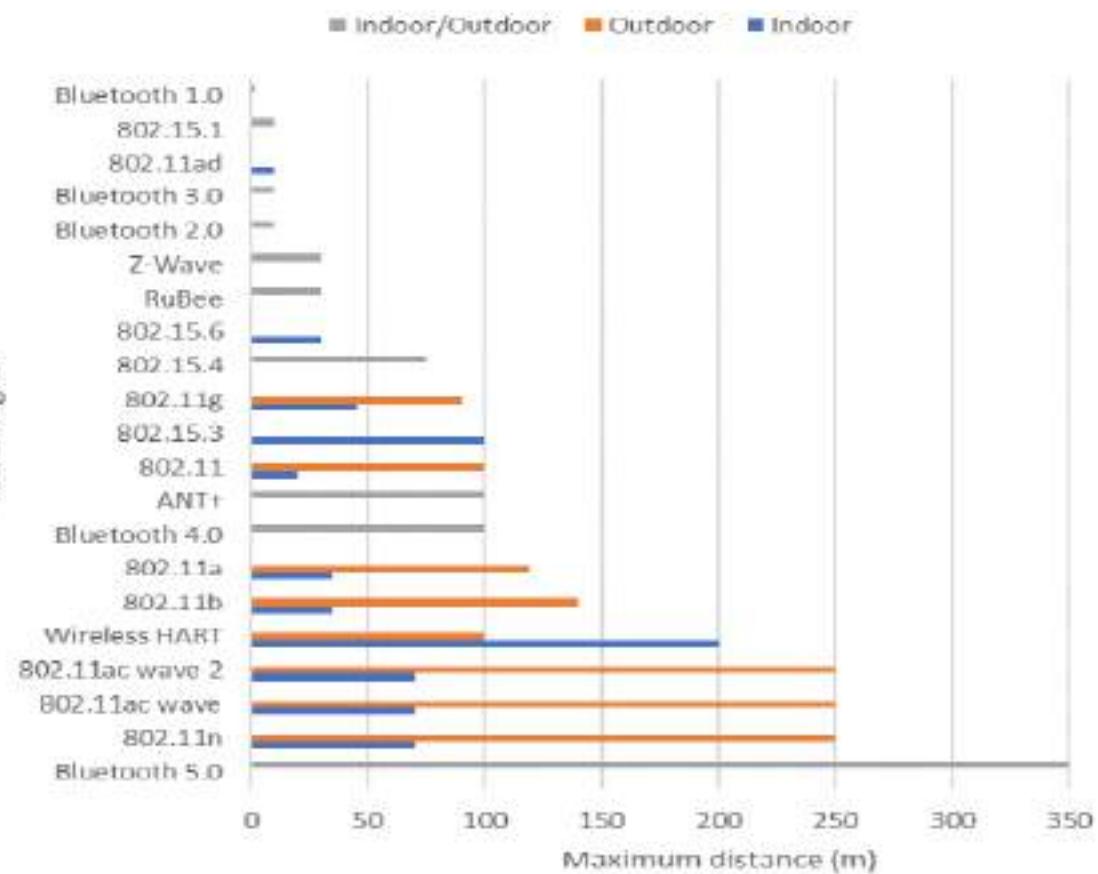
| Technologies           | Distance                                 | Bitrate                    | Freq. bands          | License              | Indoor/Outdoor  | Energy consumption | Capability              |
|------------------------|------------------------------------------|----------------------------|----------------------|----------------------|-----------------|--------------------|-------------------------|
| IEEE 802.15.1          | 10 m                                     | 1 Mbps                     | 2.4GHz               | unlicensed           | Indoor/Outdoor  | 0-10 dBm           | 2 active, 255 park mode |
| Bluetooth 1.0          | 2m                                       | 721 kbps                   | 2.4GHz               | unlicensed           | Indoor/Outdoor  | Not specified      | —                       |
| Bluetooth 1.1          | —                                        | 721 kbps                   | 2.4GHz               | unlicensed           | Indoor/Outdoor  | Not specified      | 2 active, 255 park mode |
| Bluetooth 2.0          | 10 m                                     | 2.1 Mbps                   | 2.4GHz               | unlicensed           | Indoor/Outdoor  | Not specified      | 2 active, 255 park mode |
| Bluetooth 3.0          | 10 m                                     | 24 Mbps                    | 2.4GHz               | unlicensed           | Indoor/Outdoor  | Not specified      | 2 active, 255 park mode |
| Bluetooth 4.0          | 100 m                                    | 1 Mbps                     | 2.4 GHz              | Unlicensed           | Indoor/Outdoor  | Not specified      | 2 active, 255 park mode |
| Bluetooth 5.0 [82]     | 300 m                                    | 1-2.5 Mbps                 | 2.4 GHz              | Unlicensed           | Indoor/Outdoor  | 1-100 mW           | 2 active, 255 park mode |
| IEEE 802.15.3          | 10-100m                                  | 110-400 Mbps               | 3.1-10.6 GHz         | Licensed             | Indoor          | -41.5 dBm/ -10 MHz | —                       |
| IEEE 802.15.4          | 10-75 m                                  | 20 kbps/ 40 kbps/ 150 kbps | 808-915 MHz, 2.4 GHz | Licensed/ Unlicensed | Indoor/ Outdoor | -25 dBm / -0 dBm   | 154                     |
| IEEE 802.15.6 [83]     | 10 m/ 30 m                               | 480 Mbps/ 50 Mbps          | 3.1-10.6 GHz         | Unlicensed           | Indoor          | —                  | —                       |
| ANT+ [83]              | 100 m/ 20m                               | 1 Mbps                     | 2.4 GHz              | Unlicensed           | Indoor/ Outdoor | —                  | 200                     |
| RaBe (IEEE 802.1) [83] | 30 m                                     | 1924 bps                   | 132 KHz              | Licensed             | Indoor/ Outdoor | —                  | —                       |
| Imotes [83]            | —                                        | 2100 bps                   | 900 MHz              | Unlicensed           | Indoor/ Outdoor | —                  | —                       |
| Z-Wave [83]            | 30 m                                     | 9.6 kbps/ 40 kbps          | 808-915 MHz, 2.4 GHz | Licensed/ Unlicensed | Indoor/ Outdoor | Low                | 232                     |
| WirelessHART [84]      | 75 m/ 100 m outdoor/ 100 m line-of-sight | 259 kbps                   | 2.4 GHz              | Unlicensed           | Indoor/ Outdoor | 10 dBm             | 160                     |

| Technologies         | Indoor/Outdoor | Bitrate   | Freq. bands    | License    | Bandwidth       | Modulation           | MIMO          |
|----------------------|----------------|-----------|----------------|------------|-----------------|----------------------|---------------|
| IEEE 802.11          | 20m/ 100m      | 2 Mbps    | 2.4GHz         | Unlicensed | 20 MHz          | FHSS and DSSS        | —             |
| IEEE 802.11b         | 35m/ 140m      | 11 Mbps   | 2.4GHz         | Unlicensed | 20 MHz          | HR-DSSS              | —             |
| IEEE 802.11a         | 35m/ 119m      | 54 Mbps   | 5GHz           | Unlicensed | 20 MHz          | OFDM                 | —             |
| IEEE 802.11g         | 45m/ 90m       | 54 Mbps   | 2.4 GHz        | Unlicensed | 22 MHz          | OFDM/ DSSS/ CCK      | —             |
| IEEE 802.11n         | 70m/ 250m      | 600 Mbps  | 2.4 GHz/ 5 GHz | Unlicensed | 20 MHz/ 40 MHz  | OFDM                 | 4 X 4         |
| IEEE 802.11ac wave   | 70m/ 250m      | 7000 Mbps | 5 GHz          | Unlicensed | 80 MHz          | 64-QAM               | MU-MIMO       |
| IEEE 802.11ad        | 10m/ n/a       | 7000 Mbps | 60 GHz         | Unlicensed | 2.16 GHz        | Single Carrier/ OFDM | 10 X 10       |
| IEEE 802.11ac wave 2 | 70m/ 250m      | 7000 Mbps | 5 GHz          | Unlicensed | 80 MHz/ 160 MHz | 256-QAM              | MU-MIMO 8 X 8 |

# IoT Connectivity vs Access Networks



## Indoor or Outdoor: Maximum Distance (m/km)

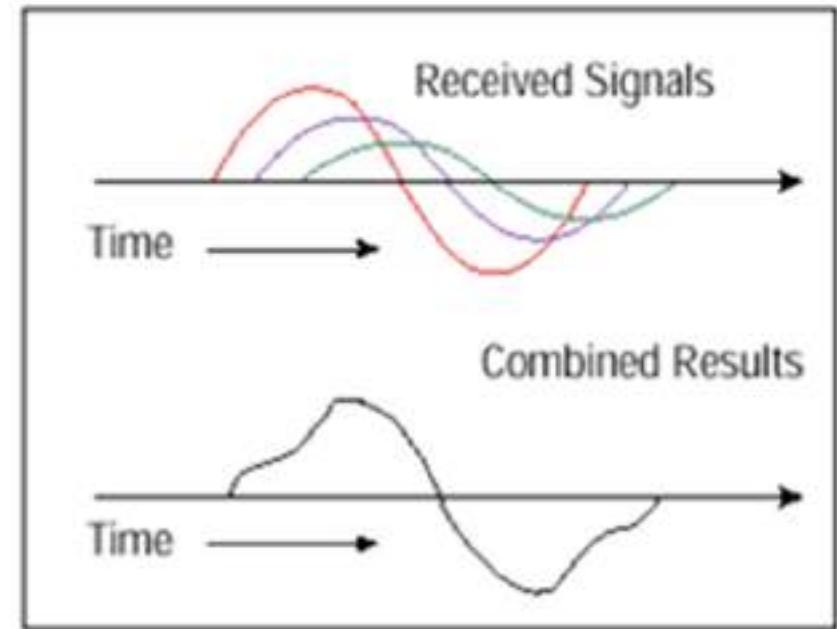
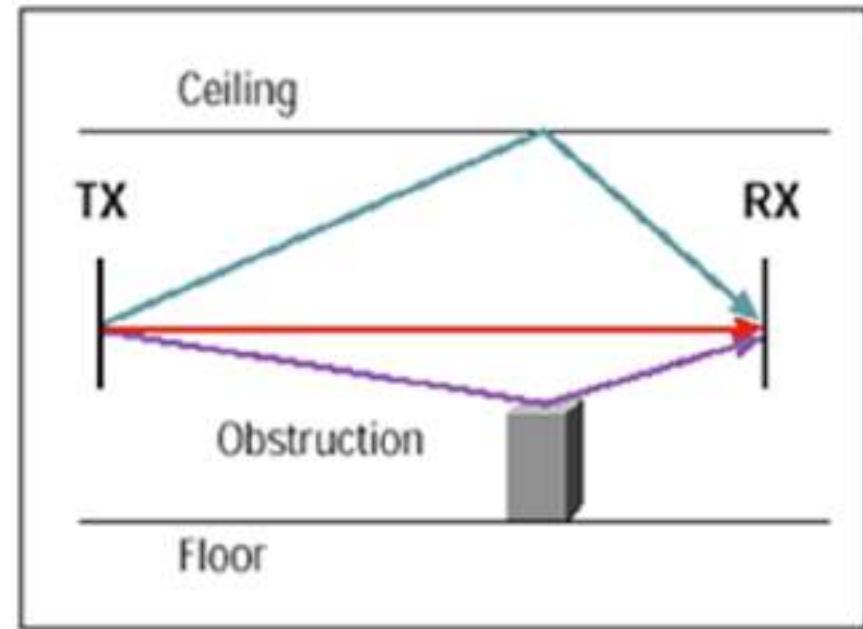




## Considerations for Wireless Technologies

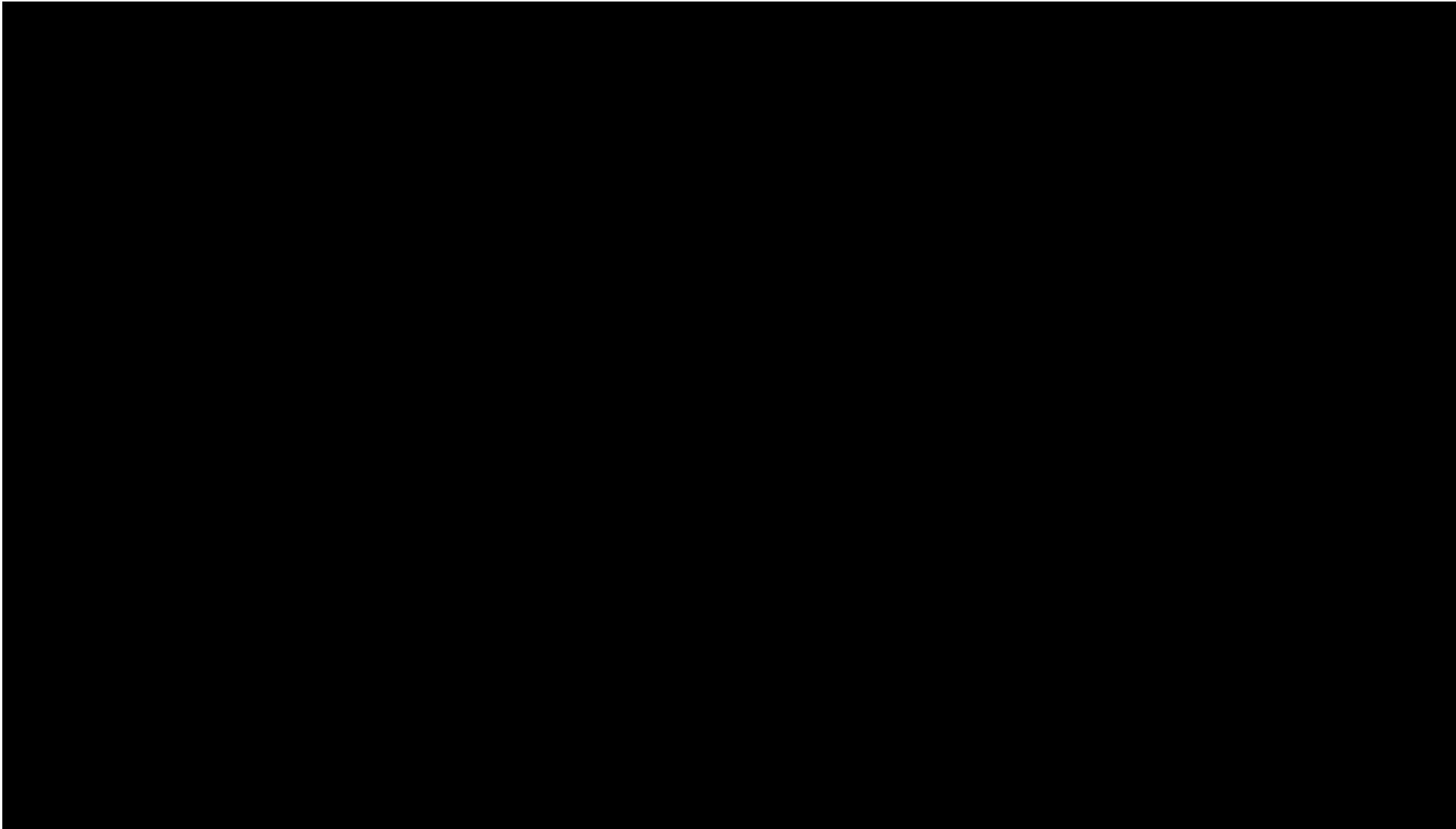
|                                 |                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Wired / Wireless communications | A wired network uses cables to connect devices to a network so that data can be transmitted between source and destination addresses. A wireless network uses radio waves to connect devices over the air without wires.                                                                                                                                         |
| Spectrum                        | Spectrum refers to the invisible radio frequencies that wireless signals travel over. Portions of electromagnetic spectrum are grouped in "bands" depending on their wavelengths—the distance over which the wave's shape repeats. Radio spectrum, we are talking about the range of radio frequencies that are used for communicating, from 30 Hertz to 300 GHz |
| Licensed / unlicensed spectrum  | <ul style="list-style-type: none"><li>• Licensed – assigned exclusively to operators for independent usage</li><li>• Unlicensed – assigned for non-exclusive usage (ie. Anyone can use) subject to some regulatory constraints, eg. restrictions in transmission power</li></ul>                                                                                 |
| Frequency band                  | A radio frequency band is a small contiguous section of the radio spectrum frequencies, in which channels are usually used or set aside for the same purpose. This is to prevent interference and allow for efficient use of the radio spectrum.                                                                                                                 |
| Data rate                       | A data rate is the theoretical maximum value that a wireless link can achieve if there were no losses or interference.                                                                                                                                                                                                                                           |
| Throughput                      | In the real world, there will be interference and losses which will result in a lower bit rate. Throughput can be seen as a practical/realistic value that a wireless link can achieve.                                                                                                                                                                          |
| Latency                         | Network latency is the term used to indicate any kind of delay that happens in data communication over a network. Eg. Queuing or processing times                                                                                                                                                                                                                |
| Mobility                        | Mobility, or roaming, is an ability of a wireless client to maintain its association seamlessly from one access point to another securely and with as little latency as possible.                                                                                                                                                                                |
| Battery/powered                 | Whether a sensor can be powered via battery or requires mains electrical power to operate.                                                                                                                                                                                                                                                                       |
| Quality of Service (QoS)        | Capabilities to manage the prioritization of traffic, and delay, jitter, bandwidth, and packet loss parameters on a network.                                                                                                                                                                                                                                     |
| Security                        | Network security is any hardware or software activity designed to protect the usability and integrity of communications network performance and the data traversing it.                                                                                                                                                                                          |

## Signal Interferences & Multipath Signaling Transmission: attenuation, destructive



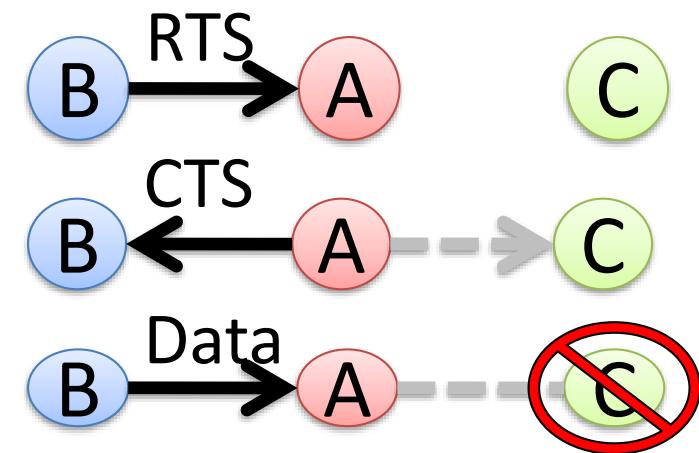


## CSMA/CA cs CSMA/CD



## RTS/CTS

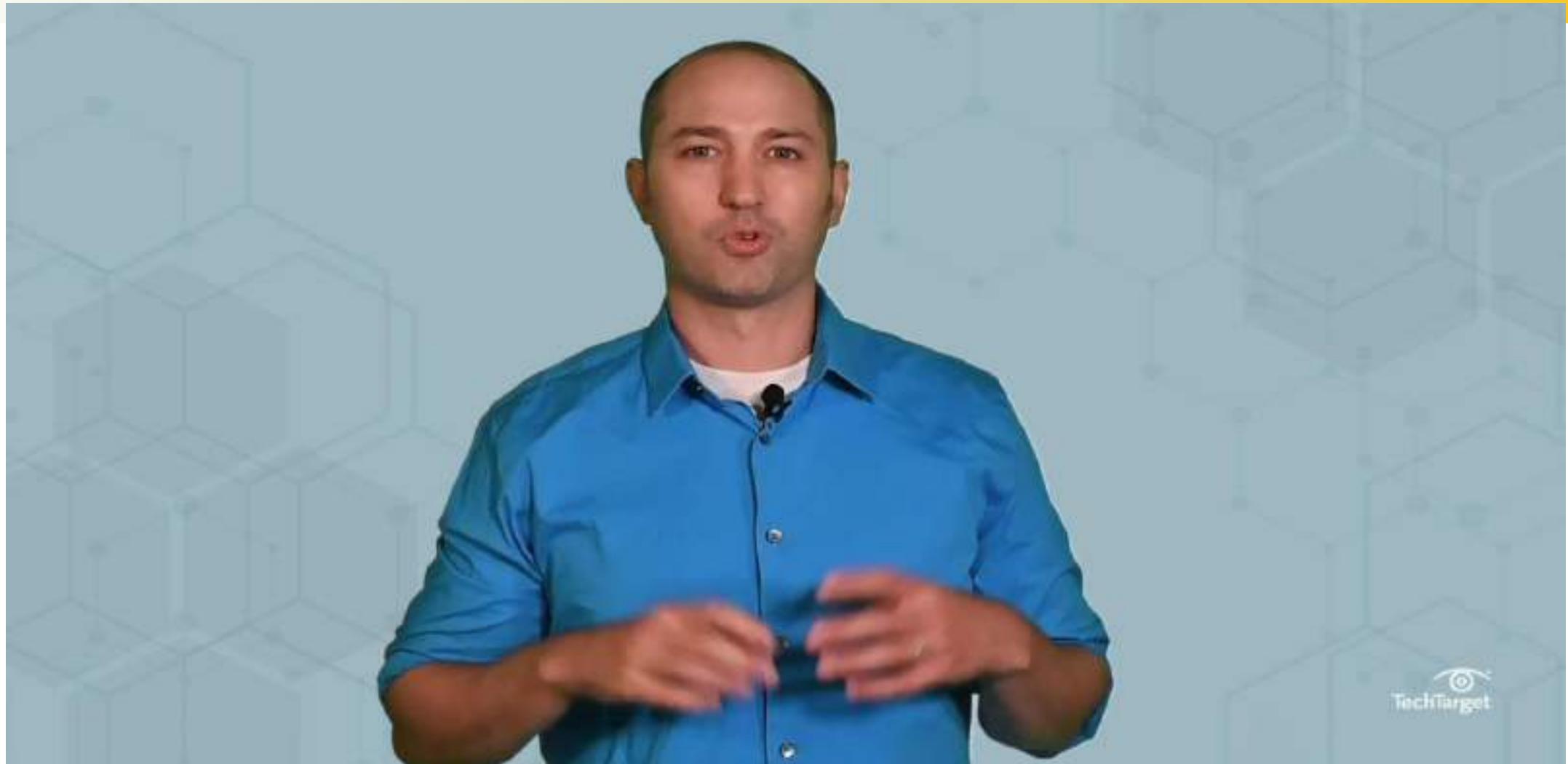
- **Idea: transmitter can check availability of channel at receiver**
- **Before every transmission**
  - Sender sends an RTS (Request-to-Send)
  - Contains length of data (in *time* units)
  - Receiver sends a CTS (Clear-to-Send)
  - Sender sends data
  - Receiver sends ACK after transmission
- **If you don't hear a CTS, assume collision**
- **If you hear a CTS for someone else, shut up**



# IoT Connectivity vs Access Networks



## OFDMA vs MIMO



[http://youtube.com/watch?v=jU\\_hacMiUAc](http://youtube.com/watch?v=jU_hacMiUAc)



## IEEE 802.11 (Wi-Fi): High Speed Ubiquitous IoT Wireless Technology

Wi-Fi: Wireless Fidelity (Trademark)

Wifi.org (Wi-Fi Alliance as a non-profit organization)





## IEEE 802.11: Evolution





## Wi-Fi vs Cell Phones

<http://youtube.com/watch?v=kxLcwIMYmr0>



## Wi-Fi 5 vs Wi-Fi 6



<http://youtube.com/watch?v=kxLcwIMYmr0>

# IoT Connectivity vs Access Networks



## Wi-Fi 6

The diagram highlights five key benefits of Wi-Fi 6:

- Increased data rates**  
*Higher modulation (Up to 1024 QAM)*
- Reduced latency**  
*Uplink resource scheduling (OFDMA)*
- Greater IoT coverage**  
*Deterministic capacity (OFDMA)*
- Higher density**  
*Efficient spectral re-use (OFDMA, BSS coloring)*
- Power efficient**  
*Flexible low-power scheduling (Target wake time)*

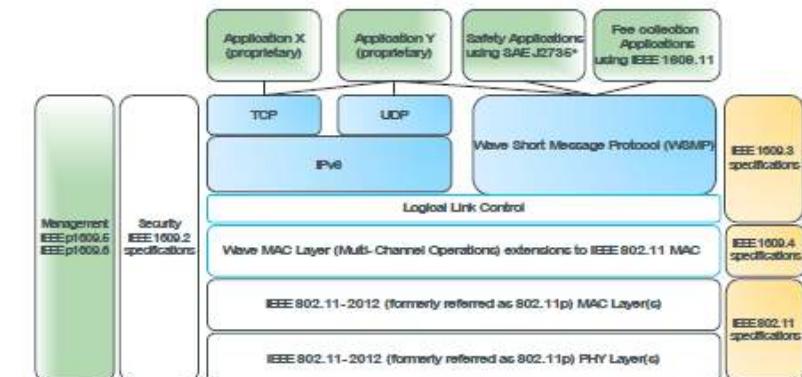
**Faster speeds | Optimized capacity | IoT ready**



## IEEE 802.11p: Dedicated Short Range Communications (DSRC)

### Connected roads use cases

- ▶ DSRC is built on IEEE 802.11p extensions, running in 6GHz band
  - Data rate: 6-27 Mbps with 10 MHz Channels
  - Range up to 1000m (3000ft)
  - Will evolve with IEEE 802.11bd
- ▶ Profile is defined for IPv6 (layer-3) and IEEE 1609 WSMP (layer-2)
- ▶ Payload: less than 100 bytes for V2V, larger (+400 bytes) for 12V
- ▶ **RSU (roadside unit)** – WAVE devices that operate only when stationary and support information exchange with OBUs
- ▶ **OBU (on-board unit)** – WAVE devices that can operate when in motion and support the information exchange with RSUs or other OBUs
- ▶ Partnering



| Region        | Unlicensed Frequency band          |
|---------------|------------------------------------|
| North-America | 5.850-5.925 GHz                    |
| Europe        | 5795-5815, 5855/5875-5905/5925 GHz |
| Japan         | 5770-5850 GHz                      |
| Singapore     | 5.855 GHz to 5.925 GHz             |
| India         | 5.725 to 5.825 GHz                 |
| Australia/NZ  | 5,725-5,795, 5,815-5,875 MHz,      |
| China         | 5,725-5,850 MHz                    |
| Korea         | 5,795-5,815 MHz                    |



# IoT Connectivity vs Access Networks



## Cellular vs IEEE 802.11 (1/2)

| Positioning                                       |                                                                                     | Cellular                                                                      | Wi-Fi                                                                                                     |                                                                               |                                                                                                  |                                                               |
|---------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|---------------------------------------------------------------|
| Features                                          |                                                                                     | 4G/LTE                                                                        | Shared License Band - CBRS Example                                                                        | 5G                                                                            | 802.11ac (Wi-Fi 5)                                                                               | 802.11ax (Wi-Fi 6)                                            |
| Fast, IP-based, mobile broadband                  | • Widespread/ubiquitous geographical coverage                                       | Citizen broadband radio service                                               | • Multi-technology (including WiFi) support capable                                                       | Enhanced mobile broadband, and wireless for industry                          | 5 <sup>th</sup> generation wireless                                                              | 6 <sup>th</sup> generation wireless                           |
| • Higher data speeds, cheaper device costs        | • Improved network responsiveness with lower latency and lower idle-to-active times | • Create own private LTE networks, replacing/supplementing Wi-Fi              | • Efficient use of wireless spectrum - less interference                                                  | • Higher data rates and improved quality of experience for wireless broadband | • Current generation of wireless with speed performance comparable to standard wired connections | • 40% higher data rates                                       |
| • Backwards compatibility                         | • Interoperability through all IP network                                           | • Ecosystem dependent                                                         | • Neutral host feature allows connectivity to multiple LTE providers to better support dense environments | • More device capacity for IoT and enterprise deployments                     | • Higher speeds over wider bandwidths than previous standards                                    | • More predictable performance in dense client environments   |
| • Enhancements to security and Quality of Service | • Mobility support                                                                  | • Potential benefits for multi-IoT application deployments                    | • Support for new high bandwidth and low latency services                                                 | • Support for new high bandwidth and low latency services                     | • More reliable long-range transmissions over previous standards                                 | • Simultaneous comms with multiple clients                    |
| • Country/SP dependent                            |                                                                                     | • US only, Netherlands, UK, Germany, Sweden, Japan planning something similar | • Improved performance with less spectrum interference                                                    | • Improved performance with less spectrum interference                        | • Improved performance in environments with obstructions                                         | • Real time applications, up to 75% less latency              |
|                                                   |                                                                                     |                                                                               | • Virtualization capabilities                                                                             | • Virtualization capabilities                                                 | • Backwards compatibility                                                                        | • Increased capacity + range                                  |
|                                                   |                                                                                     |                                                                               | • Connections in hard to reach areas and wired replacement                                                | • Connections in hard to reach areas and wired replacement                    | • Improved multi-user performance                                                                | • Improved power efficiency and battery of devices            |
|                                                   |                                                                                     |                                                                               | • Mobility support                                                                                        | • Mobility support                                                            | • High b/w backhaul links                                                                        | • More robust outdoor performance                             |
|                                                   |                                                                                     |                                                                               | • Country/SP dependent                                                                                    |                                                                               |                                                                                                  | • Easier to guarantee near wall-to-wall indoor coverage vs 5G |
|                                                   |                                                                                     |                                                                               | • High b/w backhaul links                                                                                 |                                                                               |                                                                                                  | • Backwards compatibility                                     |
|                                                   |                                                                                     |                                                                               |                                                                                                           |                                                                               |                                                                                                  | • High b/w backhaul links                                     |
| • Indoor/outdoor                                  | • Low latency and improved scale of connections vs Wi-Fi                            | • Indoor/outdoor                                                              | • Target 10Gb UL, 20Gb DL                                                                                 | • Indoor/outdoor                                                              | • Max data rate 9.6Gb                                                                            | • Medium range                                                |
| • Max data rate 300Mb-1Gb                         | • LTE services 1Gb indoors - 5-10x outdoor line-of-sight                            | • Target user 50Mb UL, 100Mb DL                                               | • Target user 50Mb UL, 100Mb DL                                                                           | • Max data rate 6.93Gb                                                        | • Multi-user access                                                                              | • More deterministic behaviour                                |
| • Av. user speed 15-50Mb                          | • mobility/roaming across access points                                             | • 1 million devices per km <sup>2</sup>                                       | • Network slicing                                                                                         | • Medium range                                                                | • Improved indoor + outdoor coverage                                                             | • Reduced latency up to 75%                                   |
| • 10,000 connections per Km <sup>2</sup>          | • More deterministic performance for real-time                                      | • Sub-1 millisecond latency                                                   | • Predictability and QoS                                                                                  | • Variable latency                                                            | • Split network capacity among groups of devices                                                 | • Symmetric U/D bandwidth                                     |
| • 30-50ms latency                                 | • Asymmetric U/D bandwidth                                                          | • Long range                                                                  | • Asymmetric U/D bandwidth                                                                                | • Symmetric U/D bandwidth                                                     |                                                                                                  | • Symmetric U/D bandwidth                                     |



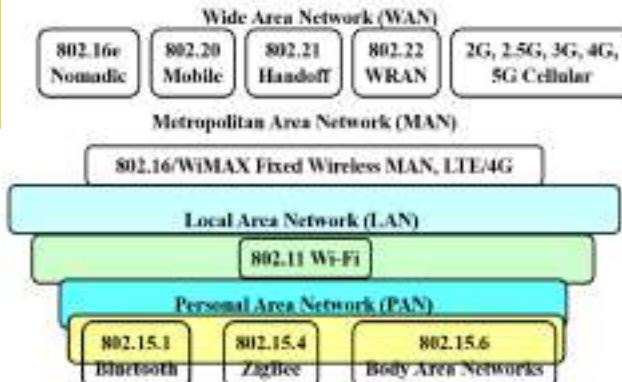
## Cellular vs IEEE 802.11 (2/2)

| Cellular                                                                                                                                                                                                                                  |                                                                                                                                                                                                                                                                              | Wi-Fi                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Technical attributes                                                                                                                                                                                                                      |                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| Example Use Cases                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>4G/LTE</b> <ul style="list-style-type: none"> <li>• Channel bandwidths 1.4, 3, 5, 10, 15, 20MHz</li> <li>• 300km/h mobility</li> <li>• Licensed and unlicensed</li> </ul>                                                              | <b>Shared License Band - CBRS Example</b> <ul style="list-style-type: none"> <li>• Spectrum: 3.55 to 3.7GHz</li> <li>• Licensed</li> <li>• Time Division Duplex (TDD)</li> <li>• Does not have to be line of sight</li> </ul>                                                | <b>5G</b> <ul style="list-style-type: none"> <li>• Spectrum: Low-band: 600MHz - 900MHz / Mid-band: 2.5G - 4.2GHz / Millimeter wave (high-band): 24GHz - 47GHz</li> <li>• Licensed</li> <li>• <b>Seamless mobility - up to 500km/h mobility</b></li> <li>• Network slicing</li> <li>• Dynamic traffic optimization</li> <li>• Dynamic policy control</li> <li>• Function/resource elasticity</li> <li>• Granular QoS/prioritization</li> <li>• Multiple-access connectivity</li> </ul>           |
| <ul style="list-style-type: none"> <li>• Smart metering</li> <li>• Distribution automation</li> <li>• Smart agriculture</li> <li>• Renewable/variable energy</li> <li>• Smart lighting</li> <li>• Mobile connectivity services</li> </ul> | <ul style="list-style-type: none"> <li>• Last mile and line of site connectivity</li> <li>• Mass IoT sensor deployments</li> <li>• Multi IoT application deployments</li> <li>• Private LTE networks</li> <li>• Private wireless</li> <li>• Neutral host networks</li> </ul> | <ul style="list-style-type: none"> <li>• High performance media applications</li> <li>• Smart vehicle-to-X transportation/autonomous</li> <li>• Dense outdoor sensor and IoT device connectivity</li> <li>• Last mile and line of site connectivity</li> <li>• Location based services</li> <li>• Virtual + augmented reality</li> <li>• Smart metering</li> <li>• Distribution automation</li> <li>• Renewable/variable energy</li> <li>• Smart agriculture</li> <li>• Edge compute</li> </ul> |
|                                                                                                                                                                                                                                           |                                                                                                                                                                                                                                                                              | <ul style="list-style-type: none"> <li>• Pervasive indoor and medium range outdoor connectivity</li> <li>• User Mobility</li> <li>• <b>Location based services</b></li> <li>• Sensor connectivity</li> <li>• Smart machines - monitoring</li> <li>• Virtual + augmented reality</li> <li>• Smart retail + customer experience</li> </ul>                                                                                                                                                        |
|                                                                                                                                                                                                                                           |                                                                                                                                                                                                                                                                              | <ul style="list-style-type: none"> <li>• Pervasive indoor and medium range outdoor connectivity</li> <li>• User Mobility</li> <li>• Location based services</li> <li>• Sensor connectivity</li> <li>• Smart machines - monitoring and control</li> <li>• Virtual + augmented reality</li> <li>• Smart retail + customer experience</li> <li>• Time sensitive applications</li> </ul>                                                                                                            |



## IEEE 802.15 Projects: Personal Area Network

- IEEE 802.15.1-2005: Bluetooth1.2
- IEEE 802.15.2-2003: Coexistence Recommended Practice
- IEEE 802.15.3-2016: High Rate (55 Mbps) Multimedia WPAN, includes 3c-2009 mm wave PHY, 3b-2005 High rate WPAN
- IEEE 802.15.3d-2017: 100 Gbps point-to-point PHY
- IEEE 802.15.3e-2017: High-Rate close proximity point-to-point MAC and PHY
- IEEE 802.15.3f-2017: High-rate wireless multi-media networks using mm waves
- IEEE 802.15.4a-2007: Precision Ranging
- IEEE 802.15.4c-2009: Chinese 314-316, 430-434, 779-787 MHz
- IEEE 802.15.4d-2009: Japanese 950 MHz
- IEEE 802.15.4e-2012: MAC Enhancements
- IEEE 802.15.4f-2012: PHY for Active RFID
- IEEE 802.15.4g-2012: PHY for Smart Utility Networks
- IEEE 802.15.4j-2013: Medical Body Area Network 2.36-2.4 GHz
- IEEE 802.15.4k-2013: Low Energy Critical Infrastructure Monitoring PHY
- IEEE 802.15.4m-2014: TV White Spaces PHY –between 56 MHz and 862 MHz
- IEEE 802.15.4n-2016: China Medical Band PHY
- IEEE 802.15.4p-2014: Rail (Train) Communications & Control PHY
- IEEE 802.15.4-2015: Low Rate (250kbps) WPAN –ZigBee
- IEEE 802.15.4md: Maintenance of IEEE 802.15.4-2015
- IEEE 802.15.4q-2016: Ultra Low Power PHY
- IEEE 802.15.4s-2018: System resource management capability
- IEEE 802.15.4t-2017: High rate (2 Mbps) PHY
- IEEE 802.15.4u-2016: 865-867 MHz band in India
- IEEE 802.15.4v-2017: Enabling use of regional sub-GHz bands (4n ,4q, 4t, 4u)
- IEEE P802.15.4w: Low-Rate Low-Power Wide Area Network (LPWAN) extension to 802.15.4 PHY to cover 10-15 km
- IEEE P802.15.4x: Field Area Network extensions for devices with no battery or very limited battery consumption (Smart Utility Network)
- IEEE P802.15.4y: Security next generation using AES-256
- IEEE P802.15.4z: Enhanced impulse radio Ultra-Wide Band(UWB)
- IEEE 802.15.5-2009: Mesh Networking. Full/partial meshes. Range Extension
- IEEE 802.15.6-2012: Body Area Networking. Medical and entertainment. Low power
- IEEE 802.15.7-2011: Short Range Optical Wireless
- IEEE P802.15.7r1: Optical wireless (infrared, ultraviolet, visible light)
- IEEE P802.15.7m: Maintenance of 802.15.7-2011
- IEEE 802.15.8-2017: Peer Aware Communications
- IEEE 802.15.9-2016: Key Management Support
- IEEE 802.15.10-2017: Routing packets in dynamically changing wireless networks
- IEEE P802.15.10a: Routing mode additions. Automated discovery of nodes and route configuration
- IEEE P802.15.12: Upper Layer Interface (ULI) to harmonize fragmentation, configuration etc. for all 802.15.4 (Upper L2 and interface to L3)
- IEEE P802.15.13: Multi-Gigabit/s Optical Wireless with ranges up to 200m
- IEEE 802.15 IG6t: Consolidate Link Layer Control interest group
- IEEE 802.15 IGdep: Enhanced Dependability interest group
- IEEE 802.15 IGvat: Vehicular Assistive Technology
- IEEE P802.15.7m: Maintenance of 802.15.7-2011
- IEEE 802.15.8-2017: Peer Aware Communications
- IEEE 802.15.9-2016: Key Management Support
- IEEE 802.15.10-2017: Routing packets in dynamically changing wireless networks
- IEEE P802.15.10a: Routing mode additions. Automated discovery of nodes and route configuration
- IEEE P802.15.12: Upper Layer Interface (ULI) to harmonize fragmentation, configuration etc. for all 802.15.4 (Upper L2 and interface to L3)
- IEEE P802.15.13: Multi-Gigabit/s Optical Wireless with ranges up to 200m
- IEEE 802.15 IG6t: Consolidate Link Layer Control interest group
- IEEE 802.15 IGdep: Enhanced Dependability interest group
- IEEE 802.15 IGvat: Vehicular Assistive Technology
- IEEE 802.15 IGguide: Guide for 15.4 use interest group
- IEEE 802.15 IGrrc: High Rate Rail Communications interest group
- IEEE 802.15 IGTHz: Terahertz interest group
- IEEE 802.15 SCwng: Wireless Next-Generation standing committee
- IEEE 802.15 SCmaint: Maintenance standing committee
- IEEE 802.15 SCietf: IETF Liaison





## IEEE 802.15.1: Bluetooth





## IEEE 802.15.4: Design Challenges

- **Battery powered:**
  - Maximize battery life.  
A few hours to a few years on a coin cell.
- **Dynamic topologies:**
  - Short duration connections and then device is turned off or goes to sleep
- **No infrastructure**
- **Avoid Interference:**
  - due to larger powered LAN devices
- **Simple and Extreme Interoperability:**
  - Billions of devices. More variety than LAN or MAN
- **Low-cost:**
  - A few dollars

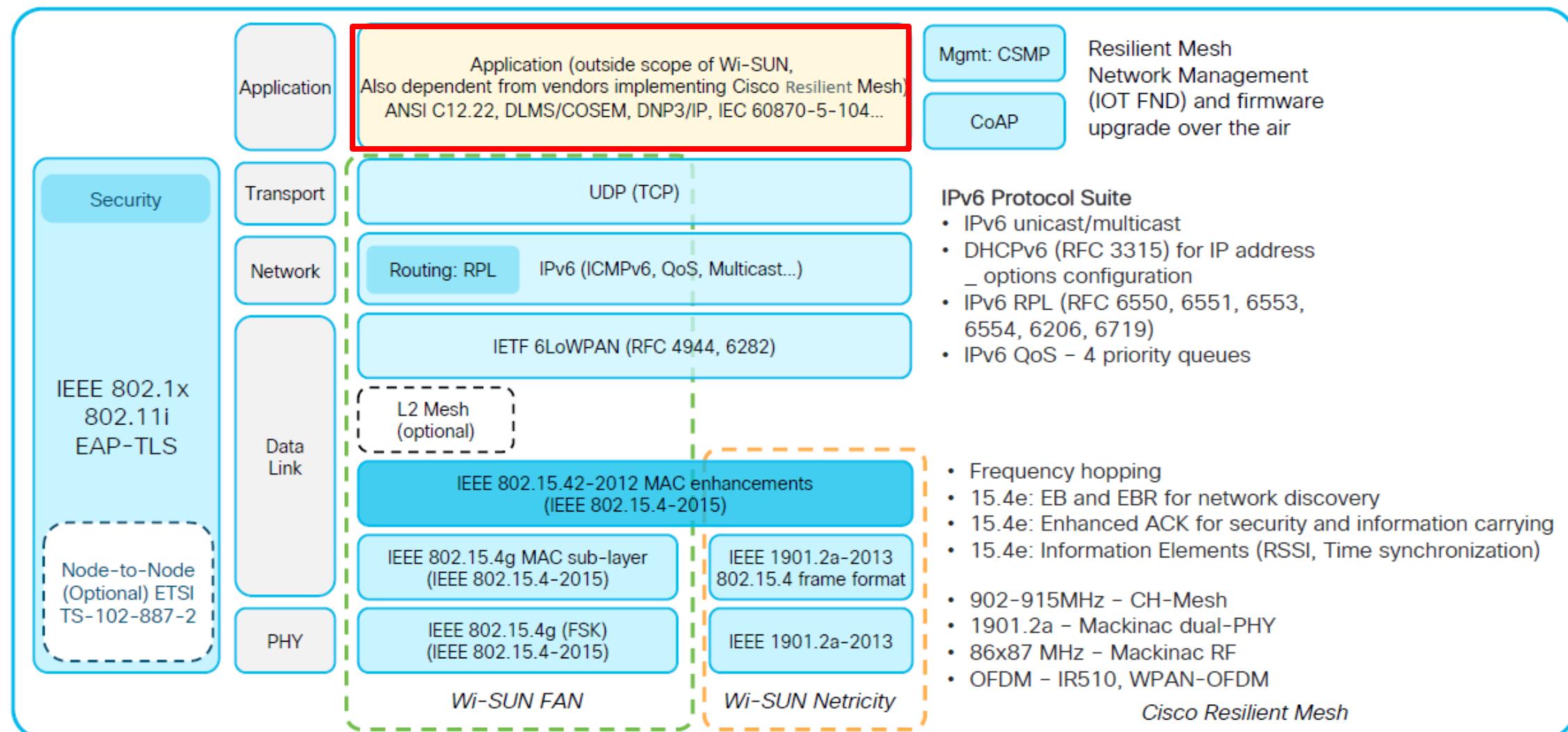


## IEEE 802.15.4: Personal Area Network

|                            | IEEE 802.15.4-2006                                                          | IEEE 802.15.4g/e<br>SUN PHYs                                                                         | Comments                                                                                                                                                                                                                      |
|----------------------------|-----------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Frequency bands            | 868 MHz 1-3 channels,<br>902-928 MHz 10-30 channels<br>2450 MHz 16 channels | In addition of 802.15.4-2011<br>frequency bands:<br>169, 450-470, 470-510, 863-870,<br>1427-1518 MHz | <ul style="list-style-type: none"> <li>Frequency bands availability per region/country</li> <li># channels: 802.15.4-2003 – 802.15.4-2006</li> <li>802.15.4-2011: 314-316, 430-434 and 779-787 MHz bands for China</li> </ul> |
| Modulation                 | BPSK, ASK (Sub-GHz)<br>O-QPSK (2.4GHz)                                      | MR-FSK, MR-OFDM and MR-O-QPSK                                                                        | BPSK/O-QPSK in 802.15.4-2003<br>802.15.4-2011 adds modulations<br>802.15.4g add 3 new PHY SUN modulation                                                                                                                      |
| Max. theoretical Data Rate | Up to 20, 40 and 250 kb/s                                                   | Up to 1200 kb/s (OFDM)                                                                               | Frequency band and modulation dependent                                                                                                                                                                                       |
| Maximum PSDU size          | 127 bytes                                                                   | 2047 bytes                                                                                           | Better aligned with IPv6 MTU (1280 bytes)                                                                                                                                                                                     |
| FCS                        | 16 bits                                                                     | 32 bits                                                                                              | Better error protection                                                                                                                                                                                                       |
| Information Elements       | No                                                                          | Yes, 15.4e                                                                                           | Allow vendor specific information                                                                                                                                                                                             |
| PAN ID                     | 0-65534                                                                     | 0-65534                                                                                              | Identifies a WPAN                                                                                                                                                                                                             |
| MAC Address                | 16 bits or 64 bits                                                          | 16 bits or 64 bits                                                                                   | 16 bits = locally managed, 64 bits = EUI-64                                                                                                                                                                                   |
| Usage                      | Zigbee, WirelessHart, ISA100                                                | Wi-SUN                                                                                               |                                                                                                                                                                                                                               |



## Wi-SUN (Smart Ubiquitous Networks)





## WirelessHart vs ISA 100.11a

|                 | WirelessHart                                                                                                | ISA100.11a                                                                                                                                            |
|-----------------|-------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| Frequency bands | IEEE 802.15.4-2006<br>2.4GHz, 16 channels                                                                   | IEEE 802.15.4-2006<br>2.4GHz, 16 channels                                                                                                             |
| Data Rate       | 250kbs                                                                                                      | 250kbs                                                                                                                                                |
| Standard        | IEC 62591                                                                                                   | IEC 62734                                                                                                                                             |
| Topology        | TDMA/CSMA based wireless mesh                                                                               | TDMA/CSMA star, mesh, star-mesh topologies                                                                                                            |
| Channel hopping | fixed channel hopping table<br>10 msec time slot                                                            | multiple channel hopping tables<br>variable slot time, default 10 msec                                                                                |
|                 | Based on HART addressing                                                                                    | 6LoWPAN, IPv6 and UDP                                                                                                                                 |
| Vendors         | Emerson, ABB, Siemens,<br>Endress+Hauser                                                                    | Honeywell, Yokogawa, GE                                                                                                                               |
| Specifications  | <a href="https://fieldcommgroup.org/hart-specifications">https://fieldcommgroup.org/hart-specifications</a> | <a href="https://www.isa.org/store/products/product-detail/?productId=118261">https://www.isa.org/store/products/product-detail/?productId=118261</a> |



# IoT Connectivity vs Access Networks

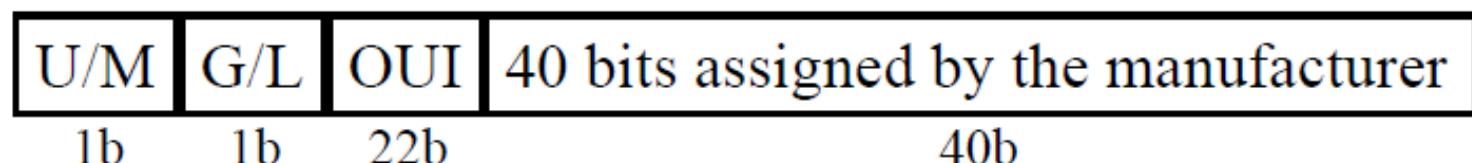


## Wi-SUN



## Overview

- Low Rate Wireless Personal Area Network (LR-WPAN)
- 2.4 GHz (most common). 16 x 5-MHz channels
- 250 kbps PHY  $\Rightarrow$  50 kbps application data rate
- Peak current depends upon symbol rate  $\Rightarrow$  multilevel 4b/symbol)
- Similar to 802.11: Direct Sequence Spread Spectrum, CSMA/CA, Backoff, Beacon, Coordinator (similar to Access point)
- Lower rate, short distance  $\Rightarrow$  Lower power  $\Rightarrow$  Low energy
- Each node has a 64-bit Extended Unique ID (EUI-64):



- No segmentation/reassembly.  
Max MAC frame size is 127 bytes with a payload of 77+ bytes.



## Topologies

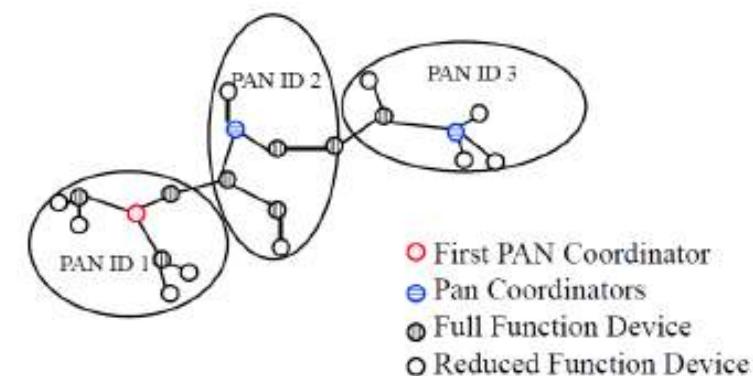
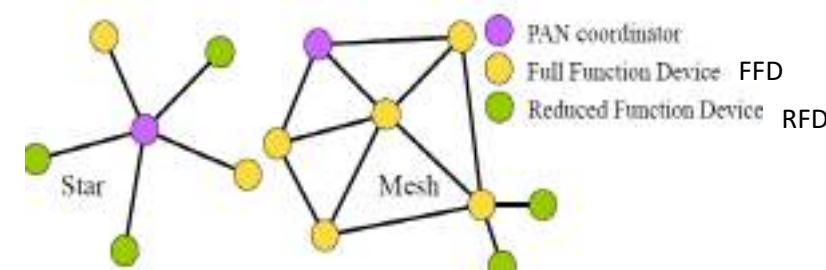
- FFDs can become coordinator and can also route messages to other nodes
- RFDs cannot become coordinator and can only be a leaf
- FFD that starts a PAN becomes the coordinator
- In star topology, all communication is to/from the coordinator
- In P2P topology, FFDs can communicate directly also.
- Each piconet has a PAN ID and is called a cluster.
- Nodes join a cluster by sending association request to the coordinator.

Coordinator assigns a 16-bit short address to the device.

Devices can use either the short address or EUI-64 address.

- A coordinator can ask another FFD to become a coordinator for a subset of nodes.

Tree  $\Rightarrow$  No loops



- First PAN Coordinator
- Pan Coordinators
- Full Function Device
- Reduced Function Device

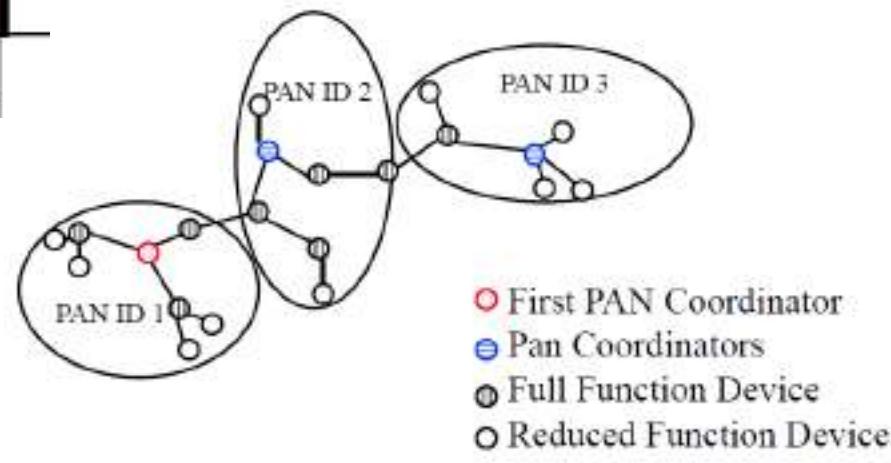
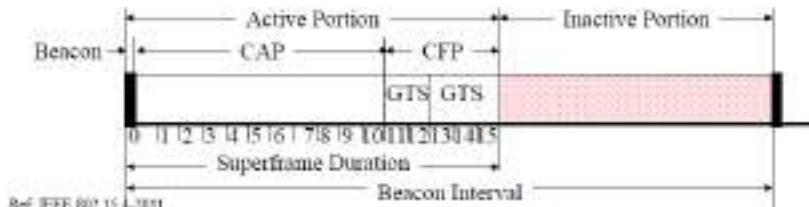
# IEEE 802.15.4: Personal Area Network



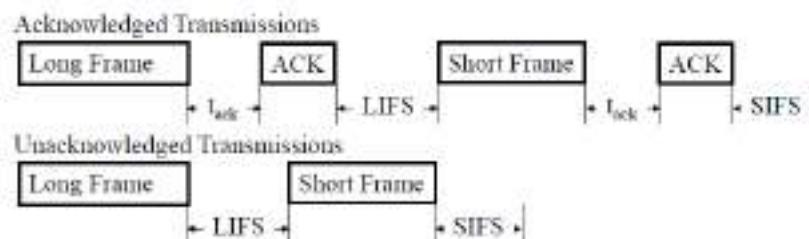
## MAC

### Beacon-Enabled CSMA/CA

- Coordinator sends out beacons periodically
- Part of the beacon interval is inactive  $\Rightarrow$  Everyone sleeps
- Active interval consists of 16 slots
- Guaranteed Timed Slots (GTS):  
For real-time services. Periodic reserved slots.
- Contention Access Period (CAP).** Slotted CSMA.



- Beaconless Operation:** Unslotted CSMA  
If coordinator does not send beacons, there are no slots
- Acknowledgements if requested by the sender.**
- Short inter-frame spacing (SIFS) if previous transmission is shorter than a specified duration.  
Otherwise, Long inter-frame spacing (LIFS)



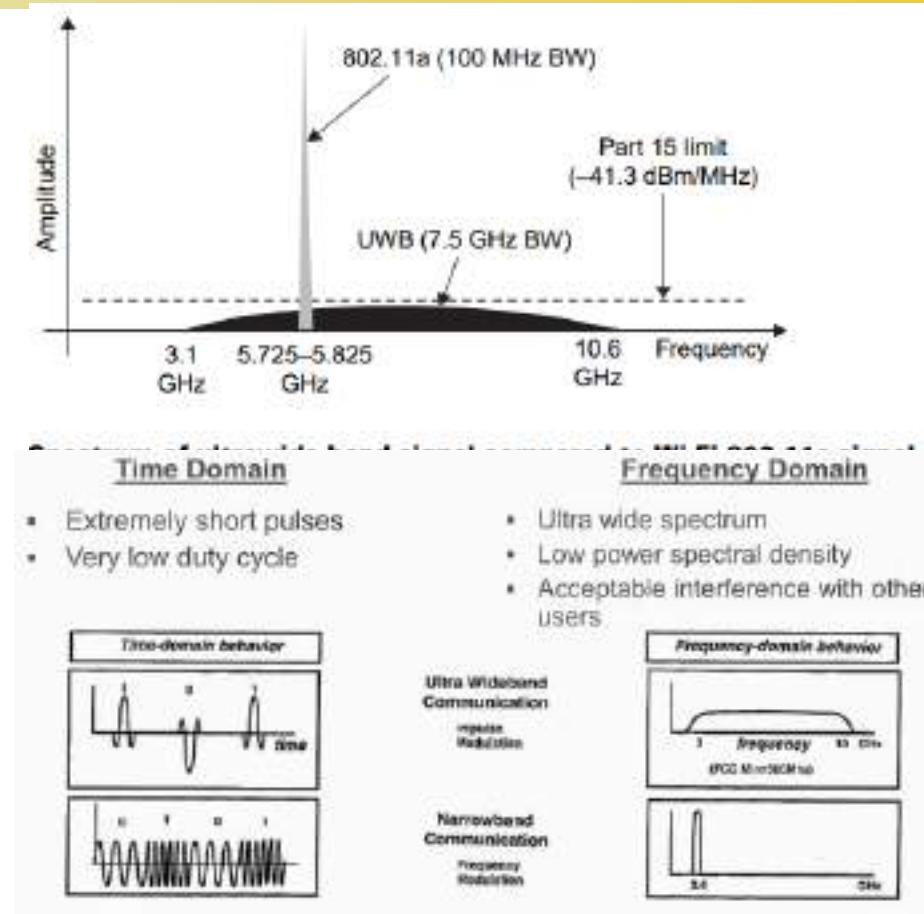
### 802.15.4 CSMA/CA

- Wait until the channel is free.
- Wait a random back-off period If the channel is still free, transmit.
- If the channel is busy, backoff again.
- Backoff exponent limited to 0-2 in battery life-extension mode.
- Acknowledgement and Beacons are sent without CSMA-CA.



## Ultra-Wideband (UWB) 1/2

- An impulse in time domain results in a ultra wide spectrum in frequency domain and essentially looks like a white noise to other devices.
- FCC rules restrict the maximum noise generated by a wireless equipment ( $0 \text{ dBm} = 1 \text{ mW}$ ,  $-40 \text{ dBm} = 0.1 \mu\text{W}$ )
- It is possible to generate very short (sub-nano sec) pulses that have spectrum below the allowed noise level  
⇒ Possible to get Gbps using 10 GHz spectrum
- FCC approved UWB operation in 2002
- UWB can be used for high-speed over short distances
- UWB can see through trees and underground (radar)  
⇒ collision avoidance sensors, through-wall motion detection
- Position tracking: cm accuracies. Track high-value assets
- Sub-nanosecond impulses are sent many million times per second
- Became feasible with high-speed switching semiconductor devices
- Pulse width = 25 to 400 ps
- Impulses may be position, amplitude, or polarity modulated
- 0.25 ns Impulse ⇒ 4 B pulses/sec ⇒ 100's Mbps
- 802.15.4 uses pulse position and binary phase shift keying modulation





## Ultra-Wideband (UWB) 2/2

### Advantages of UWB

- Very low energy consumption: Good Watts/Mbps
- Line of sight not required. Passes through walls.
- Sub-centimeter resolution allows precise motion detection
- Pulse width much smaller than path delay  $\Rightarrow$  Easy to resolve multipath  $\Rightarrow$  Can use multipath to advantage
- Difficult to intercept (interfere)
- All digital logic  $\Rightarrow$  Low cost chips
- Small size: 4.5 mm<sup>2</sup> in 90 nm process for high data rate designs

### Direct Sequence (DS-UWB)

- Championed by Motorola/XtremeSpectrum
- Uses CDMA with multiple chips per bit
- Chips are encoded using pulse
- This is the scheme used in 802.15.4
- Low power density  $\Rightarrow$  Good for body area network

### IEEE 802.15.4e Enhancements

- Low latency deterministic operation: pre-assigned slots
- Channel adaptation: Different channels used by different nodes for contention free period
- Time slotted channel hopping: Higher layers coordinate the slot allocation along with its frequency. Good for harsh industrial environments.
- Each device can select its listening channel
- Transmitter and receiver coordinate their cycles (very low duty cycle)
- Transmit only when requested by receiver

### Summary

- IoT fueled initially by smart grid is resulting in several competing protocols: **Bluetooth Smart, ZigBee Smart, ...**
- IEEE 802.15.4 is a low-data rate wireless personal area network and is the PHY and MAC layer used by many IoT protocols, such as ZigBee, and WirelessHART.
- 802.15.4 uses full function and reduced function devices. FFDs can act as coordinator. Allows a star, mesh, or a cluster tree topology.
- Uses Slotted/Unslotted CSMA/CA. Supports Guaranteed timed slots for low-latency application.
- UWB allows transmission with very low average power spread over a large band.

# NB-IoT vs LoRaWAN vs IEEE 802.15.4 (1/2)



## Positioning

IoT Focused

| NB-IoT                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | LoRaWAN                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | 802.15.4                                                                                                                                                                                                                                                                                                                                                                                                                        | LTE-M                                                                                                                                                                                                                                                                                                                                                                                                     |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>Narrowband IoT</b></p> <ul style="list-style-type: none"><li>• High density of devices</li><li>• Extremely small power budgets of devices</li><li>• Very long range</li><li>• May be able to provide deeper building and underground penetration than similar technology</li><li>• Challenges with firmware-over-the-air (FOTA) or large file transfers</li><li>• <b>Can co-exist with 2G, 3G, and 4G mobile networks.</b></li><li>• Same security benefits as mobile networks</li><li>• No mobile use cases</li><li>• Higher power consumption than LoRaWAN</li></ul> | <p><b>Long Range (LoRa)</b></p> <ul style="list-style-type: none"><li>• Designed for sensors and applications that need to send small amounts of data over long distances a few times per hour from varying environments</li><li>• Single GW covers wide area</li><li>• Multi-year battery lifetime/low power consumption</li><li>• Communications robustness</li><li>• 2 layers of security</li><li>• Uses device classes to optimize a variety of end application profiles</li><li>• Interoperability between manufacturers</li></ul> | <p><b>Wireless Sensor Networks</b></p> <ul style="list-style-type: none"><li>• aimed at providing the essential lower network layers for a wireless personal area network, WPAN</li><li>• low-cost, low-speed ubiquitous communication between devices</li><li>• <b>Provides foundation for a variety of different higher layer standards such as Zigbee, Wireless HART, ISA100, 6LowPAN (table in notes section)</b></li></ul> | <p><b>Long-Term Evolution for Machines</b></p> <ul style="list-style-type: none"><li>• Compatible with existing LTE network</li><li>• supports real-time device communication</li><li>• Device cost and power consumption reduction</li><li>• Extended battery life through sleep and power saving modes</li><li>• Lower service cost vs LTE due to reduced data b/w</li><li>• Mobile use cases</li></ul> |
| <ul style="list-style-type: none"><li>• 20-60kb/s</li><li>• Long range</li><li>• Less latency and higher throughput than LoRa</li><li>• Network security mechanisms</li><li>• Asymmetric U/D bandwidth</li></ul>                                                                                                                                                                                                                                                                                                                                                             | <ul style="list-style-type: none"><li>• Unlicensed band</li><li>• Indoor/outdoor</li><li>• 250 bps to 50 kbps</li><li>• Long range</li><li>• <b>Security mechanisms</b></li><li>• Symmetric/Asymmetric U/D bandwidth dependent on RF region</li><li>• Multicast support</li><li>• Firmware Over the Air FOTA</li></ul>                                                                                                                                                                                                                  | <ul style="list-style-type: none"><li>• Unlicensed band, i.e. EU868, US915</li><li>• Private infrastructure</li><li>• 50kbps – 1.2Mbps (US915)</li><li>• 50kbps – 300kbps (EU868)</li><li>• Medium range</li><li>• Simple or no QoS requirements</li><li>• Security mechanisms</li></ul>                                                                                                                                        | <ul style="list-style-type: none"><li>• 1Mb up / 384kb down</li><li>• High Latency Communication</li><li>• Indoor/outdoor</li><li>• Extended range</li><li>• <b>Asymmetric U/D bandwidth</b></li></ul>                                                                                                                                                                                                    |



# NB-IoT vs LoRaWAN vs IEEE 802.15.4 (2/2)

| IoT Focused          |                                                                                                                                                                                                                                                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                                                                                                                                                                                                                                                                                                                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Technical attributes | NB-IoT                                                                                                                                                                                                                                                             | LoRaWAN                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 802.15.4                                                                                                                                                                                                                                                                                                                                                                                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Use Cases            | NB-IoT                                                                                                                                                                                                                                                             | LoRaWAN                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 802.15.4                                                                                                                                                                                                                                                                                                                                                                                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|                      | <ul style="list-style-type: none"><li>• 200-kHz GSM spectrum</li><li>• In-band and Guard-band LTE, standalone GSM</li><li>• Half-duplex</li></ul>                                                                                                                  | <ul style="list-style-type: none"><li>• Spectrum variable by region eg. 868 - 868.6 Europe, 902-928 USA</li><li>• Frequency hopping</li><li>• Tunable variable data rates</li><li>• Network Session Key shared between end-device and network server</li><li>• Application Session Key (AppSKey) shared end-to-end at the application level</li><li>• AES algorithms are used to provide authentication and integrity of packets to the network server and end-to-end encryption to the application server.</li></ul> | <ul style="list-style-type: none"><li>• Spectrum variable by region, eg. 868 – 868.6 Europe, 902-928 USA, 2.4Ghz global</li><li>• PHY and MAC layers only</li><li>• Selectable levels of security – privacy (encryption), sender authentication, message integrity</li><li>• Flexible protocol design suitable for many applications</li><li>• <b>handshake protocol for transfer reliability</b></li></ul> | <ul style="list-style-type: none"><li>• Spectrum: 1.4 MHz bandwidth</li><li>• Battery life of devices extended through sleep mode or power saving mode (PSM) and through extended discontinuous reception (LTE eDRX)</li><li>• High Latency Communication<ul style="list-style-type: none"><li>◦ Support for extended coverage</li><li>◦ LTE-M Half Duplex Mode/Full Duplex</li><li>◦ Support of Category M1 device</li><li>◦ VoLTE support</li><li>◦ SMS</li></ul></li></ul> |
|                      | <ul style="list-style-type: none"><li>• Connected white goods</li><li>• Connected animals</li><li>• Logistics</li><li>• Smart metering</li><li>• Street lighting</li><li>• Smart parking</li><li>• Asset tracking</li><li>• Industrial sensor monitoring</li></ul> | <ul style="list-style-type: none"><li>• Smart agriculture</li><li>• Industrial sensor reading</li><li>• Medical sensor reading</li><li>• <b>Smart cities</b></li><li>• Environmental sensors</li><li>• Smart metering</li><li>• Logistics</li><li>• Home automation</li></ul>                                                                                                                                                                                                                                         | <ul style="list-style-type: none"><li>• General automation</li><li>• Home automation</li><li>• Smart Grids</li><li>• Industrial sensor networks</li><li>• Asset location</li></ul>                                                                                                                                                                                                                          | <ul style="list-style-type: none"><li>• Healthcare devices</li><li>• Voice (if supported by carrier)</li><li>• Smart city sensors</li><li>• <b>Environmental sensors</b></li><li>• Emergency data</li><li>• Precision tracking</li><li>• Wearables</li><li>• Telematics</li></ul>                                                                                                                                                                                             |

# Industrial IoT Wireless Technologies [CISCO]



|                                                    | LoRaWAN                        | Resilient Mesh                       | WiFi                                            | 4G/LTE – 5G                                                                                                                                  |
|----------------------------------------------------|--------------------------------|--------------------------------------|-------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| Topology                                           | Point to multipoint            | Mesh Point to Multipoint (Leaf mode) | Point to Multipoint Mesh                        | Point to multipoint                                                                                                                          |
| Coverage Range<br>(Radio signal is the real value) | ~ 2k-10km                      | ~1.5km per hop up to 8 hops          | ~100m (300 feet)                                | ~ 2k-10km (cell dependent)                                                                                                                   |
| Data Rate                                          | 250bs-21kbs                    | 50kbps -1.2Mbps                      | 11Mbs (.b)<br>1.7 Gbs (.ac W2)<br>9.6 Gbs (.ax) | 27kbs(DL)/65kbs(UL) NB-IOT HDx<br>300kbs/375kbs LTE Cat M1<br>300 Mbps (DL)/50Mbps (UL) LTE Cat.6 to 5G (500Mbs UL/5Gbs DL) on today's modem |
| Public SP vs Private Networks                      | Private/Public SP              | Private                              | Private/Public SP                               | Public SP/Private (i.e, US CBRS)                                                                                                             |
| Batteries powered devices                          | Optimized lifetime (+10 years) | Not in FAN 1.0                       | Limited lifetime (months)                       | NB-IOT provides good lifetime                                                                                                                |
| Eco-system (endpoints)                             | *****                          | *                                    | *****                                           | ***                                                                                                                                          |
| TCO                                                | Low                            | Low-Medium                           | Medium-High                                     | Medium-High                                                                                                                                  |



- [VB17] Ovidiu Vermesan and Joel Bacquet, "**Cognitive Hyperconnected Digital Transformation: Internet of Things Intelligence Evolution**", River Publishers, 2017.
- [BHC+18] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "**The Industrial Internet of Things (IIoT): An Analysis Framework**", Computers in Industry 101, October 2018.
- [BS19] Rajkumar Buyya and Satish Narayana Srirama, "**Fog and Edge Computing: Principles and Paradigms**", Wiley Series on Parallel and Distributed Computing, 2019.
- [RS19] Ammar Rayes and Samer Salam, "**Internet of Things from Hype to Reality: The Road to Digitization**", 2<sup>nd</sup> Edition, Springer, 2019.
- [LEA20] Perry Lea, "**IoT and Edge Computing for Architects: Implementing edge and IoT systems from sensors to clouds with communication systems, analytics, and security**", 2<sup>nd</sup> Edition, Packt Publishing, 2020.
- [GG20] Patrick Grossetete and Stephen Goodman, "**Wireless Technologies and Use Cases in Industrial IoT**", BKKIOT-1775, Cisco Live, January 2020.
- Lecture Course: Wireless and Mobile Networking  
<https://www.cse.wustl.edu/~jain/cse574-20/index.html>  
[https://learning.oreilly.com/videos/internet-of-things/9780137592135/9780137592135-iott\\_00\\_00\\_00\\_00/](https://learning.oreilly.com/videos/internet-of-things/9780137592135/9780137592135-iott_00_00_00_00/)



# **ITCS447**

## **Lecture 9**

# **IoT Communication and Networking**

## **Part 2: IoT Application Protocols**

**Asst. Prof. Dr. Thitinan Tantidham**



ห้ามมิให้นักศึกษาทำข้า ดัดแปลง หรือใช้ประโยชน์จากการอันมีลิขสิทธิ์ปราการอยู่ในระบบการศึกษาอิเล็กทรอนิกส์ (E-Learning) ของมหาวิทยาลัย ไม่ว่าจะทั้งหมดหรือบางส่วน โดยไม่ได้รับอนุญาตจากมหาวิทยาลัย นอกจากนี้จากการศึกษาส่วนบุคคล ทั้งนี้ การทำข้า ดัดแปลง หรือเผยแพร่ต่อสาธารณะชื่องานอันมีลิขสิทธิ์ จะมีโทษปรับตั้งแต่ 20,000 บาท ถึง 200,000 บาท และหากเป็นการกระทำเพื่อการค้า จะมีโทษจำคุกตั้งแต่ 6 เดือน ถึง 10 ปี หรือปรับตั้งแต่ 100,000 บาท ถึง 800,000 บาท หรือทั้งจำทั้งปรับ

Unless allowed by Mahidol University, the User shall not copy, modify, or exploit in part or in whole of the copyrighted materials on the Platform, other than for your own individual study. The copy, modify or communication to public of the copyrighted materials shall be inflicted with a fine from 20,000 Baht up to 200,000 Baht. If the offence is committed with the commercial purpose, the offender shall be inflicted with imprisonment for a term from 6 months up to 10 years or a fine from 100,000 Baht up to 800,000 Baht or both imprisonment and fine.

# Outline



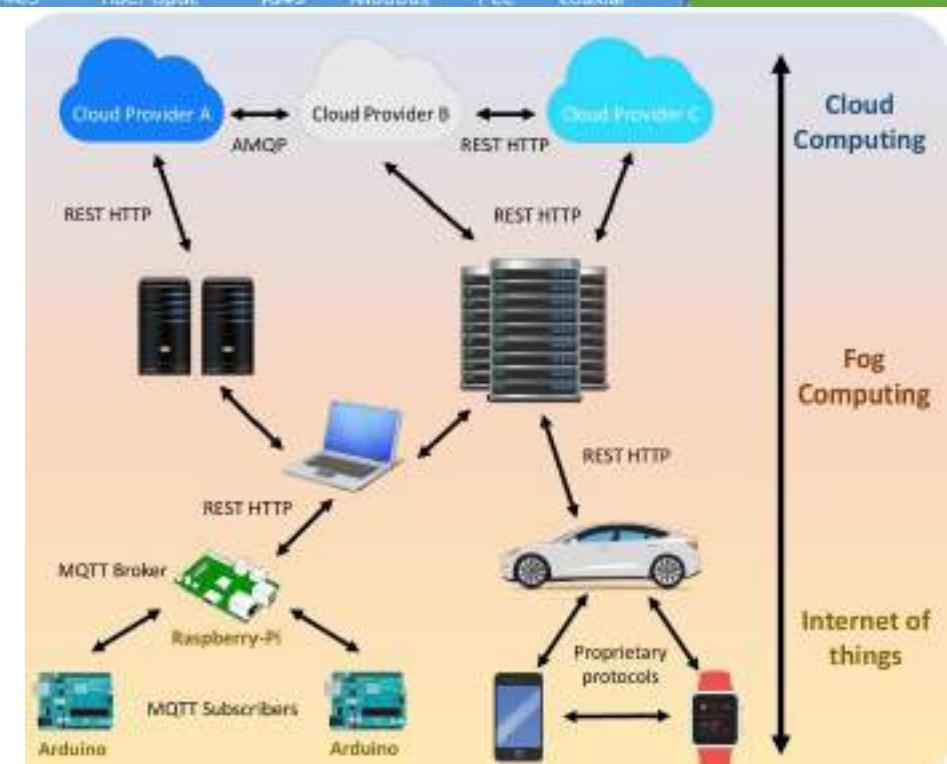
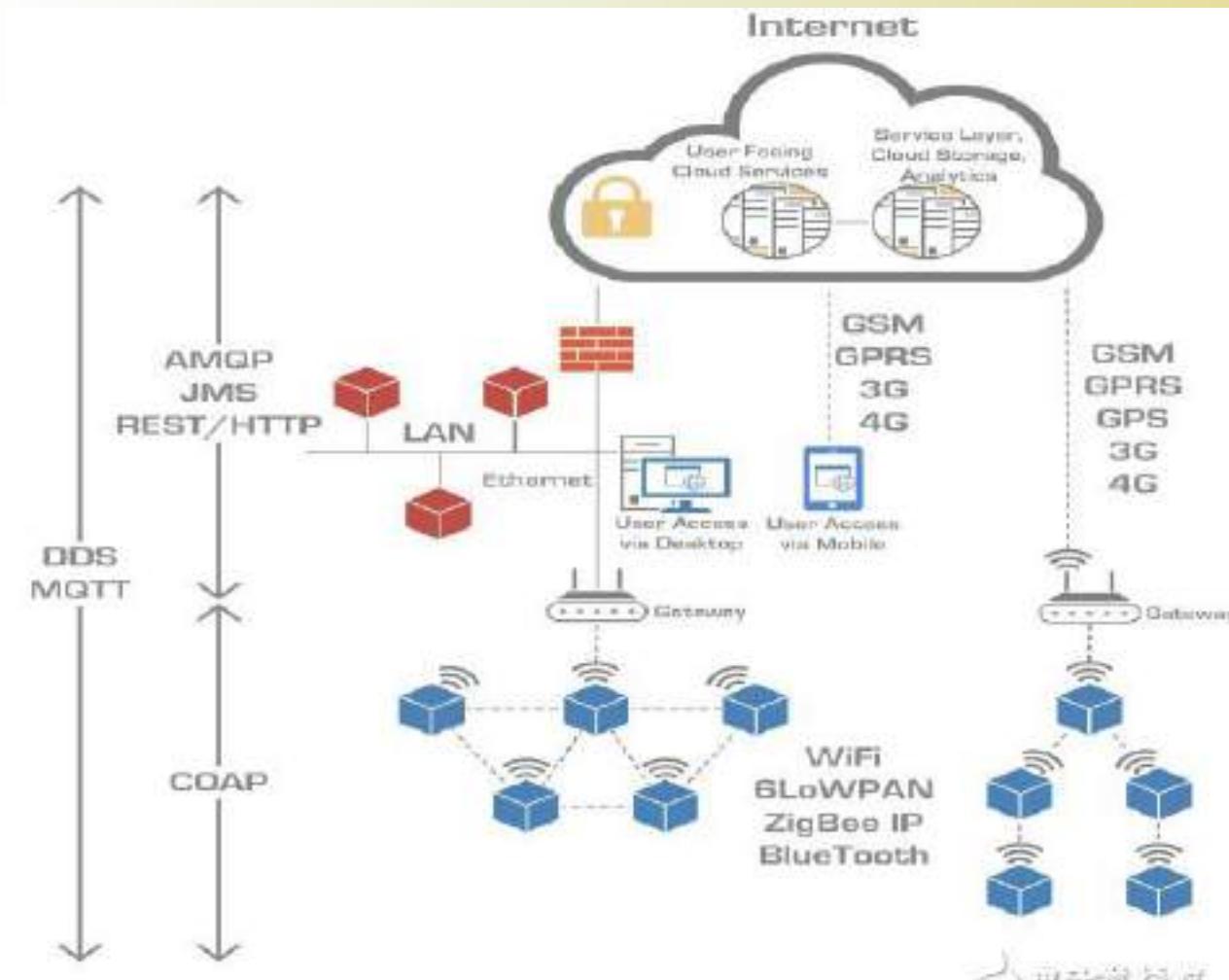
- Review: Wireless Communications in IoT
- Overview of IoT Application Protocols
- Survey of IoT Platform Protocols
- Open Source IoT EcoSystem
- List of IoT Application Protocols
- Web of Things (WoT)
- HTTP/REST API
- CoAP
- MQTT
- Comparison



# Wireless Communications in IoT

| Technology  | Standard         | Frequency            | Penetration | Range    | Max Data Rate  | Channel Bandwidth | Chipset Cost |
|-------------|------------------|----------------------|-------------|----------|----------------|-------------------|--------------|
| NFC/RFID    | ISO/ICE 18092    | 13.56 MHz            | High        | <20 cm   | 424 kbps       | 106–424 Mbps      | \$0.1+       |
| Bluetooth   | IEEE 802.15      | 2.4/2.5 GHz          | Low         | 50–100 m | 2 Mbps         | 2 MHz             | \$5+         |
| Wi-Fi       | IEEE 802.11      | 2.4/5.0 GHz          | Low         | 100 m    | 54 Mbps        | 22 MHz            | \$1.5-30+    |
| Zigbee      | IEEE 802.15.4    | 868/915 MHz, 2.4 GHz | Low/High    | <1 km    | 250 kbps       | 2 MHz             | \$2-20+      |
| DASH7       | ISO/IEC 18000-7  | 433/868/915 MHz      | High        | 0–5 km   | 167 kbps       | up to 1.75 MHz    | \$3.00+      |
| Weightless  | Weightless P/N/W | Multiple             | Low/High    | 5 km     | 100 kbps       | 200 Hz–12.5 KHz   | ~\$2.00      |
| LoRa        | Various          | 868/915 MHz          | Low         | 25 km    | 50 kbps        | 125/250/500 kHz   | ~\$2.00      |
| Ingenu-RPMA | Ingenu-RPMA      | 2.4 GHz              | Low         | 15 km    | 20 kbps        | 1 MHz             | rental       |
| SigFox      | SigFox           | 915–928 MHz          | Low/High    | 40 km    | 100 bps        | 100 Hz            | \$0.25+      |
| 3G          | UMTS/W-CDMA      | 0.4–3 GHz            | Low/High    | 5–35 km  | 0.38–21.6 Mbps | 3.6–21 Mbps       | varies       |
| 4G/LTE      | 3GPP-LTE         | 0.6–6 GHz            | Low/High    | 5–100 km | 100–300 Mbps   | 100 Mbps+         | \$6.5+       |
| 5G          | 5GTF/5G-SIG      | 0.6–4/100 GHz        | Low/High    | 5–150 km | 10 Gbps        | 500 Mbps+         | \$70+        |

# Overview of IoT Application Protocols



# Survey of IoT Platform Protocols [1]



| IoT Platform,<br><i>Year of First<br/>General Availability (GA)</i> | Protocol                                                                                                                                                                         |
|---------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Azure IoT Hub [166],<br>2014                                        | HTTP(S), MQTT, MQTT over WebSocket,<br>AMQP, AMQP over WebSocket<br><i>custom protocols transmit via a gateway</i>                                                               |
| Google IoT Core [167],<br>2018                                      | HTTP(S), MQTT<br><i>custom protocols transmit via a gateway</i>                                                                                                                  |
| IBM Watson IoT [168]<br>2014                                        | HTTP(S), MQTT                                                                                                                                                                    |
| AWS IoT Core [169]<br>2015                                          | HTTP(S), MQTT, MQTT over WebSocket,<br>WebSocket                                                                                                                                 |
| Alibaba IoT [170]<br>2015                                           | HTTP(S), CoAP, MQTT, MQTT over<br>WebSocket, WebSocket,<br><i>support network types: 3G, 4G, NB-IoT &amp; LoRa</i>                                                               |
| Oracle IoT [171]<br>2016                                            | HTTP(S), CoAP, MQTT, AMQP, XMPP,<br>WebSocket                                                                                                                                    |
| Siemens MindSphere [172]<br>2016                                    | HTTP(S), CoAP, MQTT, AMQP, XMPP,<br><i>supports wide range of device protocols via field<br/>gateways (e.g. MindConnect) such as OPC UA,<br/>LoRaWAN, Modbus, 6LoWPAN, LwM2M</i> |
| Bosch IoT Hub [173]<br>2017                                         | HTTP(S), MQTT, AMQP, LoRaWAN                                                                                                                                                     |
| Cisco Kinetic [174]<br>2017                                         | HTTP(S), MQTT, AMQP, WebSocket<br><i>custom protocols transmit via a gateway (e.g.<br/>Cisco IoT Gateway)</i>                                                                    |
| Eclipse Hono [175]<br>2018                                          | HTTP(S), CoAP, MQTT, AMQP<br><i>uses AMQP 1.0 as primary messaging protocol<br/>custom protocols transmit via a gateway</i>                                                      |

# Survey of IoT Platform Protocols [2]



| Features                | AWS                                                                | Microsoft Azure                               | Google Cloud IoT                         | IBM Watson IoT                                                                    | Oracle IoT                        |
|-------------------------|--------------------------------------------------------------------|-----------------------------------------------|------------------------------------------|-----------------------------------------------------------------------------------|-----------------------------------|
| Security                | Link Encryption (TLS), Authentication (Sig V4, X.509)              | Link Encryption (SSL/TSL)                     | SSL/TLS                                  | Link Encryption (TLS), Authentication (IBM cloud SSO), Identity Management (LDAP) | REST API                          |
| Data analytics          | Real Time analytics (Rule engine, Kinesis, AWS Lambda)             | Real Time analytics                           | Real Time analytics (Cloud IoT Core)     | Real Time analytics (IBM IoT Real time insights)                                  | Real Time analytics               |
| Protocols               | MQTT, HTTP1.1                                                      | MQTT, HTTP, AMQP                              | MQTT                                     | MQTT, HTTPS                                                                       | MQTT, HTTP                        |
| Visualization tool      | AWS IoT dashboard                                                  | web portal                                    | Google data studio (Dashboard)           | web portal                                                                        | web portal                        |
| Data format             | JSON                                                               | JSON                                          | JSON                                     | JSON, CSV                                                                         | CSV, REST API                     |
| Application Environment | Java, C, NodeJs, Javascript, Python, SDK for Arduino, iOS, Android | .Net, UWP, Java, C, NodeJS, Ruby, Android,iOS | Go,Java, Python, .NET, NodeJS, php, Ruby | C#, C, Python, Java, NodeJS                                                       | Java, iOS, Javascript, C, Android |



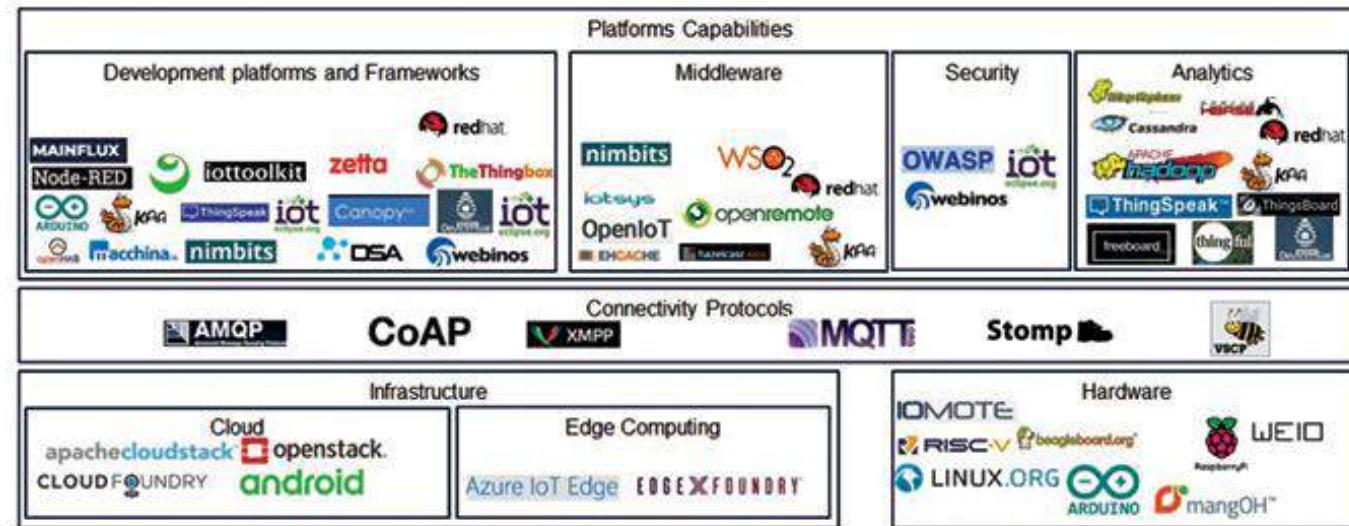
# Survey of IoT Platform Protocols [3]

| IoT Software                                        | Integration                                                                       | Data collection                                                          | Analyses                                         | Visualization                                     | Data Base                                   |
|-----------------------------------------------------|-----------------------------------------------------------------------------------|--------------------------------------------------------------------------|--------------------------------------------------|---------------------------------------------------|---------------------------------------------|
| ThingsBoard                                         | REST APIs,<br>MQTT APIs                                                           | HTTP, MQTT,<br>OPC-UA, CoAP                                              | Real time<br>analytics (Apache<br>Spark, kafka). | Yes                                               | PostgreSQL,<br>Cassandra,<br>HSQLDB         |
| Kaa                                                 | Portable SDK<br>available to<br>integrate any<br>particular platform,<br>REST API | MQTT, CoAP,<br>XMPP, TCP,<br>HTTP, WiFi,<br>Ethernet,<br>Zigbee          | Real time                                        | Yes (Doesn't<br>have own<br>dashboards)<br>NoSQL. | MongoDB,<br>Cassandra,<br>Hadoop, Oracle    |
| WSO2                                                | REST APIs                                                                         | HTTP, WSO2<br>ESB, MQTT                                                  | Yes, WSO2<br>Data Analytics<br>Server            | Yes                                               | Oracle,<br>PostgreSQL,<br>MySQL, or MS SQL  |
| Site Where                                          | REST API                                                                          | MQTT, AMQP,<br>Stomp,<br>WebSockets,<br>and direct socket<br>connections | Real-time<br>analytics<br>(Apache Spark)         | No                                                | MongoDB,<br>HBase,<br>InfluxDB              |
| Thing Speak                                         | REST API,<br>MQTT APIs                                                            | HTTP                                                                     | MATLAB<br>Analytics                              | No                                                | MySQL                                       |
| DeviceHive                                          | REST API,<br>MQTT APIs                                                            | REST API,<br>WebSockets or<br>MQTT                                       | Real-time<br>analytics<br>(Apache Spark)         | Yes (Doesn't<br>have own<br>dashboards)           | PostgreSQL,<br>SAP Hana<br>DB               |
| Zetta<br>Distributed Services<br>Architecture (DSA) | REST APIs                                                                         | HTTP                                                                     | Using Splunk                                     | No                                                | Unknown                                     |
| Thinger.io                                          | REST APIs                                                                         | MQTT, CoAP<br>and HTTP                                                   | Yes                                              | No                                                | ETSDB<br>Embedded<br>Time Series<br>MongoDB |

# Open Source IoT EcoSystem



- There is a need for organizations to provide a validated, modular, flexible IoT solution built to be **open, interoperable and cost-effective**.
- The solution should deliver **end-to-end open source IoT** that addresses enterprise level needs.
- The characteristics of open source IoT architecture are:
  - Loosely coupled, modular and secure
  - Platform independent
  - Scalable, flexible and can be deployed anywhere
  - Based on open standards
  - Streaming analytics and machine learning
  - Open and interoperable on the hybrid cloud
  - Application agility and integration
  - No vendor lock-in, no rigid architectures or proprietary formats and components



Ref:

<https://www.opensourceforu.com/2020/02/leveraging-open-source-tools-for-iot/>

<https://vizah.ch/en/developing-iot-applications-best-technologies-and-tools-for-iot-developers/>



# List of IoT Application Protocols (1/2)

- **CoAP (Constrained Application Protocol):** RFC 7252
  - CoAP makes use of the UDP protocol RESTful architecture (based on HTTP protocol).
  - It is used within mobiles and social network based applications.
- **MQTT:** OASIS
  - This messaging protocol is used for remote monitoring in IoT.
  - MQTT connects devices and networks with applications and middleware using **hub-and-spoke** architecture.
  - The MQTT protocol provides efficient information routing functions to small, cheap, low-memory and power consuming devices in vulnerable and low bandwidth based networks.
- **XMPP (Extensible Messaging and Presence Protocol):** RFC3920/RFC6120 - (<https://xmpp.org/>)
  - P2P or Pub/Sub methodology.
  - This is a communications IoT protocol for **message-oriented middleware based on the XML language**.
  - It enables real-time exchange of structured yet extensible data between any two or more network entities.
  - XMPP enables messaging applications to attain authentication, access control, hop-by-hop and end-to-end encryption.
- **AMQP (Advanced Message Queuing Protocol):** ISO/IEC 19464:2014 - (<https://www.amqp.org/>)
  - It supports reliable communication via message delivery assurance primitives like ‘at-most once’, ‘at least once’ and ‘exactly once’.
  - This protocol enables **client applications to talk to the broker, route and store messages within a broker**, and interact with the AMQP model.
- **DDS (Data Distribution Service):** OMG
  - This is used for real-time machine-to-machine communication.
  - Middleware connectivity framework
  - It enables scalable, real-time, dependable, high-performance and interoperable data exchange via the **publish-subscribe methodology**.
  - DDS can be deployed in platforms ranging from low-footprint devices to the cloud, and supports efficient bandwidth usage as well as the agile orchestration of system components.
- Others: OPC UA (Open Platform Communications—Unified Architecture), NDN (Named Data Networking) for IoT



# List of IoT Application Protocols (2/2)

- **VSCP (Very Simple Control Protocol):** <https://www.vscp.org/>
  - Device discovery and identification.
  - Device configuration.
  - Autonomous device functionality.
  - Secure update of device firmware
  - A solution from sensor to UI
  - It uses CAN, RS-232, Ethernet, TCP/IP, MQTT, 6LowPan or whatever as its transport mechanism and work over cable and over the air.
- **STOMP (Simple Text Oriented Messaging Protocol):** <https://stomp.github.io/>
  - STOMP provides an interoperable wire format
  - STOMP clients can communicate with any STOMP message broker to provide easy and widespread messaging interoperability among many languages, platforms and [brokers](#).
  - STOMP is a very simple and easy to implement protocol, coming from the HTTP school of design; the server side may be hard to implement well, but it is very easy to write a client to get yourself connected. For example you can use Telnet to login to any STOMP broker and interact with it!
  - Many developers have told us that they have managed to write a STOMP client in a couple of hours to in their particular language, runtime or platform into the STOMP network. So if your favored language/runtime of choice does not offer a good enough STOMP client don't be afraid to write one.
- **QUIC (Quick UDP Internet Connections):** <https://quicwg.org/>
  - A flexible multiplexed and secure general-purpose transport protocol that supports multiplexed streams.



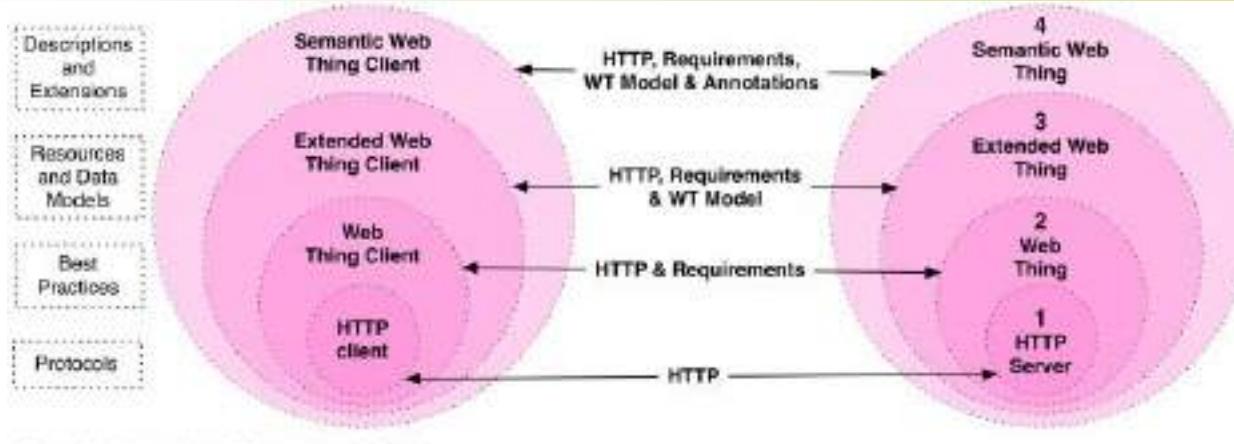
Ref:

<http://dave.thehorners.com/aboutme/cooking-recipes/499-messaging-messagequeue-pubsub-stomp-amqp-mqtt>

# Web of Things (WoT): Model (1/2)



Ref: <https://www.w3.org/blog/2017/01/web-thing-model-member-submission/>



Source: Building the Web of Things book <http://bit.ly/2oJLWfj>  
Creative Commons Attribution 4.0

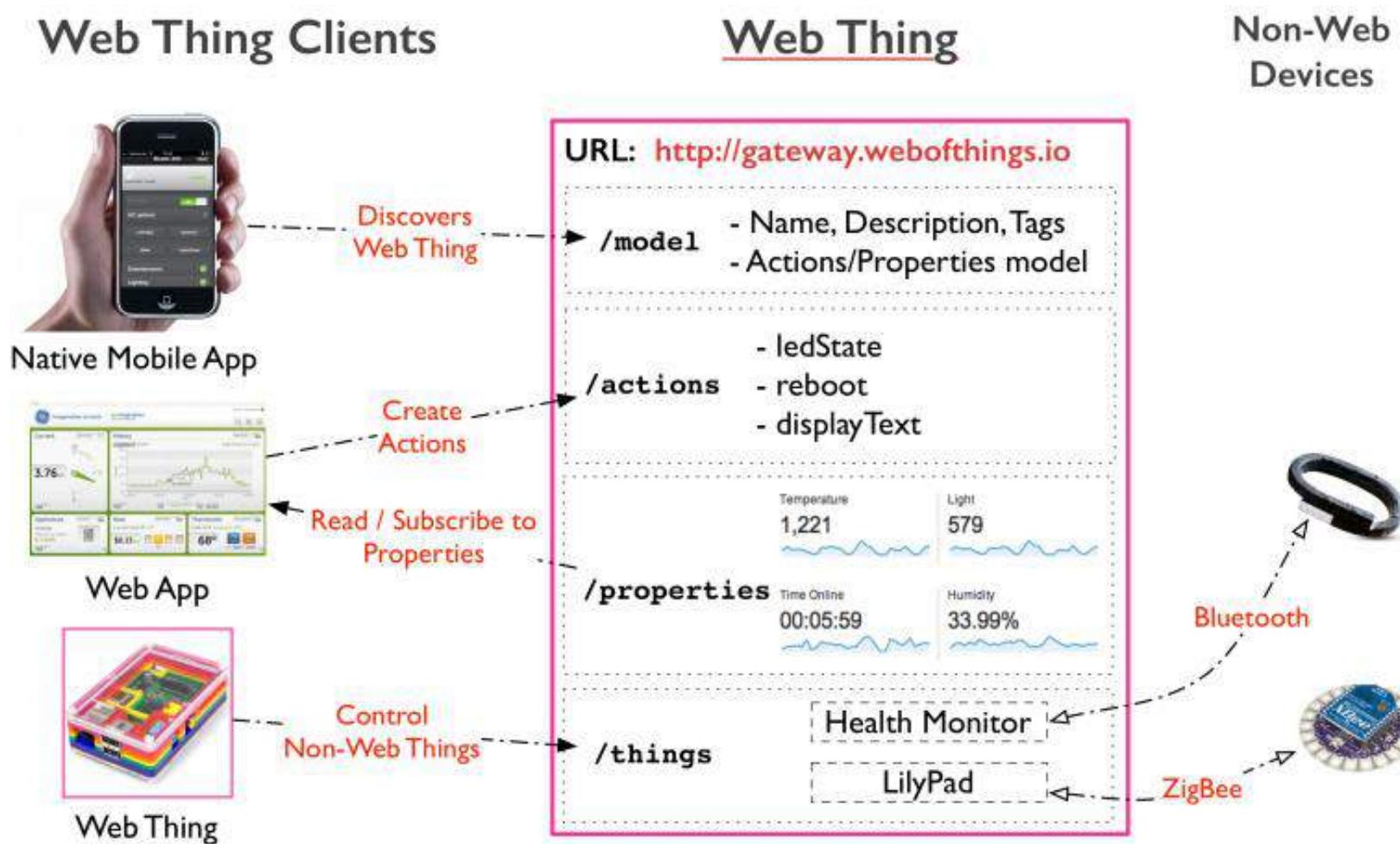
- **Things** – A web Thing can be a gateway to other devices that don't have an internet connection. This resource contains all the web Things that are proxied by this web Thing. This is mainly used by clouds or gateways because they can proxy other devices.
- **Model** – A web Thing always has a set of metadata that defines various aspects about it such as its name, description, or configurations.
- **Properties** – A property is a variable of a web Thing. Properties represent the internal state of a web Thing. Clients can subscribe to properties to receive a notification message when specific conditions are met; for example, the value of one or more properties changed.

- **Actions** – An action is a function offered by a web Thing. Clients can invoke a function on a web Thing by sending an action to the web Thing.
  - ✓ Examples of actions are “open” or “close” for a garage door, “enable” or “disable” for a smoke alarm, and “scan” or “check in” for a bottle of soda or a place.
  - ✓ The direction of an action is usually from the client to the web Thing.
  - ✓ Actions represent the public interface of a web Thing and properties are the private parts. Much like in any programming languages, you can access the public interface, and whatever is private remains accessible only for privileged parties, like the instance itself or, in this case, the web Thing. But limiting access to actions – that is, the public interface – also allows you to implement various control mechanisms for external requests such as access control, data validation, updating a several properties atomically, and the like.

# Web of Things (WoT): Model (2/2)



Ref: <https://www.w3.org/blog/2017/01/web-thing-model-member-submission/>



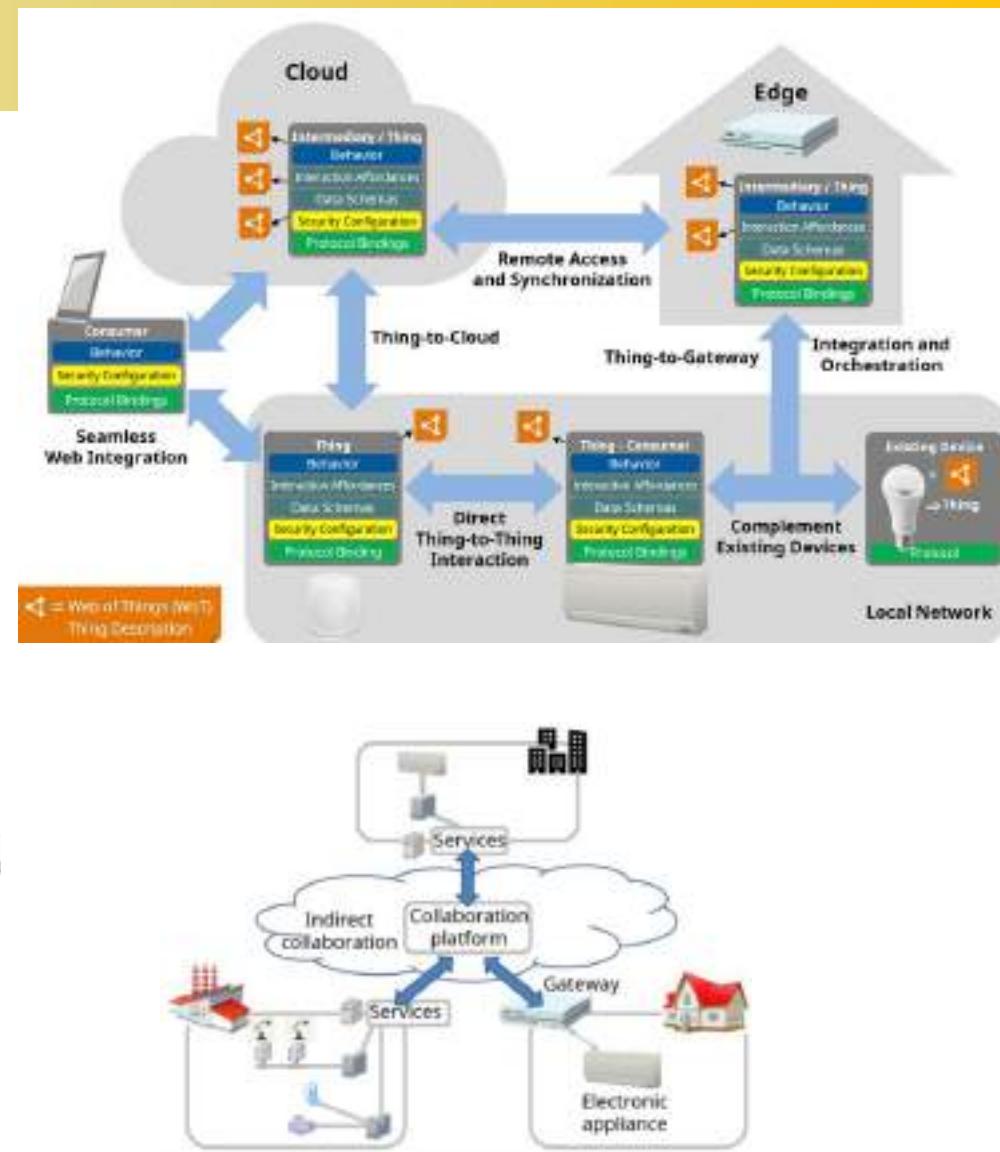
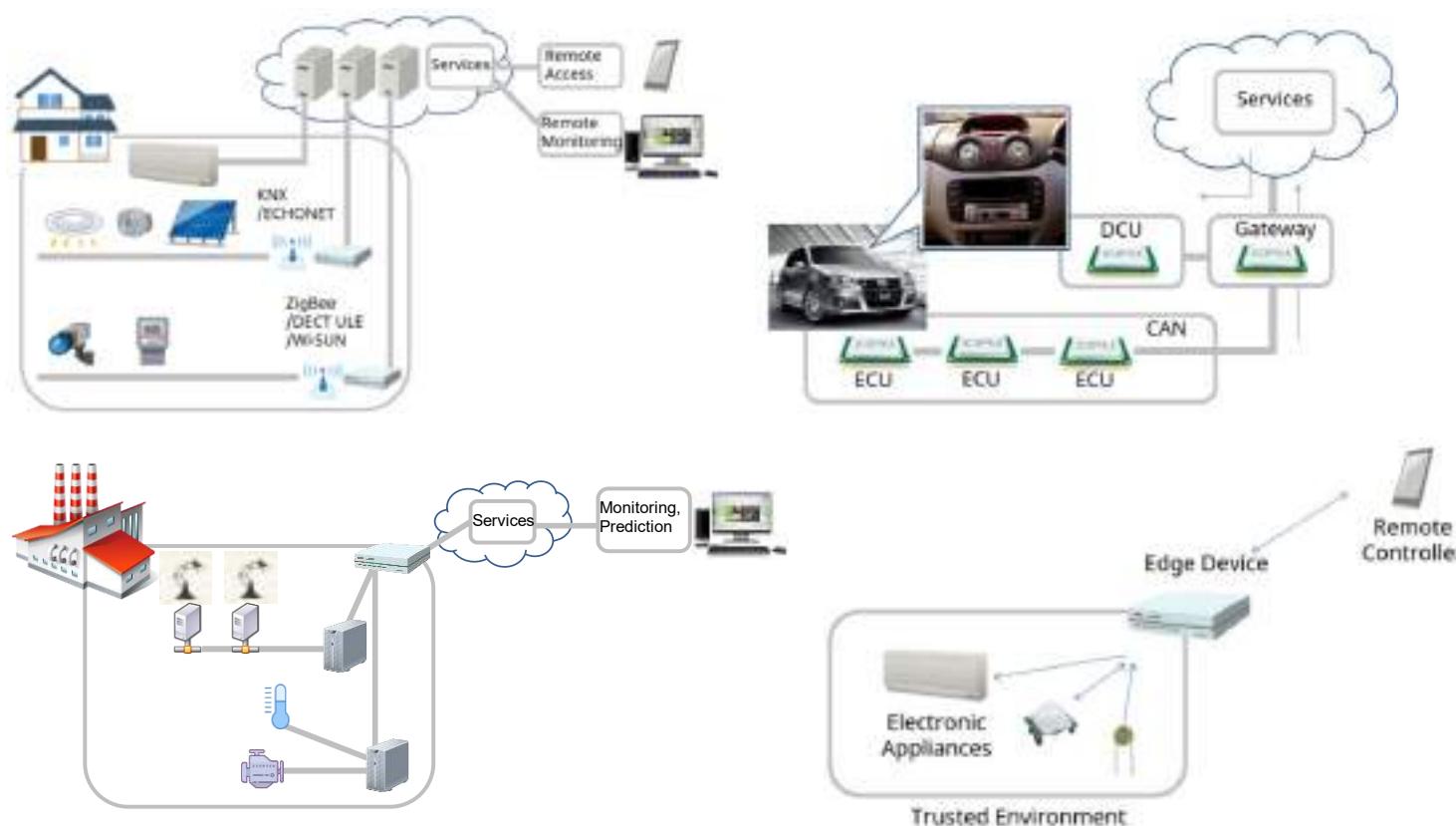
Source: Building the Web of Things: book.webofthings.io  
Creative Commons Attribution 4.0



# Web of Things (WoT): Architecture

Ref: <https://www.w3.org/2020/04/pressrelease-wot-rec.html.en>

- The Web of Things is applicable to multiple IoT domains, including Smart Home, Industrial, Smart City, Retail, and Health applications, where usage of the W3C WoT standards can simplify the development of IoT systems that combine devices from multiple vendors and ecosystems.



# Web Thing API (1/2)



Ref: <https://iot.mozilla.org/wot/>

- The [Web Thing Description](#) provides a vocabulary for describing physical devices connected to the World Wide Web in a machine readable format with a default JSON encoding.
- Common device capabilities can be specified using optional [semantic annotations](#).
- The [Web Thing REST API](#) and [Web Thing WebSocket API](#) allow a web client to access the properties of devices, request the execution of actions and subscribe to events representing a change in state.

A simple Thing Description

```
{  
  "id": "https://mywebthingserver.com/things/switch",  
  "title": "On/Off Switch",  
  "description": "A web connected switch",  
  "properties": {  
    "on": {  
      "title": "On/Off",  
      "type": "boolean",  
      "description": "Whether the lamp is turned on",  
      "links": [ { "href":  
        "/things/switch/properties/on" } ]  
    }  
  }  
}
```

# Web Thing API (2/2)



```
{  
    "@context": "https://iot.mozilla.org/schemas/",  
    "@type": ["Light", "OnOffSwitch"],  
    "id": "https://mywebthingserver.com/things/lamp",  
    "title": "My Lamp",  
    "description": "A web connected lamp",  
    "properties": {  
        "on": {  
            "@type": "OnOffProperty",  
            "type": "boolean",  
            "title": "On/Off",  
            "description": "Whether the lamp is turned on",  
            "links": [{"href": "/things/lamp/properties/on"}]  
        },  
        "brightness": {  
            "@type": "BrightnessProperty",  
            "type": "integer",  
            "title": "Brightness",  
            "description": "The level of light from 0-100",  
            "minimum": 0,  
            "maximum": 100,  
            "links": [{"href": "/things/lamp/properties/brightness"}]  
        }  
    },  
    "actions": {  
        "fade": {  
            "@type": "FadeAction",  
            "title": "Fade",  
            "description": "Fade the lamp to a given level",  
            "input": {  
                "type": "object",  
                "properties": {  
                    "level": {  
                        "type": "integer",  
                        "minimum": 0,  
                        "maximum": 100  
                    },  
                    "duration": {  
                        "type": "integer",  
                        "minimum": 0,  
                        "unit": "milliseconds"  
                    }  
                }  
            },  
            "links": [{"href": "/things/lamp/actions/fade"}]  
        }  
    }  
}
```

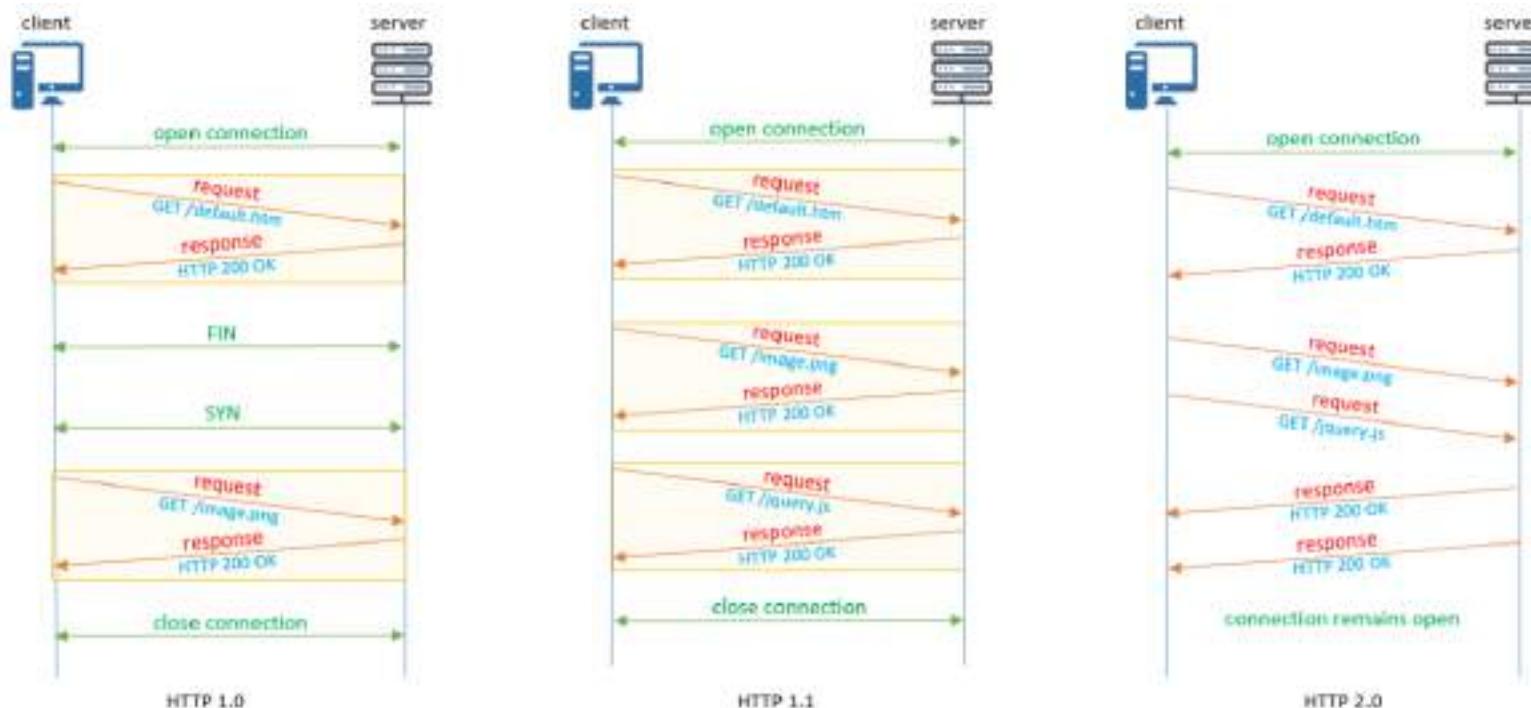
```
    "events": {  
        "overheated": {  
            "title": "Overheated",  
            "@type": "OverheatedEvent",  
            "type": "number",  
            "unit": "degree celsius",  
            "description": "The lamp has exceeded its safe operating temperature",  
            "links": [{"href": "/things/lamp/events/overheated"}]  
        }  
    },  
    "links": [  
        {  
            "rel": "properties",  
            "href": "/things/lamp/properties"  
        },  
        {  
            "rel": "actions",  
            "href": "/things/lamp/actions"  
        },  
        {  
            "rel": "events",  
            "href": "/things/lamp/events"  
        },  
        {  
            "rel": "alternate",  
            "href": "wss://mywebthingserver.com/things/lamp"  
        },  
        {  
            "rel": "alternate",  
            "mediaType": "text/html",  
            "href": "/things/lamp"  
        }  
    ]  
}
```

Ref: <https://iot.mozilla.org/wot/>

# HTTP (HyperText Transfer Protocol)



## Client (Request)/ Server (Response)



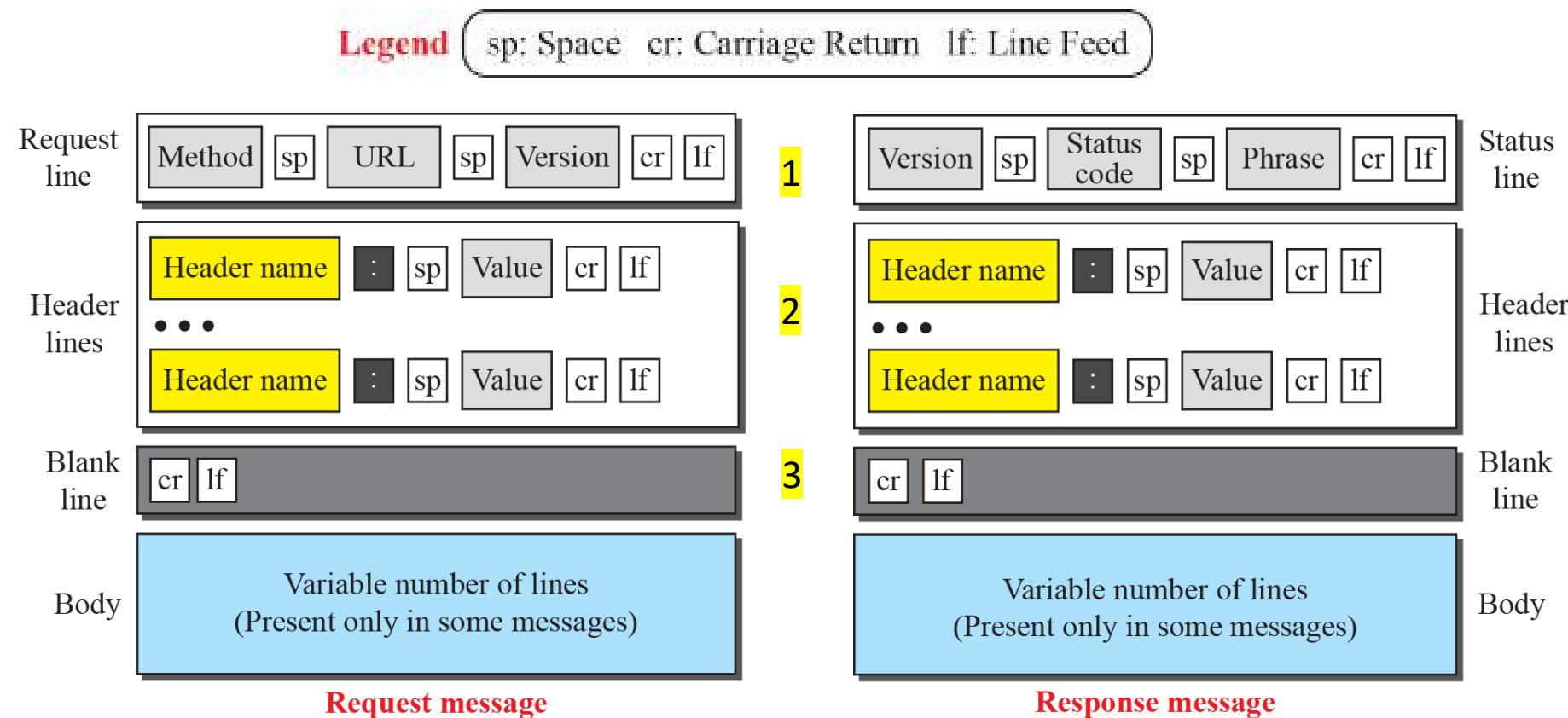
HTTP Client: Web Browser  
HTTP Server: Web Server

Web Server:  
<https://httpd.apache.org>

- An application-level and stateless protocol based on request/response and client/server model for data communication over the World Wide Web.
- **TCP Connection, port 80**
- One of the key features of HTTP is content negotiation of data representation.
- This enables different heterogeneous devices built independently of the data to be shared.



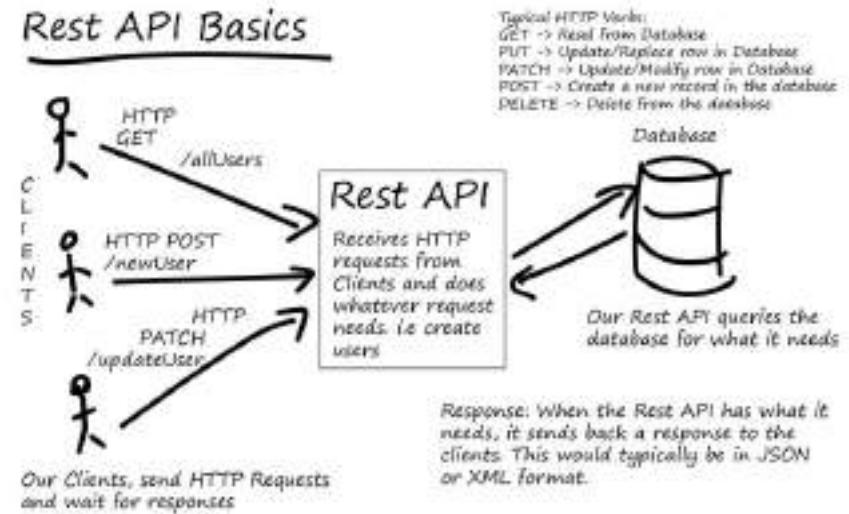
## The Format of Two Message Types



# REST (Representational State Transfer)



- REST was introduced and defined in 2000 by [Roy Fielding](#) in his doctoral dissertation.
- **It is NOT standard**, but REST is an architectural style.  
Note: Standards like HTTP, URL
  - Resource Representations: XML/HTML/GIF/JPEG/etc.
  - Resource Types, MIME Types: text/xml, text/html, image/gif, image/jpeg, etc.
- The motivation for REST was to **capture the characteristics of the Web which made the Web successful**
  - URI Addressable resources
  - HTTP Protocol
  - Make a Request – Receive Response – Display Response
  - Stateless: simple request/response. No conversational state. Easy to scale. The request must contain all that is required for the reply to be computed.
- **HATEOAS** (Hypermedia As The Engine Of Application State)



- **Exploits the use of the HTTP protocol**
  - HTTP POST
  - HTTP GET
  - HTTP PUT
  - HTTP DELETE



## Examples (JSON-LD)

- GET /basement/water/temperature      200 OK  
application/text  
40.5 F
- GET /basement/water/volume      200 OK  
application/text  
200 G

JSON-LD format specification for linked open data to interoperate at web-scale

Ref:

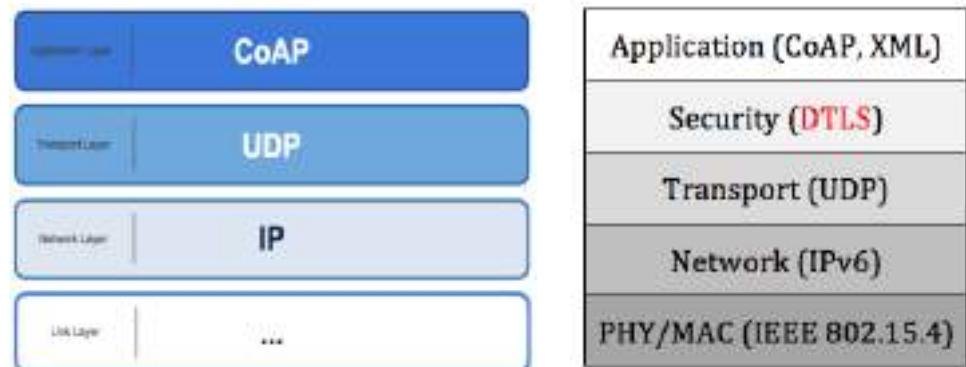
<https://www.w3.org/TR/json-ld-api/>

<https://www.w3.org/2018/json-ld-wg/>

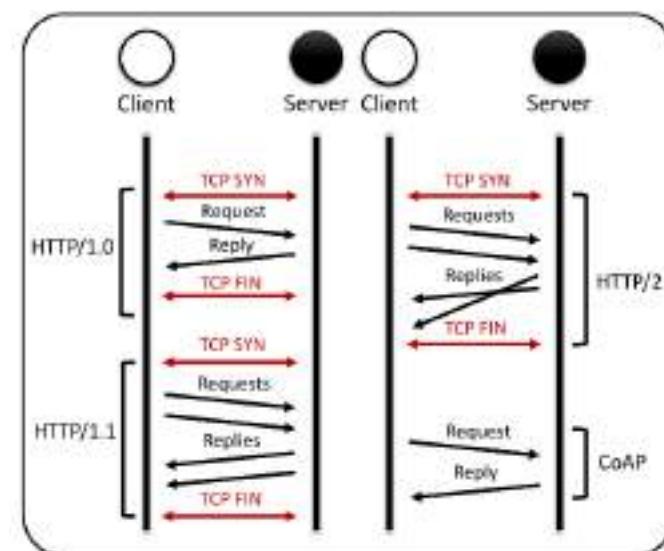
# CoAP (Constrained Application Protocol)



- REST-based web transfer protocol
- Manipulates Web resources using the same methods as HTTP: GET, PUT, POST, and DELETE
- Subset of HTTP functionality redesigned for low power embedded devices such as sensors (for IoT and M2M)
- TCP overhead is too high and its flow control is not appropriate for short-lived transactions
- UDP has lower overhead and supports multicast



DTLS (Datagram Transport Layer Security)



# CoAP (Constrained Application Protocol)

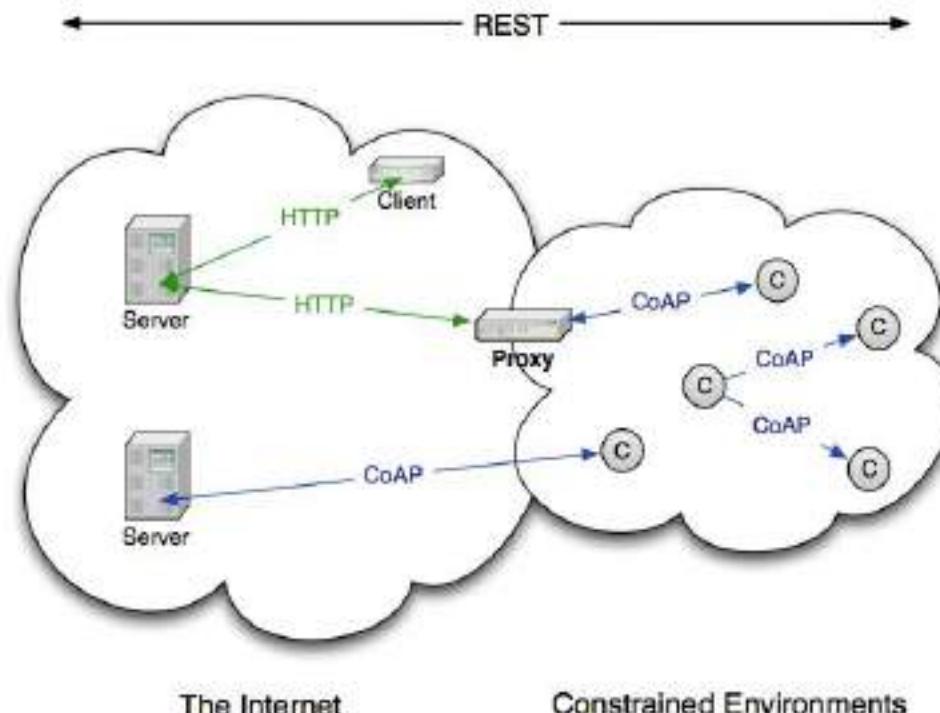


- Four message types:
  - **Confirmable** – requires an ACK
  - **Non-confirmable** – no ACK needed
  - **Acknowledgement** – ACKs a Confirmable
  - **Reset** - indicates a Confirmable message has been received but context is missing for processing
- CoAP provides reliability **without using TCP as transport protocol**
- CoAP enables **asynchronous** communication
  - e.g., when CoAP server receives a request which it cannot handle immediately, it first ACKs the reception of the message and sends back the response in an off-line fashion
- Also supports multicast and congestion control

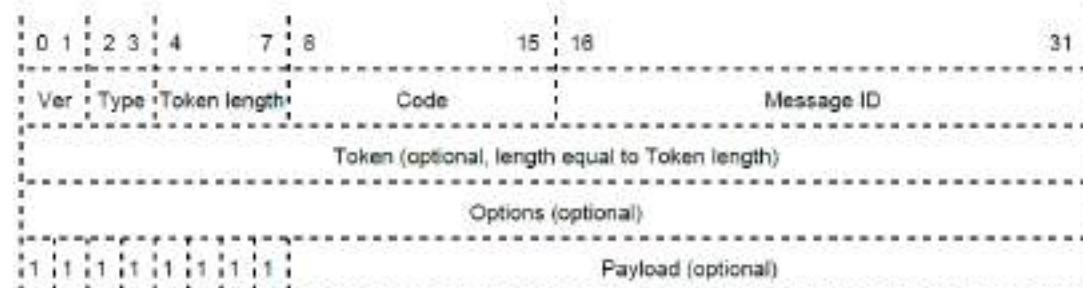


Copper plugin in Firefox  
Californium

# CoAP (Constrained Application Protocol)



- A RESTful protocol
- Both synchronous and asynchronous
- For constrained devices and networks
- Specialized for M2M applications
- Easy to proxy to/from HTTP
- Implement: <https://libcoap.net>
- Benchmark:  
<https://www.eclipse.org/californium>



Learn more:

<https://tools.ietf.org/id/draft-keranen-t2trg-rest-iot-05.html>



## Introduction

- MQTT was invented by **Andy Stanford-Clark (IBM)** and **Arlen Nipper (Arcom, now Cirrus Link)** back in 1999, where their use case was to create a protocol for **minimal battery loss** and minimal bandwidth connecting oil pipelines over satellite connections.
- They specified the following goals, which the future protocol should have:
  - Simple to implement
  - Unreliable networks
  - Provide a Quality of Service Data Delivery
  - Lightweight and Bandwidth Efficient
  - Data Agnostic
  - Continuous Session Awareness

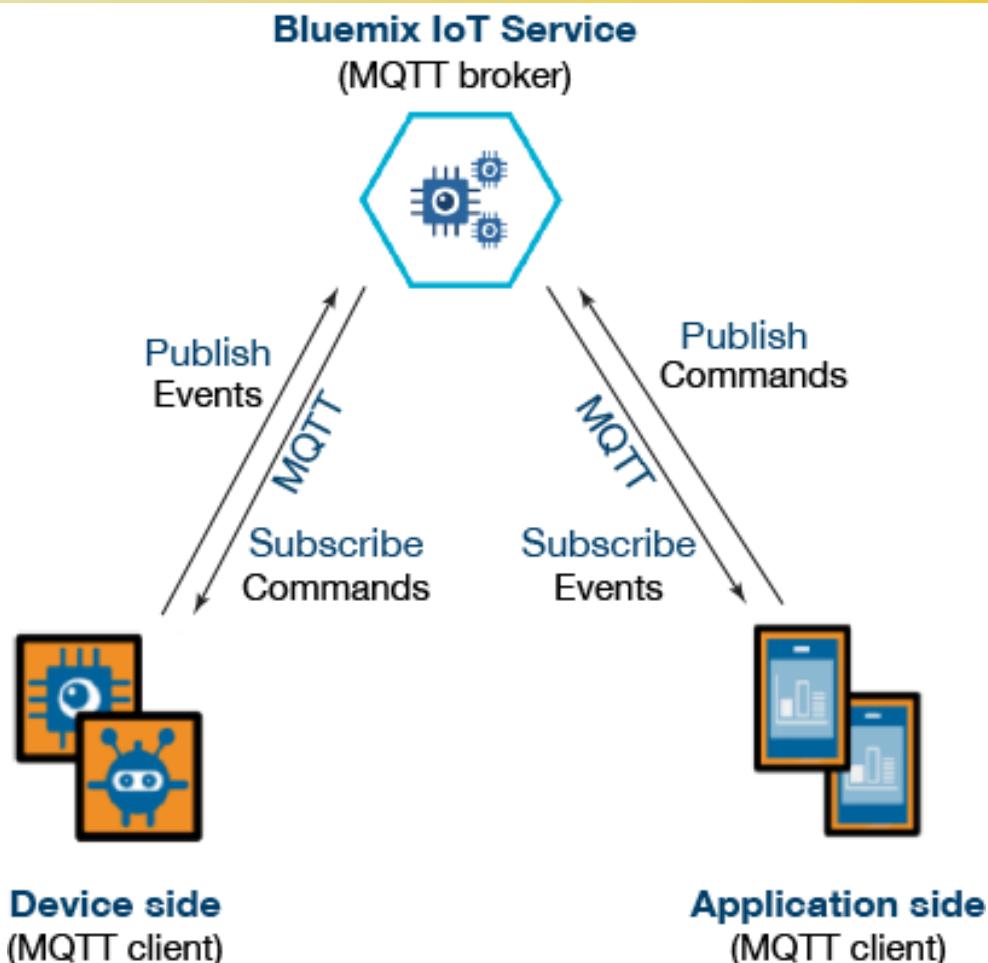


# MQTT (Message Queuing Telemetry Transport)



## Architecture

- In a nutshell, MQTT consist of three parts:
  - Broker
  - Subscribers
  - Publishers
- You can send anything as a message; up to 256 MB



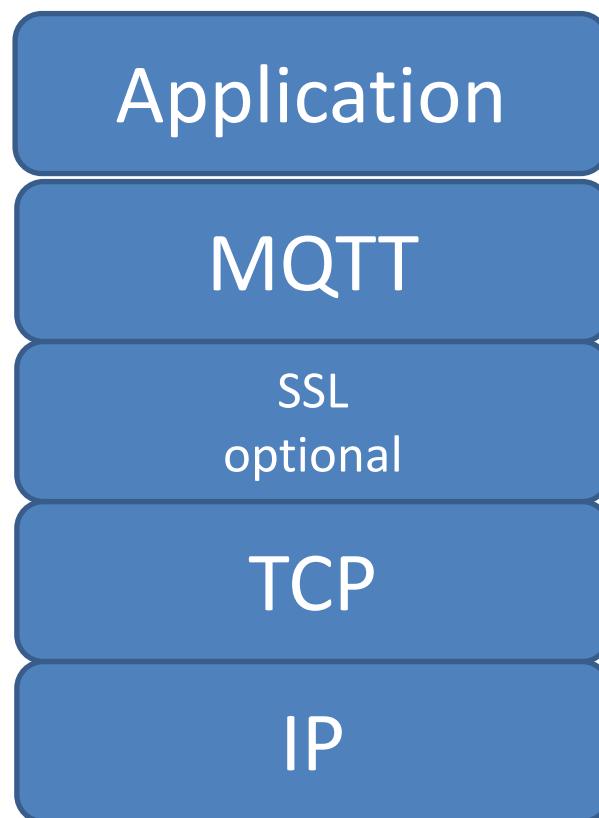
[https://www.reddit.com/r/homeautomation/comments/a1gest/diy\\_home\\_automation\\_esp32\\_raspberry\\_pi\\_node\\_red/](https://www.reddit.com/r/homeautomation/comments/a1gest/diy_home_automation_esp32_raspberry_pi_node_red/)

## Protocol Stack

TCP/IP Port: 1883

When running over SSL, TCP/IP port 8883

SSL: Secure Socket Layer (encryption)



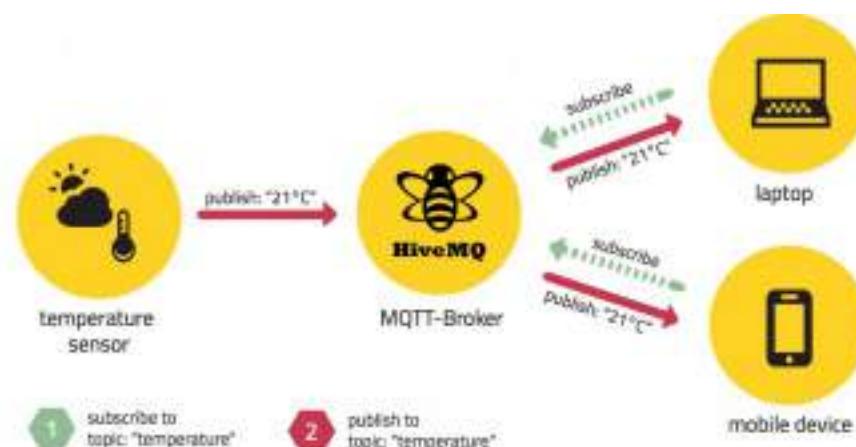
Learn more about MQTT

<https://www.youtube.com/c/HiveMQ>



## Publish/Subscribe Concept

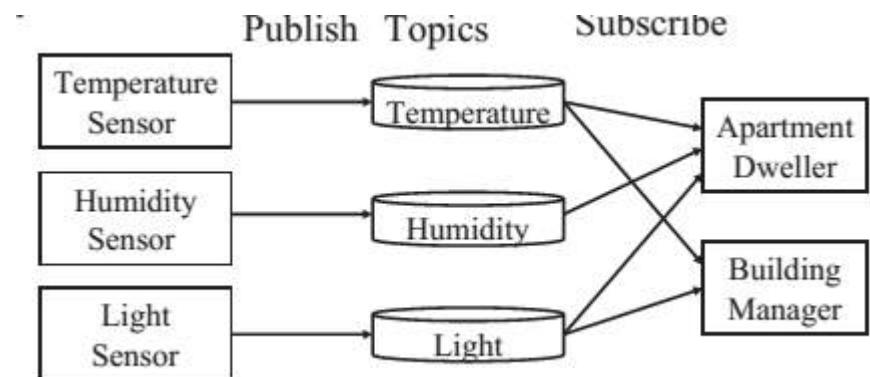
- **Decoupled in space and time:**
  - The clients do not need each others IP address and port (space) and they do not need to be running at the same time (time).
- **The broker's IP and port must be known by clients**
  - Namespace hierarchy used for topic filtering
  - It may be the case that a published message is never consumed by any subscriber
- **Broker**
  - Receives all the messages
  - Filters the messages
  - Publishes the messages to all subscribed clients





## Publish/Subscribe Concept

- **Topics/Subscriptions:**
  - Messages are published to topics.
  - Client can subscribe to a topic or a set of related topics
- **Publish/Subscribe:**
  - Clients can subscribe to topics or publish to topics



[Ref: ces570-15@www.cse.wustl.edu/]



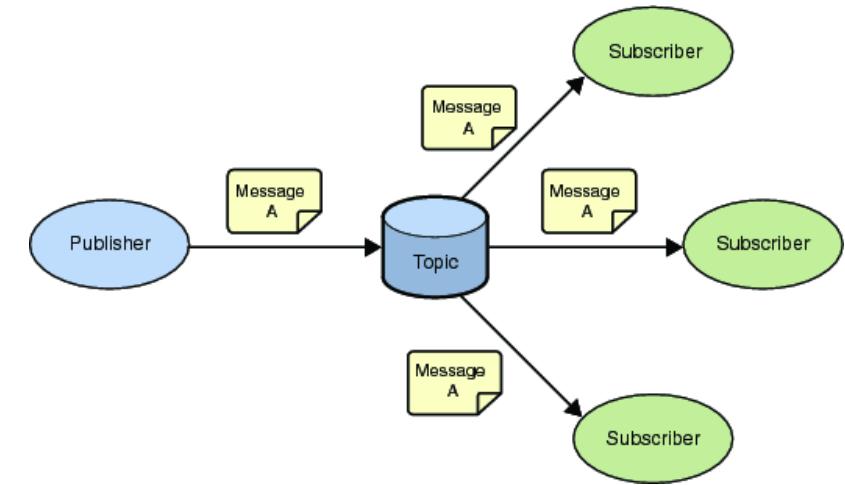
## Publish / Subscribe Messaging (One to Many)

Producer: Publishes a message (publication) on a topic (subject)

Consumer: Subscribes (makes a subscription) for messages on a topic (subject)

A message server matches publications to subscriptions:

- If none of them match the message is discarded
- If one or more matches the message is delivered to each matching consumer



Publish / Subscribe has three important characteristics:

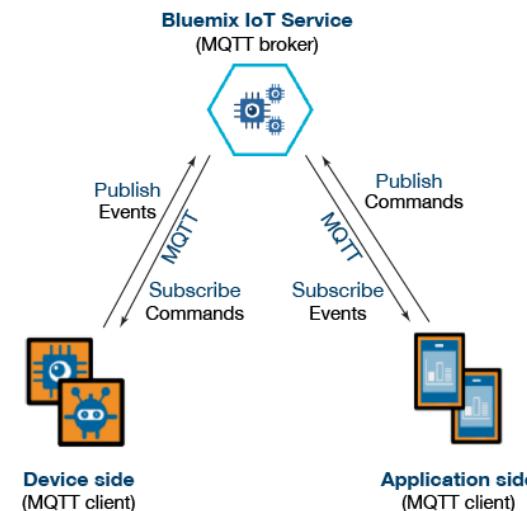
1. It decouples message senders and receivers, allowing for more flexible applications
2. It can take a single message and distribute it to many consumers
3. This collection of consumers can change over time, and vary based on the nature of the message.

# MQTT (Message Queuing Telemetry Transport)



## Example

- Clients connect to a “Broker”
- Clients subscribe to topics e.g.,
  - `client.subscribe('toggleLight/1')`
  - `client.subscribe('toggleLight/2')`
  - `client.subscribe('toggleLight/3')`
- Clients can publish messages to topics:
  - `client.publish('toggleLight/1', 'toggle');`
  - `client.publish('toggleLight/2', 'toggle');`
- All clients receive all messages published to topics they subscribe to
- **Messages can be anything**
  - Text
  - Images
  - etc.

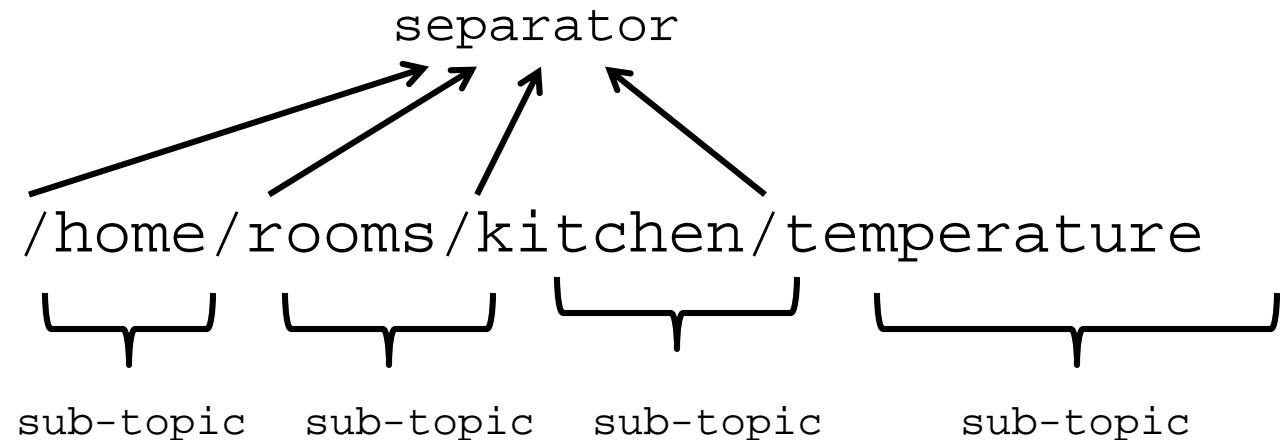


## Topics: General Rules

- Each published data specifies a topic
- Each subscriber subscribed to that topic will receive it
- Topics are case-sensitive:

home/office/lamp <> home/office/LAmp

- Topic format:



<https://www.youtube.com/watch?v=jZL4uohxuPc&t=7s>



## Topics: Wildcards

- When a client subscribes to a topic, it can subscribe to the exact topic of a published message or it can use wildcards to subscribe to multiple topics simultaneously.
- A wildcard can only be used to subscribe to topics, not to publish a message.
- There are two different kinds of wildcards: *single-level* and *multi-level*.

### (1) Single Level: +

- ✓ To replace one topic level
- ✓ Any topic matches a topic with single-level wildcard if it contains an arbitrary string instead of the wildcard.



- ✓ myhome / groundfloor / livingroom / temperature
- ✓ myhome / groundfloor / kitchen / temperature
- ✗ myhome / groundfloor / kitchen / brightness
- ✗ myhome / firstfloor / kitchen / temperature
- ✗ myhome / groundfloor / kitchen / fridge / temperature

### (2) Multi Level: #

- ✓ To cover many topic levels
- ✓ The hash symbol represents the multi-level wild card in the topic and must be placed as the last character in the topic and preceded by a forward slash.



- ✓ myhome / groundfloor / livingroom / temperature
- ✓ myhome / groundfloor / kitchen / temperature
- ✓ myhome / groundfloor / kitchen / brightness
- ✗ myhome / firstfloor / kitchen / temperature



## Topics: Beginning with \$

- Generally, you can name your MQTT topics as you wish.
- However, there is one exception: **Topics that start with a \$ symbol have a different purpose.**
- These topics **are not part** of the subscription when you subscribe to the multi-level wildcard as a topic (#).
- The \$-symbol topics are reserved for internal statistics of the MQTT broker.**
- Clients cannot publish messages to these topics.
- At the moment, there is no official standardization for such topics.
- Commonly, **\$SYS/** is used for all the following information, but broker implementations varies.
- One suggestion for \$SYS-topics is in the [MQTT GitHub wiki](#) [1]. Here are some examples:
  - \$SYS/broker/clients/connected
  - \$SYS/broker/clients/disconnected
  - \$SYS/broker/clients/total
  - \$SYS/broker/messages/sent
  - \$SYS/broker/uptime

Mosquitto

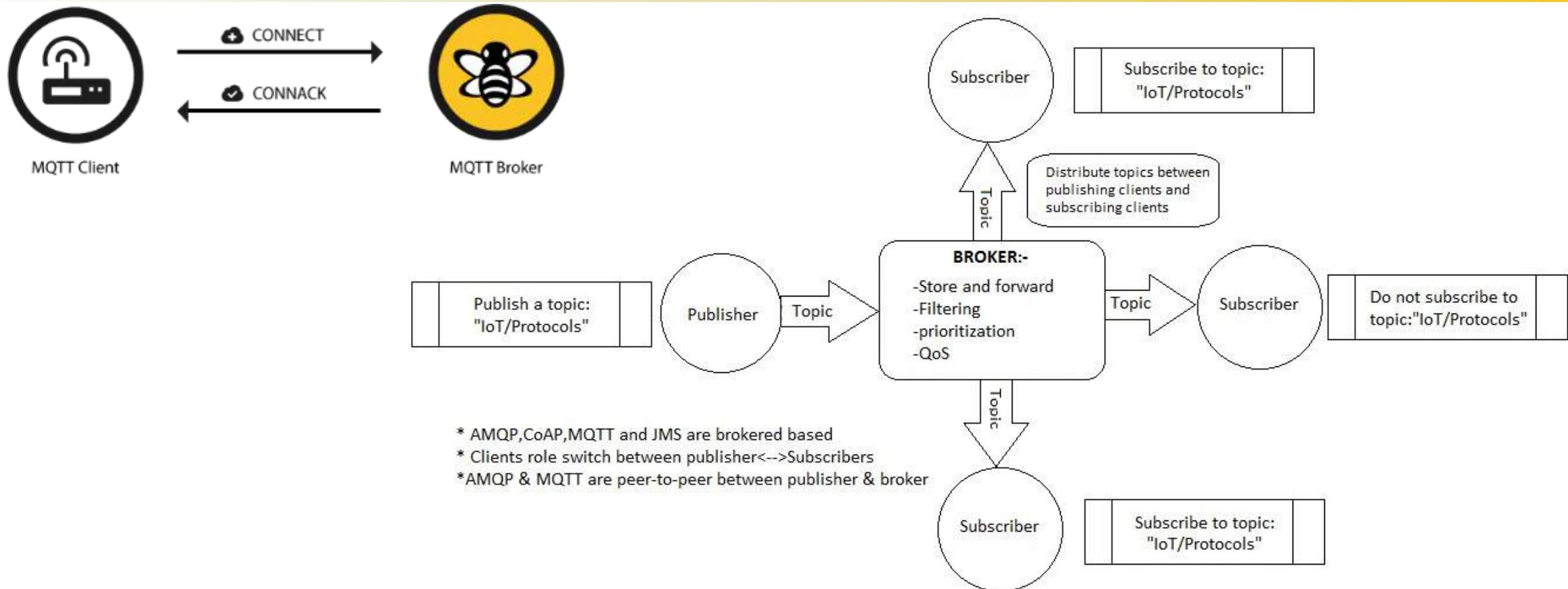
| TOPIC                             | VALUE                           |
|-----------------------------------|---------------------------------|
| \$SYS/broker/version              | mosquitto version 1.4.10        |
| \$SYS/broker/timestamp            | Fri, 22 Dec 2017 08:19:25 +0000 |
| \$SYS/broker/uptime               | 2251623 seconds                 |
| \$SYS/broker/clients/total        | 2                               |
| \$SYS/broker/clients/inactive     | 0                               |
| \$SYS/broker/clients/disconnected | 0                               |
| \$SYS/broker/clients/active       | 2                               |
| \$SYS/broker/clients/connected    | 2                               |
| \$SYS/broker/clients/expired      | 0                               |
| \$SYS/broker/clients/maximum      | 3                               |
| \$SYS/broker/messages/stored      | 61                              |
| \$SYS/broker/messages/received    | 419773                          |

[1] <https://github.com/mqtt/mqtt.org/wiki/SYS-Topics>

# MQTT (Message Queuing Telemetry Transport)



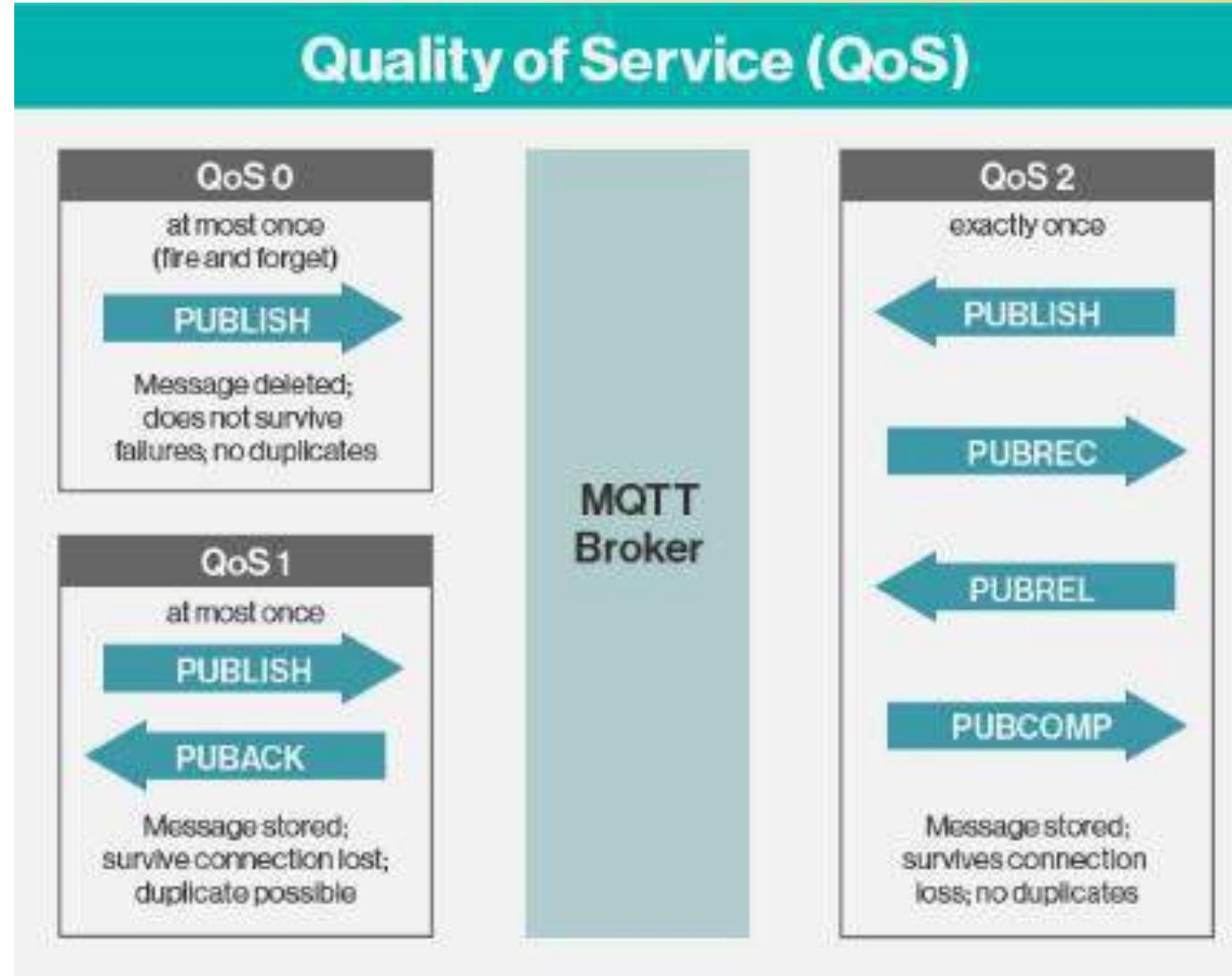
## Open Connectivity for Mobile, M2M and IoT



- \* AMQP, CoAP, MQTT and JMS are brokered based
- \* Clients role switch between publisher<-->Subscribers
- \* AMQP & MQTT are peer-to-peer between publisher & broker



## Quality of Service



- Higher QoS produces **a throughput reduction** in the order of 40%.
- Higher QoS **slows down the message transmission** (higher latency) by approximately 75%



## Quality of Service



### MQTT : Quality of Service

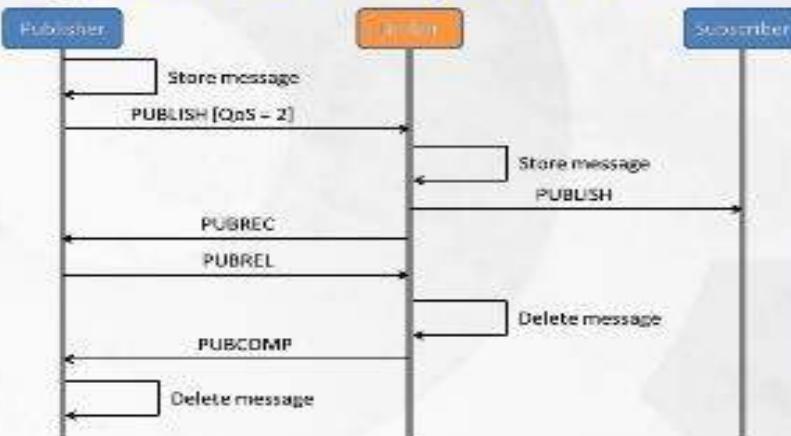
QoS 0 : At most once (fire and forget)



QoS 1 : At least once



QoS 2 : Exactly once





## Publishing “QoS” (0-Unreliability)

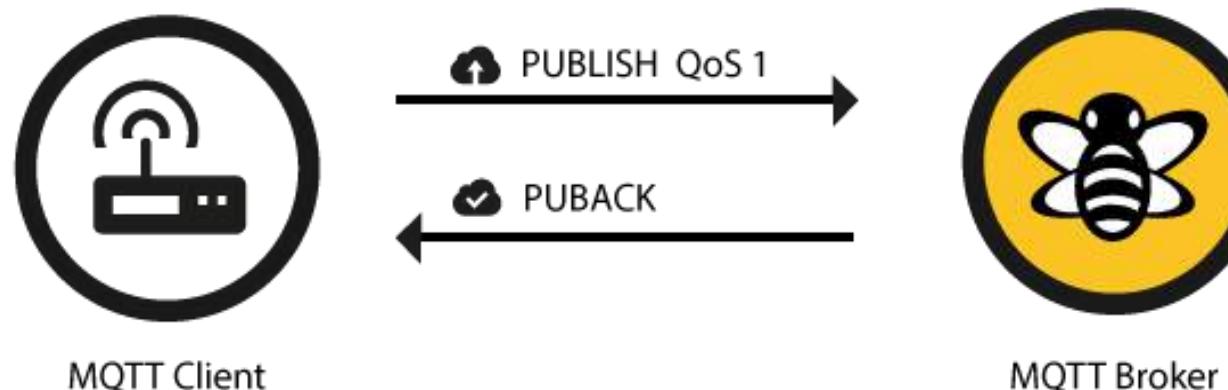
- 0 – unreliable (aka “at most once”)
  - OK for continuous streams, least overhead (1 message)
  - “Fire and forget”
  - TCP will still provide reliability
- Best-effort delivery.
- No guarantee of delivery.
- Recipient does not acknowledge receipt of the message.
- The message is not stored and retransmitted by the sender.





## Publishing “QoS” (1-Reliability)

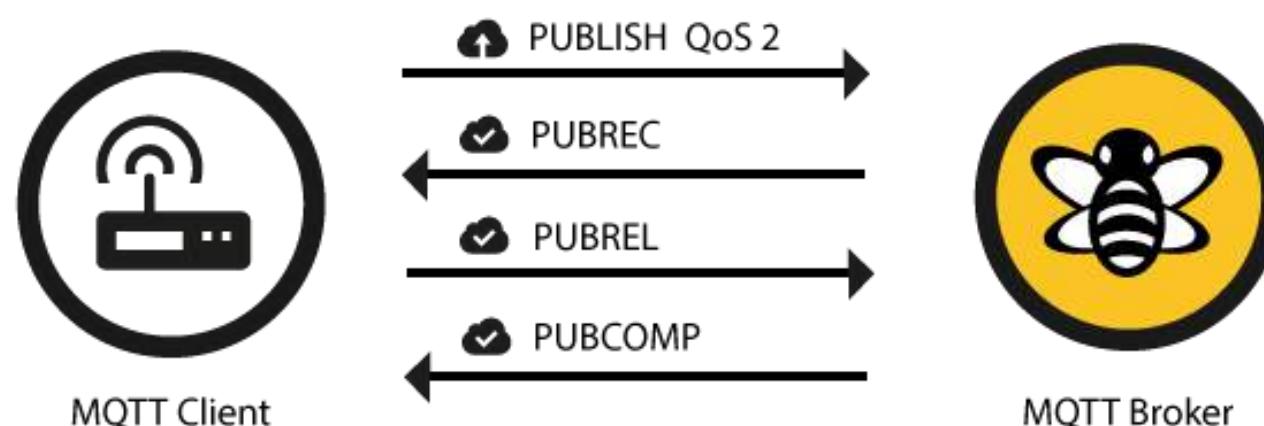
- 1 – delivery “at least once” (duplicates possible)
  - Used for alarms – more overhead (2 messages)
  - Contains message ID (to match with ACKed message)
- It guarantees that a message is delivered at least one time to the receiver.
- The sender stores the message until it gets a packet from the receiver that acknowledges receipt of the message.
- Message can be sent or delivered multiple times.





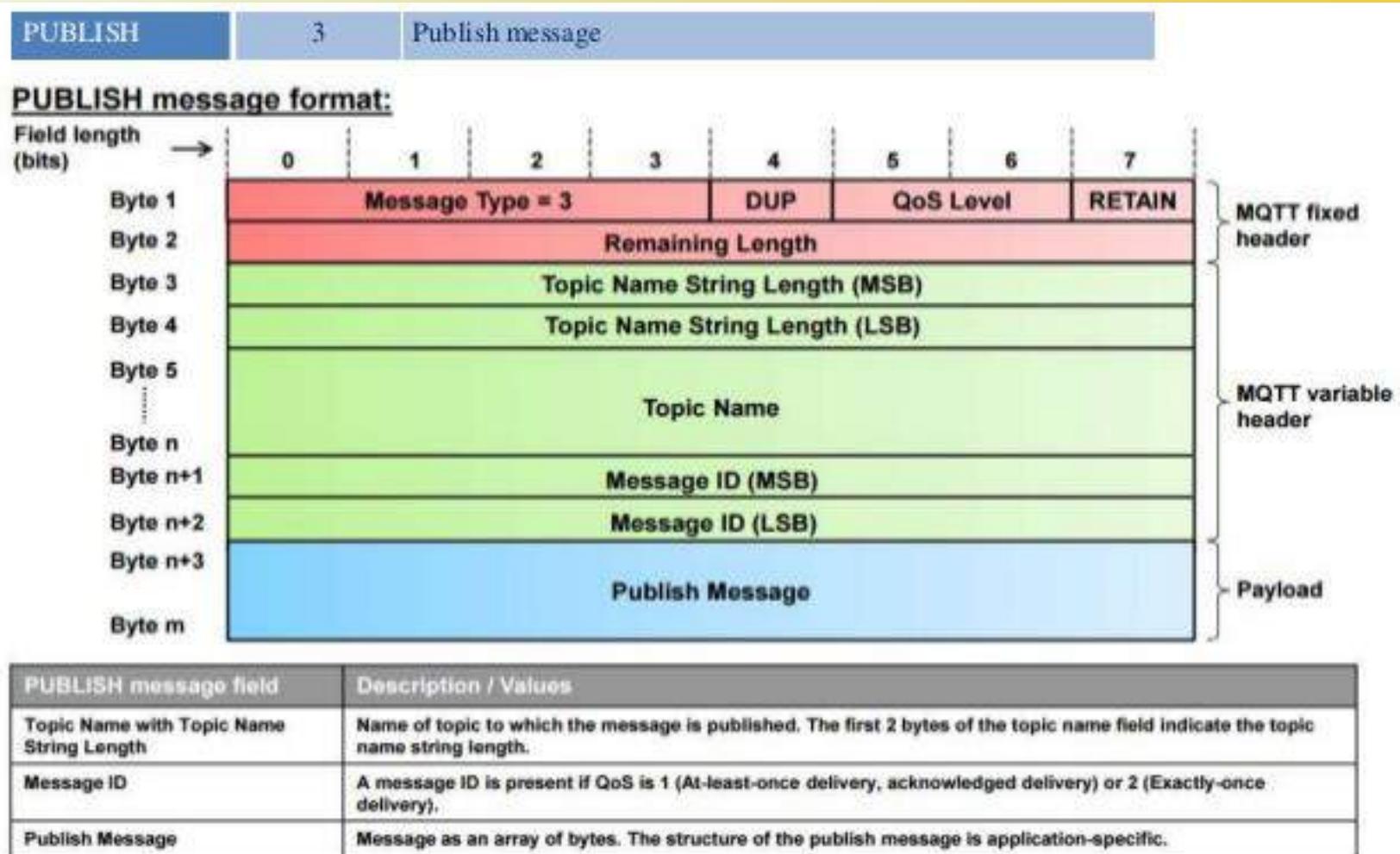
## Publishing “QoS” (2-Reliability)

- 2 – delivery “exactly once”
  - Utmost **reliability** is important – most overhead (4 messages) and slowest
- It guarantees that each message is received only once by the intended recipients.
- Safest and **slowest** QoS.
- **Guarantee is provided by at least two request/response flows (a four-part handshake)** between the sender and the receiver





## Message Format



# MQTT (Message Queuing Telemetry Transport)



## Message Types

| Name        | Value | Direction of flow                          | Description                                |
|-------------|-------|--------------------------------------------|--------------------------------------------|
| Reserved    | 0     | Forbidden                                  | Reserved                                   |
| CONNECT     | 1     | Client to Server                           | Client request to connect to Server        |
| CONNACK     | 2     | Server to Client                           | Connect acknowledgment                     |
| PUBLISH     | 3     | Client to Server<br>or                     | Publish message                            |
|             |       | Server to Client                           |                                            |
| PUBACK      | 4     | Client to Server<br>or<br>Server to Client | Publish acknowledgment                     |
| PUBREC      | 5     | Client to Server<br>or<br>Server to Client | Publish received (assured delivery part 1) |
| PUBREL      | 6     | Client to Server<br>or<br>Server to Client | Publish release (assured delivery part 2)  |
| PUBCOMP     | 7     | Client to Server<br>or<br>Server to Client | Publish complete (assured delivery part 3) |
| SUBSCRIBE   | 8     | Client to Server                           | Client subscribe request                   |
| SUBACK      | 9     | Server to Client                           | Subscribe acknowledgment                   |
| UNSUBSCRIBE | 10    | Client to Server                           | Unsubscribe request                        |
| UNSUBACK    | 11    | Server to Client                           | Unsubscribe acknowledgment                 |
| PINGREQ     | 12    | Client to Server                           | PING request                               |
| PINGRESP    | 13    | Server to Client                           | PING response                              |
| DISCONNECT  | 14    | Client to Server                           | Client is disconnecting                    |
| Reserved    | 15    | Forbidden                                  | Reserved                                   |



## Message Types

| Message fixed header field | Description / Values                                                                                                                                                                                                                                                                     |                 |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| Message Type               | 0: Reserved                                                                                                                                                                                                                                                                              | 8: SUBSCRIBE    |
|                            | 1: CONNECT                                                                                                                                                                                                                                                                               | 9: SUBACK       |
|                            | 2: CONNACK                                                                                                                                                                                                                                                                               | 10: UNSUBSCRIBE |
|                            | 3: PUBLISH                                                                                                                                                                                                                                                                               | 11: UNSUBACK    |
|                            | 4: PUBACK                                                                                                                                                                                                                                                                                | 12: PINGREQ     |
|                            | 5: PUBREC                                                                                                                                                                                                                                                                                | 13: PINGRESP    |
|                            | 6: PUBREL                                                                                                                                                                                                                                                                                | 14: DISCONNECT  |
|                            | 7: PUBCOMP                                                                                                                                                                                                                                                                               | 15: Reserved    |
| DUP                        | <p>Duplicate message flag. Indicates to the receiver that this message may have already been received.</p> <p>1: Client or server (broker) re-delivers a PUBLISH, PUBREL, SUBSCRIBE or UNSUBSCRIBE message (duplicate message).</p>                                                      |                 |
| QoS Level                  | <p>Indicates the level of delivery assurance of a PUBLISH message.</p> <p>0: At-most-once delivery, no guarantees, «Fire and Forget».</p> <p>1: At-least-once delivery, acknowledged delivery.</p> <p>2: Exactly-once delivery.</p> <p>Further details see <a href="#">MQTT QoS</a>.</p> |                 |
| RETAIN                     | <p>1: Instructs the server to retain the last received PUBLISH message and deliver it as a first message to new subscriptions.</p> <p>Further details see <a href="#">RETAIN (keep last message)</a>.</p>                                                                                |                 |
| Remaining Length           | <p>Indicates the number of remaining bytes in the message, i.e. the length of the (optional) variable length header and (optional) payload.</p> <p>Further details see <a href="#">Remaining length (RL)</a>.</p>                                                                        |                 |



## Publish Message

### MQTT Publish Message

#### Fixed Header

- DF: Duplication Flag
- QoS = 0, 1, 2
- R: Retain

#### Topic Name (Variable Header)

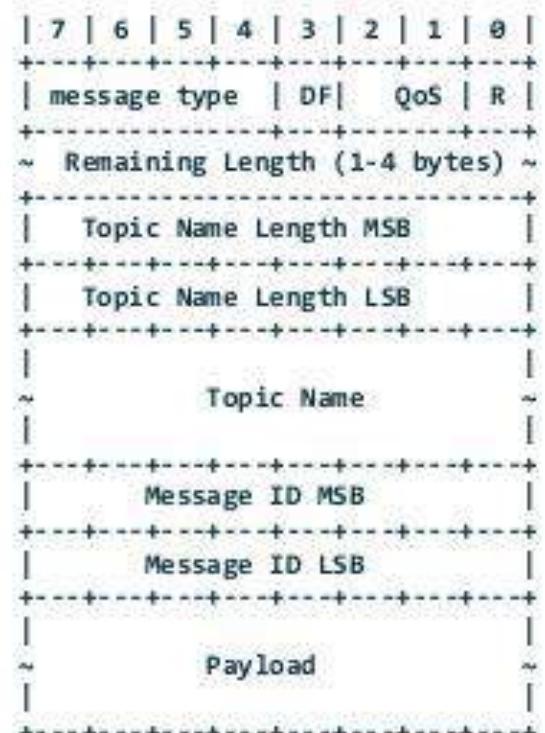
- The length depends on the string chosen by the application
- It is likely, that to avoid collisions, users will give relatively long names such as:
  - com/acme/myapp/mytopic

#### Message ID

- Only present for QoS=1,2 and used to uniquely identify messages

#### Payload

- Contains the data, but no information on the serialisation format



Original work by Matt Wilson





## Topic Name

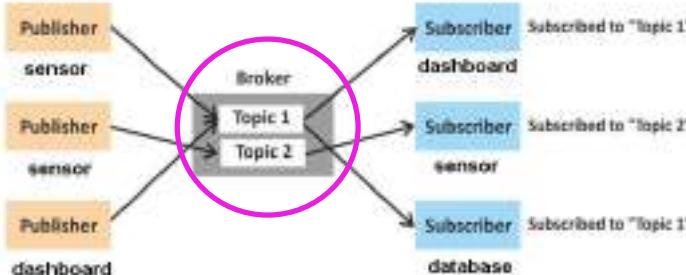
### 20 Character MQTT Topic Name

| Proto | QoS                                     | Data Message Size (bytes)             | IP (v4)<br>Headers                | Total Overhead (bytes)                                                                                                                                                                               |
|-------|-----------------------------------------|---------------------------------------|-----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MQTT  | 0                                       | (2 to 5) + 2 + 20 + length(Payload)   | IP: 20-40<br>TCP: 20-40           | <b>min</b> = 20 + 20 + 24 = <b>64</b> [TCP/IP]<br><b>max</b> = 40 + 40 + 27 = <b>107</b> [TCP/IP]                                                                                                    |
| MQTT  | 1,2                                     | (2 to 5) + 2 + 20 + 2 length(Payload) | IP: 20-40<br>TCP: 20-40           | <b>min</b> = 20 + 20 + 26 = <b>66</b> [TCP/IP]<br><b>max</b> = 40 + 40 + 29 = <b>109</b> [TCP/IP]                                                                                                    |
| DDS   | DestinationOrder = DestinationTimestamp | 44 + length(Payload)                  | IP: 20-40<br>UDP: 8<br>TCP: 20-40 | <b>min</b> = 20 + 8 + 44 = <b>72</b> [UDP/IP]<br><b>max</b> = 40 + 8 + 44 = <b>94</b> [UDP/IP]<br><b>min</b> = 20 + 20 + 44 = <b>84</b> [TCP/IP]<br><b>max</b> = 40 + 40 + 44 = <b>124</b> [TCP/IP]  |
| DDS   | DestinationOrder = SourceTimestamp      | 56 + length(Payload)                  | IP: 20-40<br>UDP: 8<br>TCP: 20-40 | <b>min</b> = 20 + 8 + 56 = <b>84</b> [UDP/IP]<br><b>max</b> = 40 + 8 + 56 = <b>106</b> [UDP/IP]<br><b>min</b> = 20 + 20 + 56 = <b>96</b> [TCP/IP]<br><b>max</b> = 40 + 40 + 56 = <b>136</b> [TCP/IP] |

Copyright © 2016-2021



## Implementation

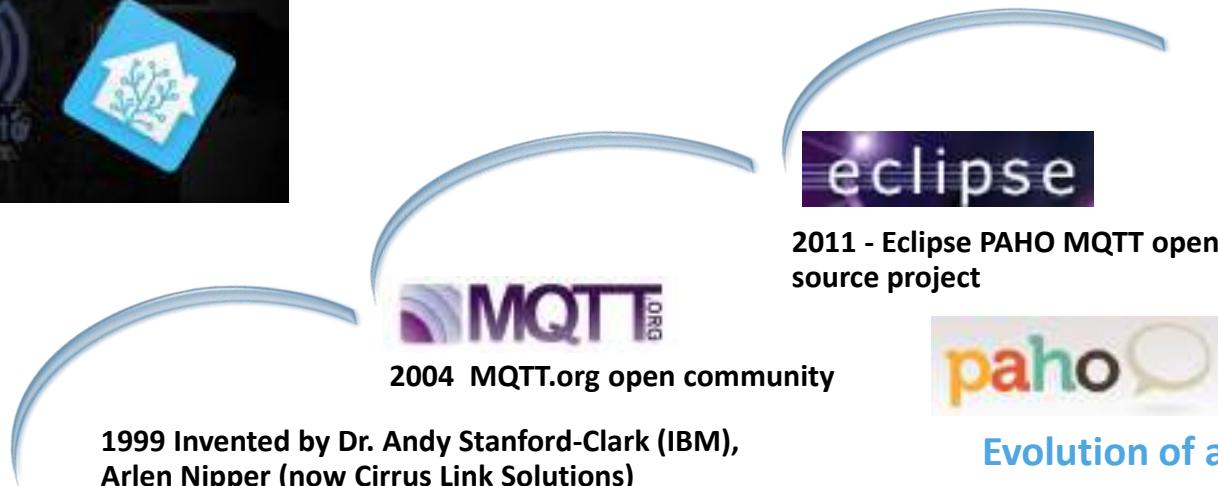


<https://mosquitto.org/>

### Mosquitto MQTT Broker Installation and Use



2011 - Eclipse PAHO MQTT open source project



2013 – MQTT Technical Committee formed

Cimetrics, Cisco, Eclipse, dc-Square, Eurotech, IBM, INETCO Landis & Gyr, LSI, Kaazing, M2Mi, Red Hat, Solace, Telit Comms, Software AG, TIBCO, WSO2



## Public MQTT Brokers

- <https://test.mosquitto.org/>
- <https://iot.eclipse.org/getting-started/#sandboxes>
- <https://www.hivemq.com/public-mqtt-broker/>
- <https://www.cloudmqtt.com/>
- <https://netpie.io/>

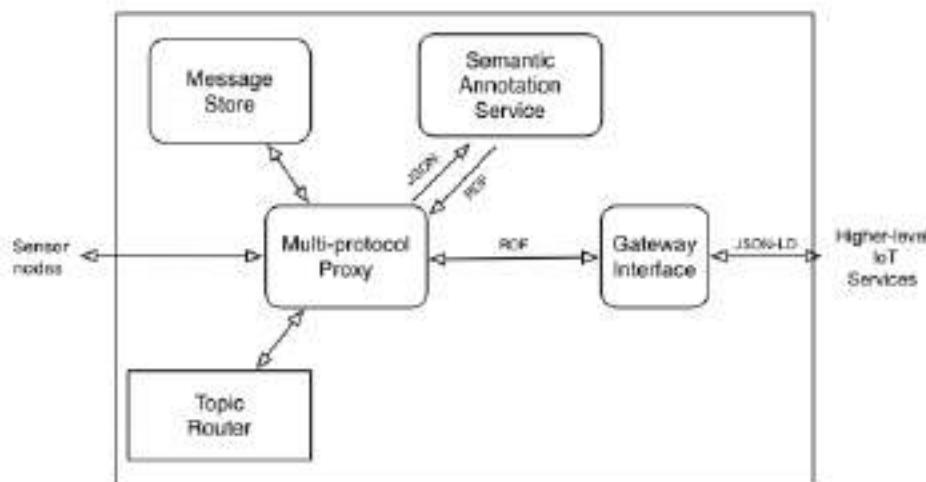
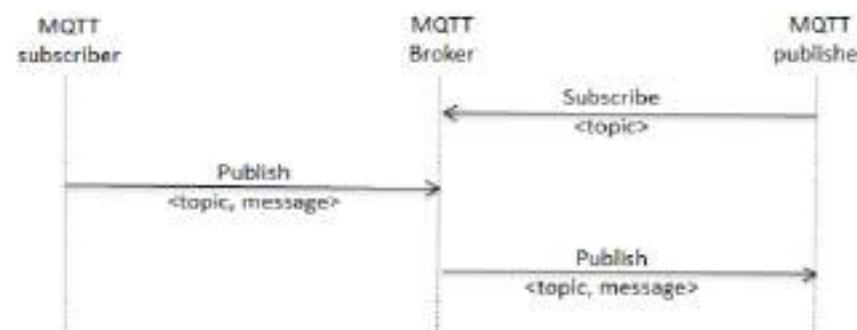
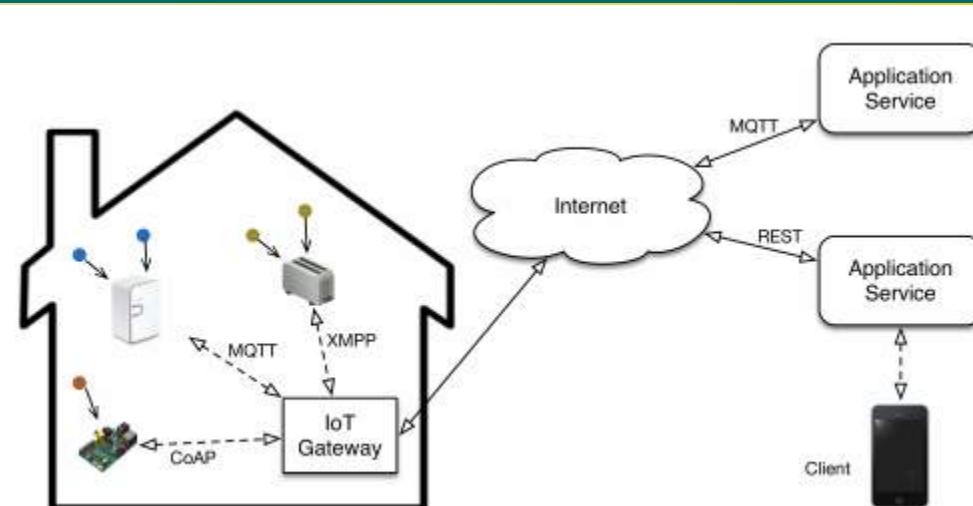
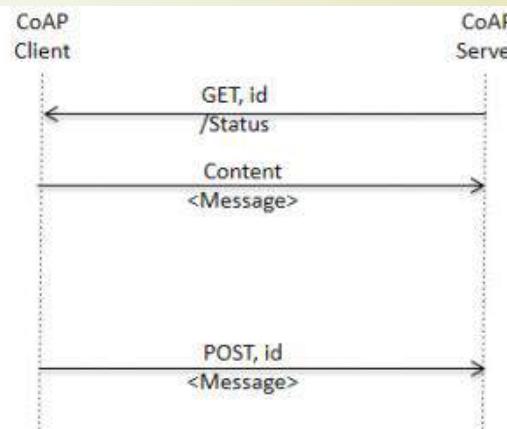
Benchmark Tool

<https://github.com/krylovsk/mqtt-benchmark>



# Implementation Examples with CoAP and MQTT

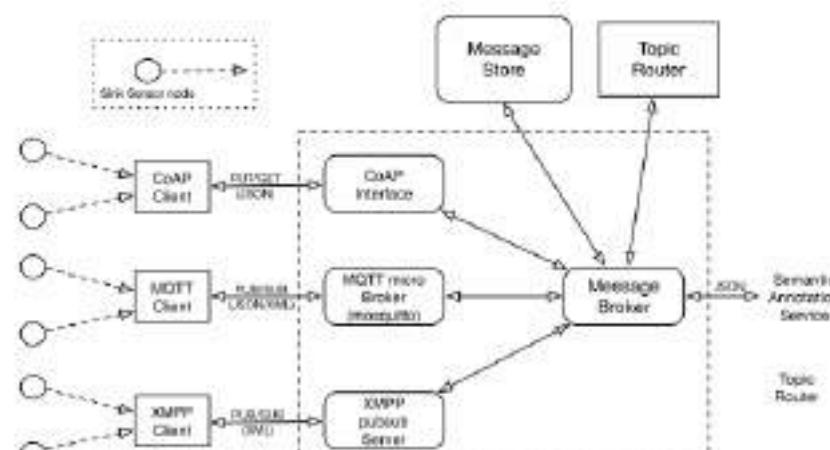
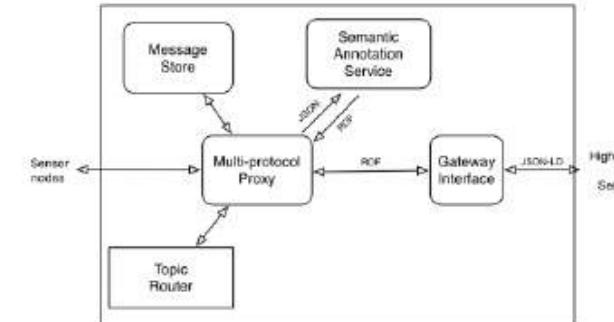
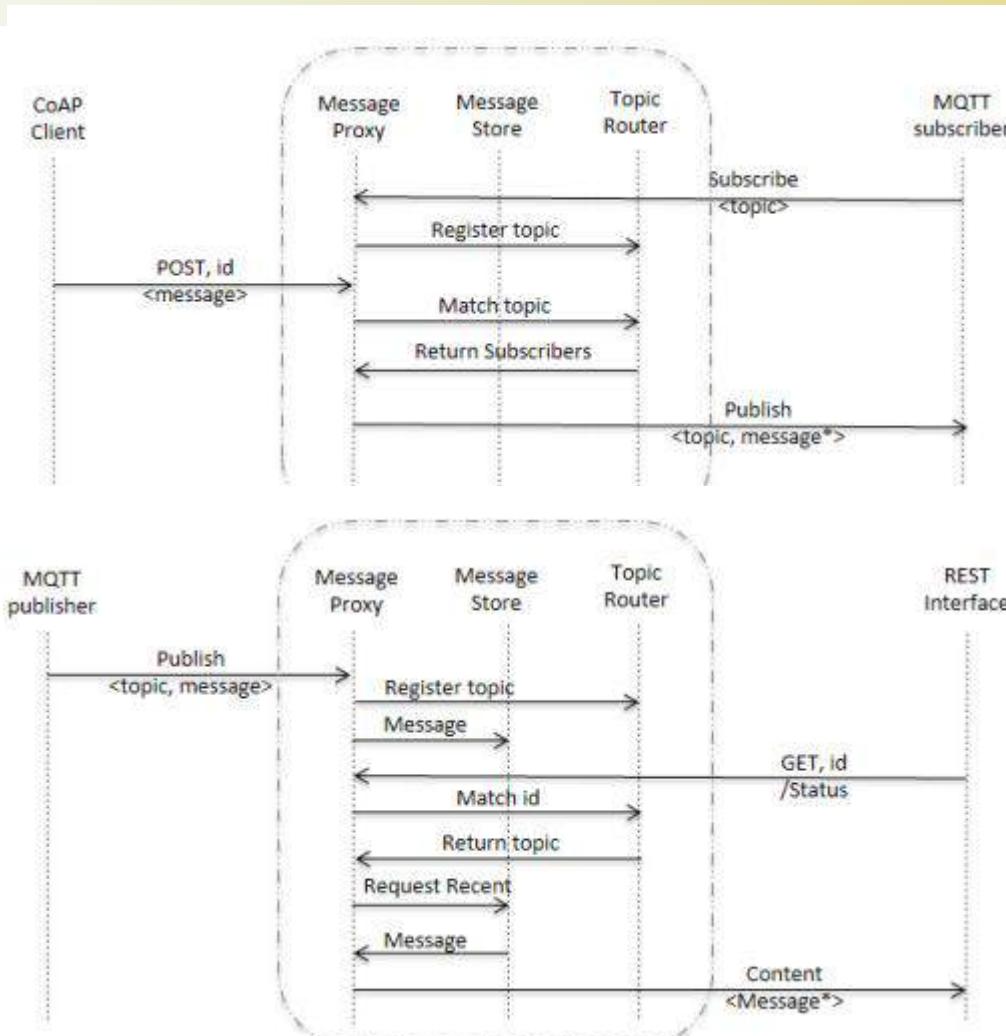
## Message Transfer (1/3)



# Implementation Examples with CoAP and MQTT



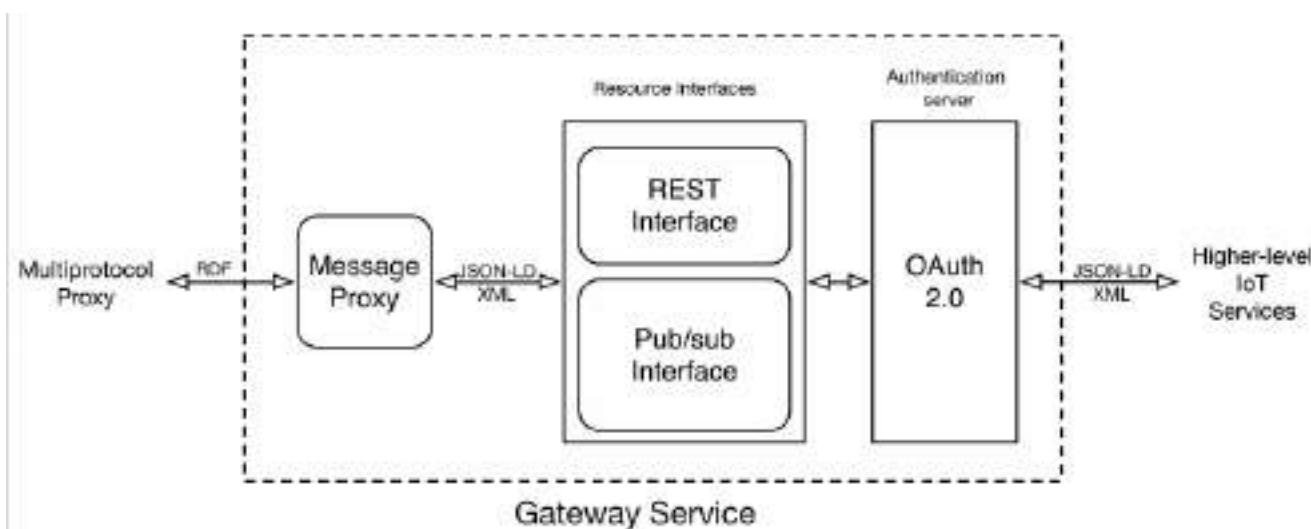
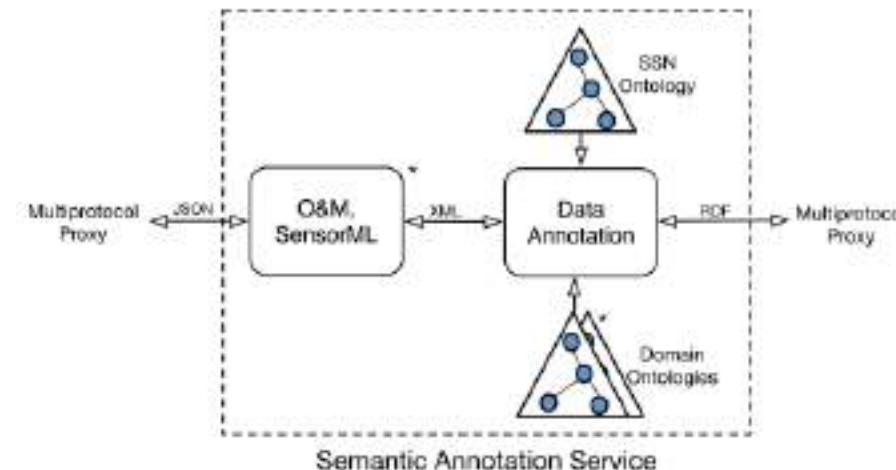
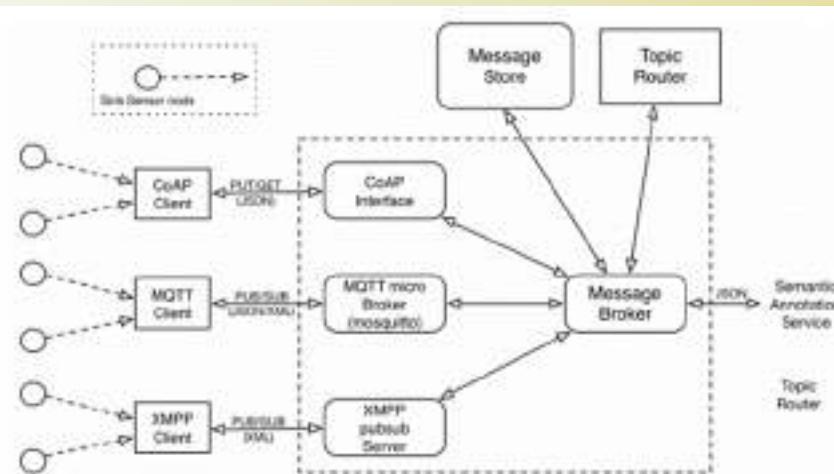
## Message Transfer (2/3)



# Implementation Examples with CoAP and MQTT



## Message Transfer (3/3)



# A Comparison between HTTP, CoAP and MQTT

| Basis of                                | HTTP (HyperText Transfer Protocol)                                                  | COAP (Constrained Application Protocol)                                        | MQTT (Message Query Telemetry Transport)                       |
|-----------------------------------------|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|----------------------------------------------------------------|
| Architecture                            | Client/Server                                                                       | Client/Server or Client/Broker                                                 | Client/Broker                                                  |
| Communication Type                      | Request-Response Model                                                              | Request-Response model.<br>One-to-one communication                            | Publish-Subscribe model.<br>Many-to-many communication         |
| Messaging Mode                          | Asynchronous                                                                        | Asynchronous/Synchronous.                                                      | Asynchronous                                                   |
| Semantics/Methods                       | Get, Post, Head, Put, Patch, Options,<br>Connect, Delete                            | Get, Post, Put, Delete                                                         | Connect, Disconnect, Publish, Subscribe,<br>Unsubscribe, Close |
| Transport layer protocol:<br>Port - QoS | TCP: <u>80/443</u><br><u>QoS Limited via TCP</u>                                    | UDP: <u>5683/5684</u><br><u>Confirmable/non-confirmable message</u>            | TCP: <u>1883/8883</u><br><u>QoS 0/1/2</u>                      |
| Header size                             | Undefined                                                                           | It has 4 bytes sized header                                                    | It has 2 bytes sized header                                    |
| Message Size                            | Large and Undefined (depends on the<br>web server or the programming<br>technology) | Small and Undefined (normally small to fit in<br>single IP datagram)           | Small and Undefined (up to 256 MB<br>maximum size)             |
| RESTful based                           |                                                                                     | Yes it uses REST principles                                                    | No it does not uses REST principles                            |
| Persistence support                     | Upon HTTP version                                                                   | It does not has such support                                                   | It supports and best used for live data<br>communication       |
| Message Labelling                       |                                                                                     | It provides by adding labels to the messages.                                  | It has no such feature.                                        |
| Usability/Security                      | TLS/SSL                                                                             | DTLS/IPSEC<br>It is used in Utility area networks and has secured<br>mechanism | TLS/SSL<br>It is used in IoT applications and is secure        |

# Key Performance Indicators



## CoAP/MQTT/AMQP/HTTP-REST/XMPP

| Key performance indicator                               | Most promising protocol |           |               |            |                       | Least promising protocol |
|---------------------------------------------------------|-------------------------|-----------|---------------|------------|-----------------------|--------------------------|
|                                                         | Over a LAN              | CoAP      | MQTT QoS 0    | AMQP       | HTTP/REST             |                          |
| Over a mobile network                                   | MQTT QoS 0              | CoAP      | WebSocket     | MQTT QoS 1 |                       |                          |
| Bandwidth consumption                                   | CoAP                    | MQTT      | AMQP and XMPP | DDS        | HTTP/REST             |                          |
| Throughput                                              | MQTT                    | DDS       | CoAP          | AMQP       | XMPP                  |                          |
| Reliability                                             | MQTT                    | AMQP      | CoAP          | HTTP/REST  |                       |                          |
| Energy consumption                                      | CoAP                    | MQTT      | AMQP          | HTTP/REST  |                       |                          |
| Developers' preference in recent IoT applications       | MQTT                    | HTTP/REST | WebSocket     | HTTP 2.0   | CoAP, AMQP, XMPP, DDS |                          |
| Researchers' preference in IoT agriculture applications | MQTT                    | HTTP/REST | CoAP          |            |                       |                          |



- [VB17] Ovidiu Vermesan and Joel Bacquet, “**Cognitive Hyperconnected Digital Transformation: Internet of Things Intelligence Evolution**”, River Publishers, 2017.
- [BHC+18] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, “**The Industrial Internet of Things (IIoT): An Analysis Framework**”, Computers in Industry 101, October 2018.
- [BS19] Rajkumar Buyya and Satish Narayana Srirama, “**Fog and Edge Computing: Principles and Paradigms**”, Wiley Series on Parallel and Distributed Computing, 2019.
- [RS19] Ammar Rayes and Samer Salam, “**Internet of Things from Hype to Reality: The Road to Digitization**”, 2<sup>nd</sup> Edition, Springer, 2019.
- [LEA20] Perry Lea, “**IoT and Edge Computing for Architects: Implementing edge and IoT systems from sensors to clouds with communication systems, analytics, and security**”, 2<sup>nd</sup> Edition, Packt Publishing, 2020.

AMQP 1.0: <https://www.youtube.com/watch?v=ODpeldUdClc&list=PLmE4bZU0qx-wAP02i0I7PJWvDWoCytEjD&index=2>  
<https://www.youtube.com/watch?v=g3e9lDlMn5M>

XMPP: <https://www.youtube.com/watch?v=92egt5-UDwo>  
<https://www.youtube.com/watch?v=fz0yDNwEydU>

DDS: <https://www.youtube.com/watch?v=7IV49wKxs4c> (Security Issues)  
<https://www.youtube.com/watch?v=k1P1cQUZI-A> (Demo)  
<https://www.youtube.com/watch?v=u-saogMmKOo>



# **ITCS447**

## **Lecture 9**

# **ESP32 Web Server**

**Asst. Prof. Dr. Thitinan Tantidham**

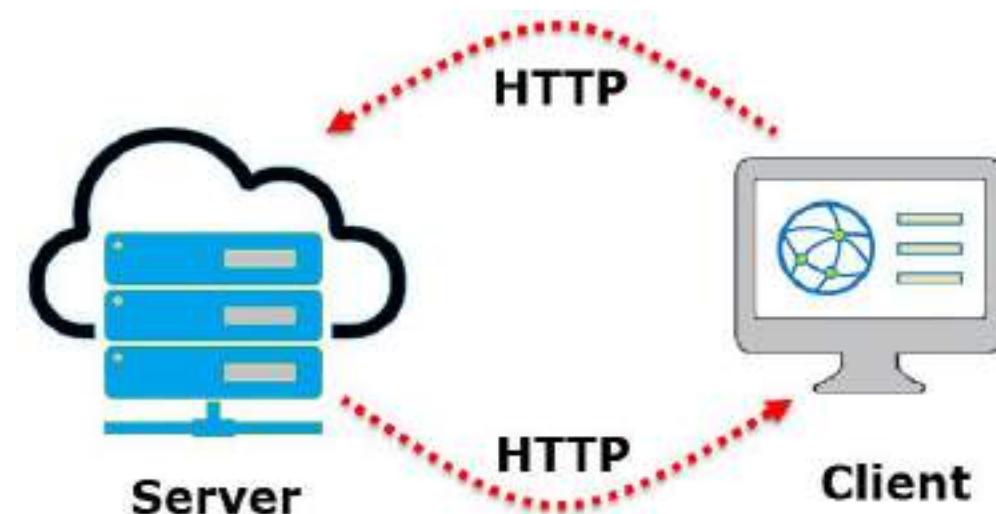


ห้ามมิให้นักศึกษาทำข้า ดัดแปลง หรือใช้ประโยชน์จากการอันมีลิขสิทธิ์ปราการอยู่ในระบบการศึกษาอิเล็กทรอนิกส์ (E-Learning) ของมหาวิทยาลัย ไม่ว่าจะทั้งหมดหรือบางส่วน โดยไม่ได้รับอนุญาตจากมหาวิทยาลัย นอกจากนี้จากการศึกษาส่วนบุคคล ทั้งนี้ การทำข้า ดัดแปลง หรือเผยแพร่ต่อสาธารณะชื่องานอันมีลิขสิทธิ์ จะมีโทษปรับตั้งแต่ 20,000 บาท ถึง 200,000 บาท และหากเป็นการกระทำเพื่อการค้า จะมีโทษจำคุกตั้งแต่ 6 เดือน ถึง 10 ปี หรือปรับตั้งแต่ 100,000 บาท ถึง 800,000 บาท หรือทั้งจำทั้งปรับ

Unless allowed by Mahidol University, the User shall not copy, modify, or exploit in part or in whole of the copyrighted materials on the Platform, other than for your own individual study. The copy, modify or communication to public of the copyrighted materials shall be inflicted with a fine from 20,000 Baht up to 200,000 Baht. If the offence is committed with the commercial purpose, the offender shall be inflicted with imprisonment for a term from 6 months up to 10 years or a fine from 100,000 Baht up to 800,000 Baht or both imprisonment and fine.

- ESP32 Mode
- ESP32 Webserver
- ESP32 REST API
- Serial Peripheral Interface Flash File System (SPIFFS)

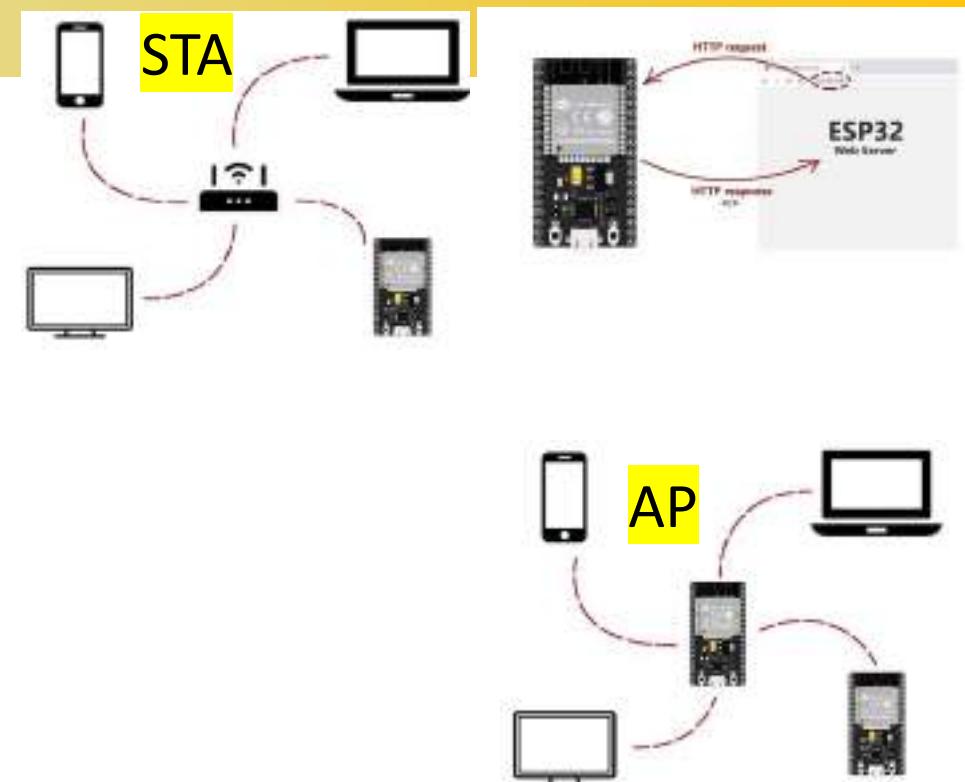
- HTTP 1.1/2.0/3.0
- HTTP status code
  - When a request from a web browser is sent to a web server, the server returns a code as a response, for examples:
    - 200: means the connection is established correctly,
    - 404: indicates that the address is not correct).
- Web Page: HTML, XML, CSS, JS ...



# ESP32 Mode



- ESP32 as a Station Mode (STA)
  - ESP32 is connected to the Wi-Fi router as a Client and can access the Internet through the router.
- ESP32 as an Access Point Mode (AP)
  - ESP32 can act as a router and creates a local wifi network with the desired name and password.
  - The number of devices is limited. It's also called Soft Access Point.
- Both modes can be programmed as a Webserver
  - HTTP/HTTPs
  - TCP Port 80/443



<https://www.hackster.io/electropeak/create-a-web-server-w-esp32-tutorial-a9a392>

# ESP32 vs ESP8266 Web Server



```
#include <Arduino.h>
#ifndef ESP32
    #include <WiFi.h>
    #include <AsyncTCP.h>
#else
    #include <ESP8266WiFi.h>
    #include <ESPAsyncTCP.h>
#endif
#include <ESPAsyncWebServer.h>
```

```
AsyncWebServer server(80);

#include "config.h"
```

```
String HTML = "<!DOCTYPE html> \
<html> \
<body> \
<h1>My First Web Server with ESP32 - \
Station Mode &#128522;</h1> \
</body> \
</html>" ;

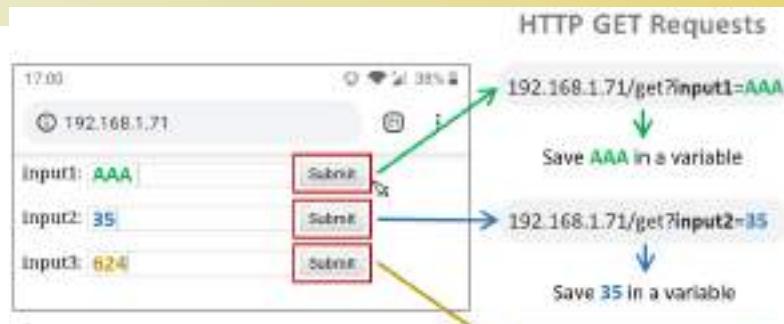
server.send(200, "text/html", HTML);
```

```
client.println("<!DOCTYPE html><html>");
client.println("<head><meta name=\"viewport\" \
content=\"width=device-width, initial-scale=1\"> ");
client.println("<link rel=\"icon\" href=\"data:, \">");
```

# ESP32 Web Server [RANDOMNERDTUTORIALS]



## Example: Input Data



|                                  |
|----------------------------------|
| Web Servers                      |
| Output Web Server                |
| PWM Slider Web Server            |
| PWM Multiple Sliders Web Server  |
| Async Web Server                 |
| Relay Web Server                 |
| Servo Web Server                 |
| DHT Web Server                   |
| BME280 Web Server                |
| BME680 Web Server                |
| DS18B20 Web Server               |
| LoRa Web Server                  |
| Plot/Chart Web Server            |
| Chart Multiple Series Web Server |
| SPIFFS Web Server                |
| Thermostat Web Server            |
| Momentary Switch Web Server      |
| Physical Button Web Server       |
| Input Fields Web Server          |
| Images Web Server                |
| RGB LED Web Server               |
| Timer/Pulse Web Server           |
| HTTP Auth Web Server             |
| MPU-6050 Web Server              |
| MicroSD Card Web Server          |
| Stepper Motor Web Server         |
| Stepper Motor WebSocket          |
| Gauges Web Server                |

<https://randomnerdtutorials.com/esp32-esp8266-input-data-html-form/>

# ESP32 Web Server (include other HTML Libraries)



## Example: Page Creation and Content Updating

The screenshot shows a web browser window titled "Sensor Monitor". The address bar indicates the URL is "Not reached | 192.168.1.66". The page header includes the title "Sensor Monitor" and the date/time "10/23/2021 11:48:42 PM".

**Sensor Readings:**

| Pin            | Bits          | Volts |
|----------------|---------------|-------|
| Analog pin 34  | 0             | 0.0   |
| Analog pin 35  | 1             | 0.0   |
| Digital switch | Switch is OFF |       |

**Sensor Controls:**

LED: Turn ON

Switch: Turn ON

Fan Speed Control (RPM: 1995)

ESP32 Web Page Creation and Data Update Demo

<https://www.youtube.com/watch?v=pL3dhGtmcMY>

[https://github.com/KrisKasprzak/ESP32\\_WebPage](https://github.com/KrisKasprzak/ESP32_WebPage)



- SPIFFS is a lightweight filesystem created for microcontrollers with a flash chip, which is connected by SPI bus.
- ESP32 contains a flash memory with SPIFFS.
- Setup and Test:
  - <https://randomnerdtutorials.com/install-esp32-filesystem-uploader-arduino-ide/>
- ESP32 Webserver using SPIFFS
  - <https://randomnerdtutorials.com/esp32-web-server-spiffs-spi-flash-file-system/>

<https://randomnerdtutorials.com/install-esp32-filesystem-uploader-arduino-ide/>



- [VB17] Ovidiu Vermesan and Joel Bacquet, "**Cognitive Hyperconnected Digital Transformation: Internet of Things Intelligence Evolution**", River Publishers, 2017.
- [BHC+18] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "**The Industrial Internet of Things (IIoT): An Analysis Framework**", Computers in Industry 101, October 2018.
- [BS19] Rajkumar Buyya and Satish Narayana Srirama, "**Fog and Edge Computing: Principles and Paradigms**", Wiley Series on Parallel and Distributed Computing, 2019.
- [RS19] Ammar Rayes and Samer Salam, "**Internet of Things from Hype to Reality: The Road to Digitization**", 2<sup>nd</sup> Edition, Springer, 2019.
- [LEA20] Perry Lea, "**IoT and Edge Computing for Architects: Implementing edge and IoT systems from sensors to clouds with communication systems, analytics, and security**", 2<sup>nd</sup> Edition, Packt Publishing, 2020.

HTTP Status Code:

[https://en.wikipedia.org/wiki/List\\_of\\_HTTP\\_status\\_codes](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes)

<https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>

<https://www.electronicshub.org/esp32-web-server/>

<https://www.dfrobot.com/blog-1574.html>

<https://github.com/lorol/LITTLEFS>

<https://techtutorialsx.com/2019/04/07/esp32-https-web-server/>

[https://www.tutorialspoint.com/esp32\\_for\\_iot/esp32\\_for\\_iot\\_transmitting\\_data\\_over\\_wifi\\_using\\_https.htm](https://www.tutorialspoint.com/esp32_for_iot/esp32_for_iot_transmitting_data_over_wifi_using_https.htm)



# ITCS447

## Lecture 10

### IoT Dashboards

Cloud (eg. NETPIE, AWS, IBM Cloud)  
Opensource (eg. Thingsboard, Node-RED)  
Smartphone (eg.Blynk)

### LINE Notification

Asst. Prof. Dr. Thitinan Tantidham



ห้ามมิให้นักศึกษาทำข้า ดัดแปลง หรือใช้ประโยชน์จากการอันมีลิขสิทธิ์ปราภรอยู่ในระบบการศึกษาอิเล็กทรอนิกส์ (E-Learning) ของมหาวิทยาลัย ไม่ว่าจะทั้งหมดหรือบางส่วน โดยไม่ได้รับอนุญาตจากมหาวิทยาลัย นอกจากนี้จากการศึกษาส่วนบุคคล ทั้งนี้ การทำข้า ดัดแปลง หรือเผยแพร่ต่อสาธารณะชื่องานอันมีลิขสิทธิ์ จะมีโทษปรับตั้งแต่ 20,000 บาท ถึง 200,000 บาท และหากเป็นการกระทำเพื่อการค้า จะมีโทษจำคุกตั้งแต่ 6 เดือน ถึง 10 ปี หรือปรับตั้งแต่ 100,000 บาท ถึง 800,000 บาท หรือทั้งจำทั้งปรับ

Unless allowed by Mahidol University, the User shall not copy, modify, or exploit in part or in whole of the copyrighted materials on the Platform, other than for your own individual study. The copy, modify or communication to public of the copyrighted materials shall be inflicted with a fine from 20,000 Baht up to 200,000 Baht. If the offence is committed with the commercial purpose, the offender shall be inflicted with imprisonment for a term from 6 months up to 10 years or a fine from 100,000 Baht up to 800,000 Baht or both imprisonment and fine.

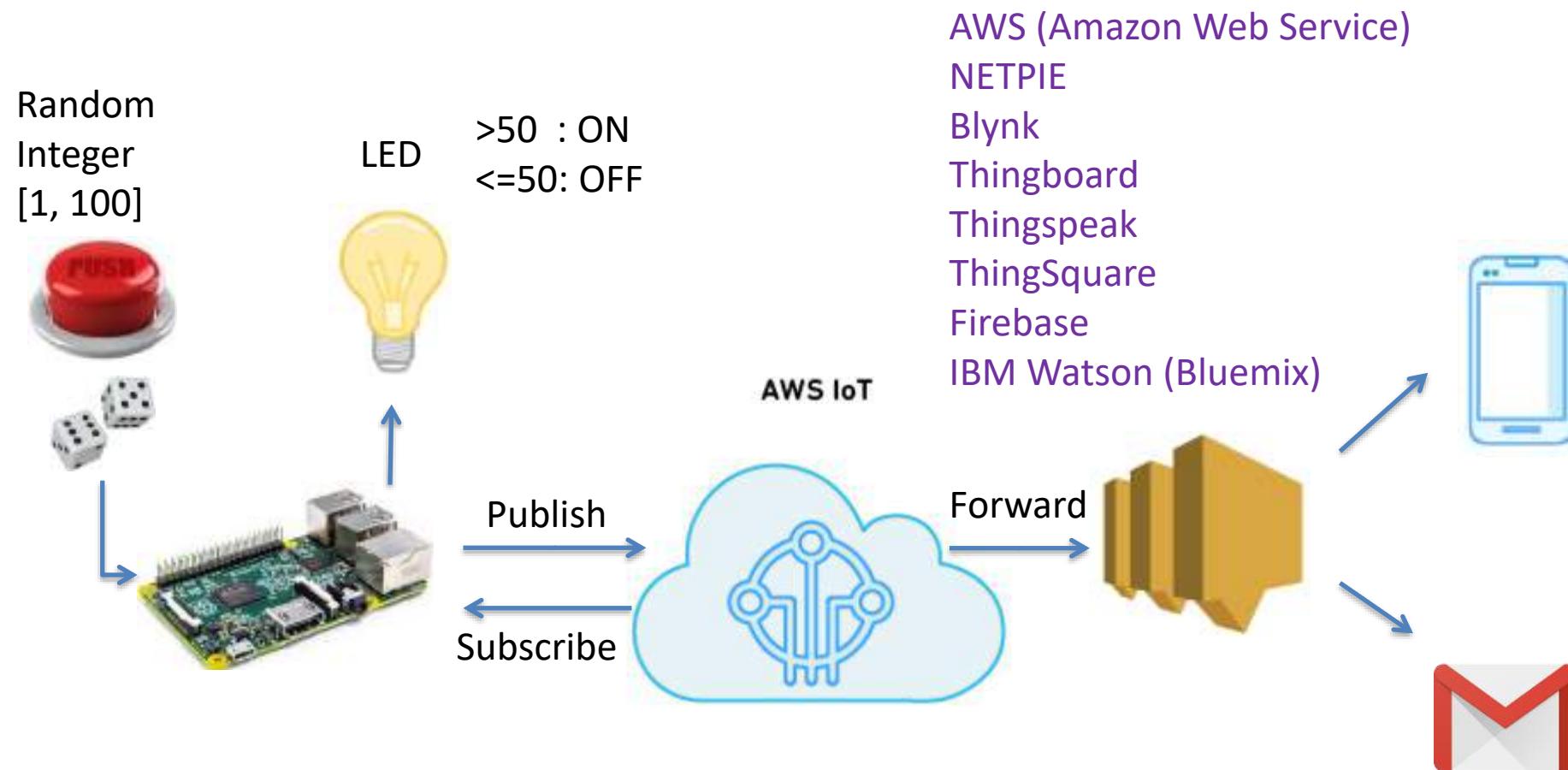
- Components in IoT Platform
- IoT for Monitoring and Control
- IoT for Proactive Protection
- IoT Data Visualization (IoT Dashboards)
- IoT Dashboards on Edge – NODE-RED
- IoT Dashboard on Cloud – NETPIE
- IoT Dashboard on Smartphone – Blynk
- Data Notification - LINE Notify

# Components in IoT Platform

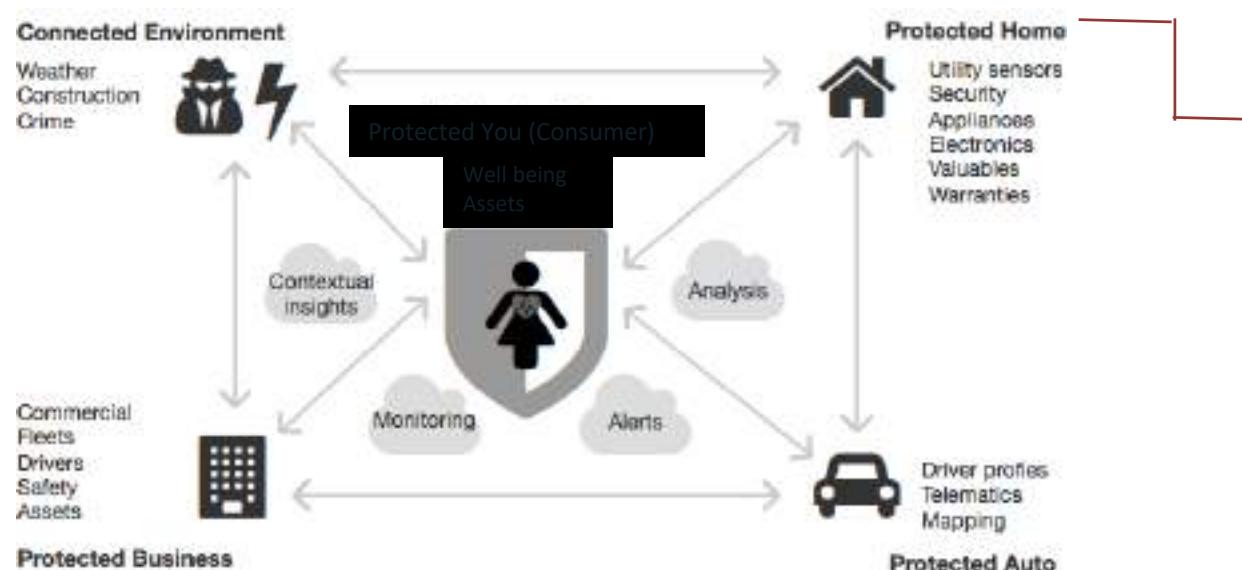


|                                                                           |                                                                                                                                                 |                                                                                                                    |
|---------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|
| <p><b>Database</b><br/>Repository that stores the important data sets</p> | <p><b>External interfaces</b><br/>APIs, SDKs and gateways that act as interfaces for 3rd party systems (e.g., ERP, CRM)</p>                     |                                                                                                                    |
|                                                                           | <p><b>Analytics</b><br/>Algorithms for advanced calculations and machine learning</p>                                                           | <p><b>Additional tools</b><br/>Further development tools (e.g., app prototyping, access management, reporting)</p> |
|                                                                           | <p><b>Data visualization</b><br/>Graphical depiction of (real-time) sensor data</p>                                                             | (Dashboards)                                                                                                       |
|                                                                           | <p><b>Processing &amp; action management</b><br/>Rule engine that allows for (real-time) actions based on incoming sensor &amp; device data</p> |                                                                                                                    |
|                                                                           | <p><b>Device management</b><br/>Backend tool for the management of device status, remote software deployment and updates</p>                    |                                                                                                                    |
|                                                                           | <p><b>Connectivity &amp; Normalization</b><br/>Agents and libraries that ensure constant object connectivity and harmonized data formats</p>    |                                                                                                                    |

# IoT for Monitoring and Control



# IoT for Proactive Protection



## Water Leak Detection

- Reduced claims
- Real time notifications
- Personalized risk assessment
- Satisfied customers

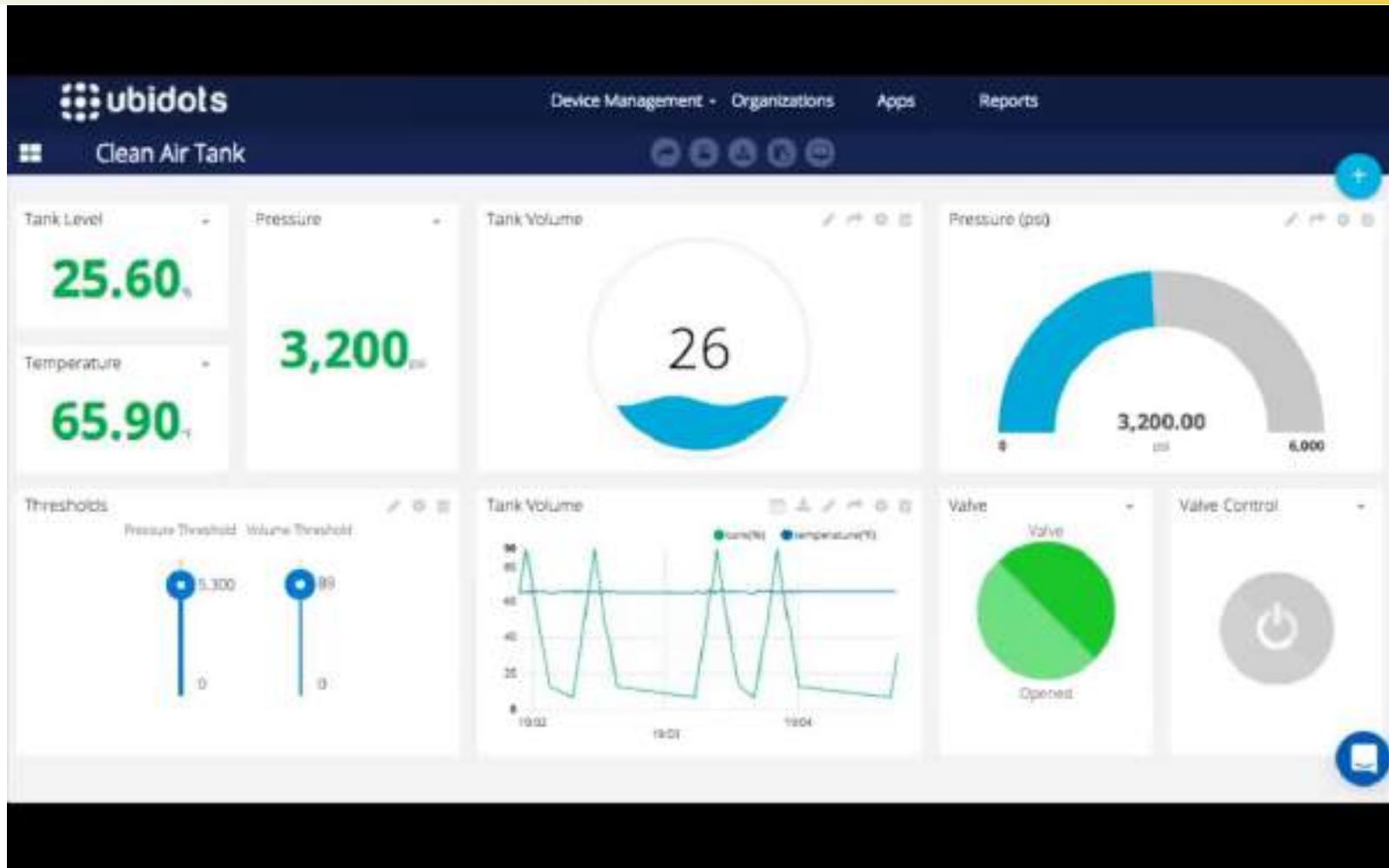
# IoT Data Visualization (IoT Dashboards)



- For the IoT, or any control system, IoT dashboard is the **key HMI** (Human-Machine Interface) component that organizes and presents digital information from our physical world into a simply **understood display** on a computer or mobile device.
- With the help of IoT Dashboards, **users and operators can (remotely) monitor and control specific assets and processes**, and depending on safety requirements, access and control an environment from anywhere in the world.
- IoT dashboards populated with graphs, charts, control switches, maps, tables, and countless other widgets are the digital tools we use to visualize and display data coming from the physical world to our computers.



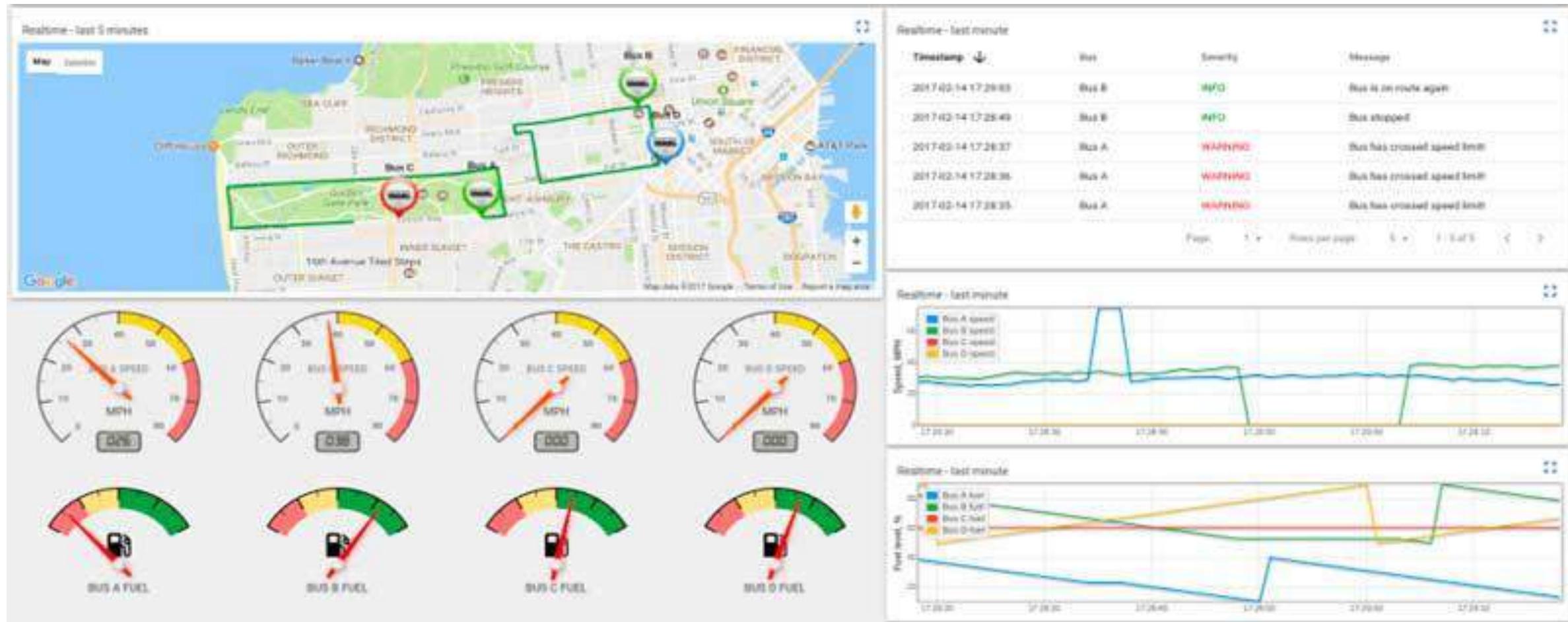
## Examples: ubidots (Monitoring and Control)



# IoT Dashboards

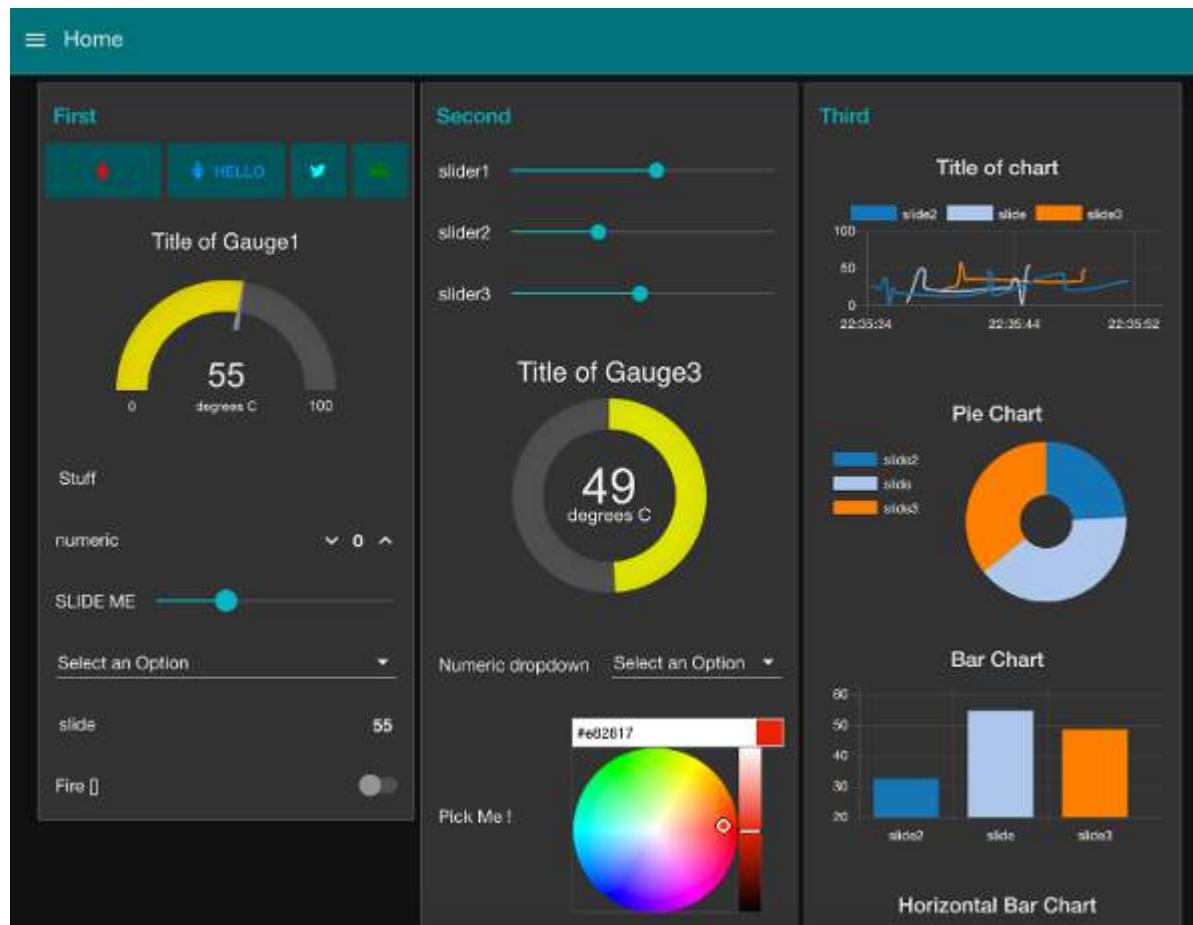


Examples: with Map (Regions) e.g. Thingspeak



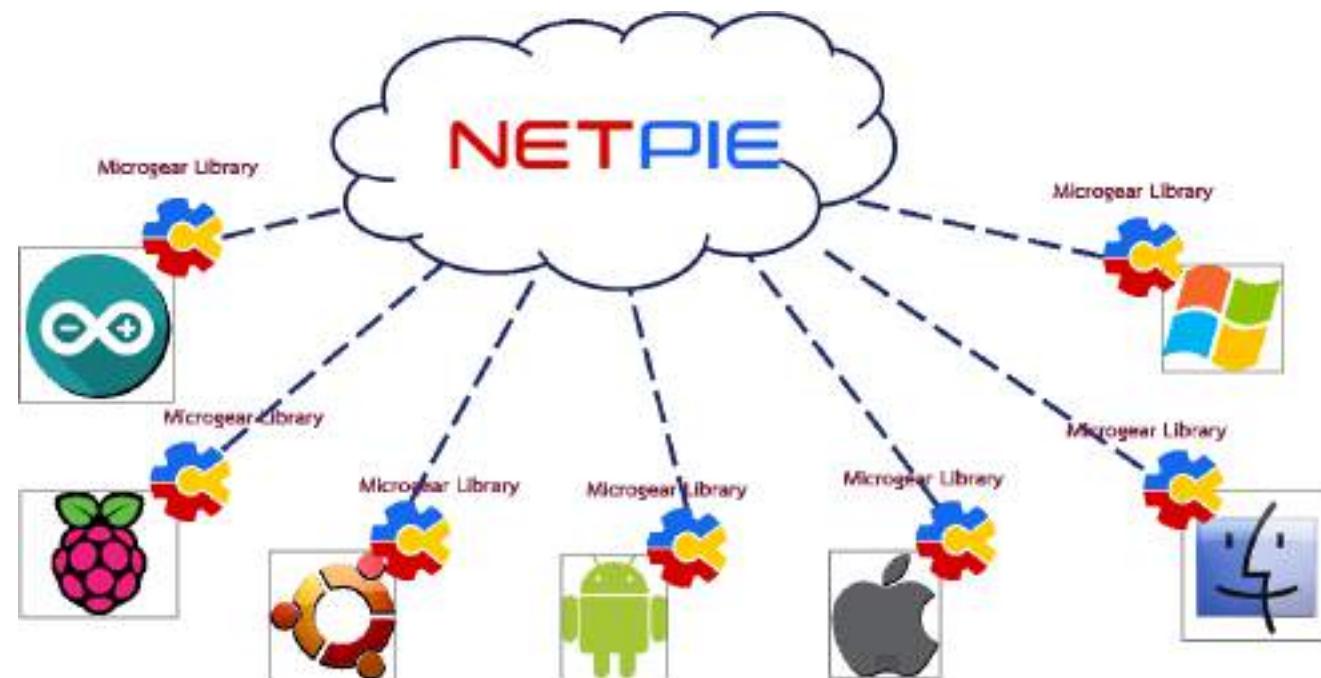


<https://nodered.org/>



<http://netpie.io>

Ref: NETPIE2015-WS\_v25\_TH.pdf or NETPIE2015\_WS\_vx\_EN.pdf



# NETPIE2020 vs NETPIE2015



|                                  | NETPIE 2020                                                                        | NETPIE 2015                                                          |
|----------------------------------|------------------------------------------------------------------------------------|----------------------------------------------------------------------|
| <b>Design Philosophy</b>         | Platform - Centric                                                                 | Device – Centric                                                     |
| <b>Commercial Ability</b>        | Commercial – Ready                                                                 | Need to re-program Hardware migrate to commercial platform           |
| <b>Suitable Usage</b>            | Consumer product, Project-based (3 <sup>rd</sup> -Party)                           | Project-based                                                        |
| <b>Target User</b>               | IoT consumer product makers, Hobbyists, Students                                   | Makers, hobbyists, students                                          |
| <b>Communication Protocol</b>    | MQTT, HTTP                                                                         | Microgear                                                            |
| <b>Programming Language</b>      | Any languages with MQTT library support                                            | Limited to Microgear library                                         |
| <b>Hardware Support</b>          | Unlimited as long as it supports MQTT                                              | Limited to those with Microgear support                              |
| <b>Device Identity and Group</b> | No APPID Device identity and group can be adjusted after product is sold/installed | Use APPID Device identity and group must be programmed into firmware |
| <b>Rate Limit</b>                | Allow burst                                                                        | Everyone is subject to the same rate limit                           |
| <b>Trigger</b>                   | Can set trigger action in cloud platform                                           | Set trigger action inside IoT devices                                |

## NETPIE2015

- NETPIE2015 library

```
#include "MicroGear.h"
```

- Application related keys

```
#define APPID    "..." // from NetPie AppID
#define KEY      "..." // Application Key from NetPie
#define SECRET   "..." // Session Key from NetPie
#define ALIAS    "PM25" // Alias of this device
```



## NETPIE2015

- Feed related keys

```
#define FEEDID "..." // Feed ID from NetPie
#define FEED_INTERVAL 2000 // How often you are sending data to feed
#define PRES_NAME "pres" // Same feed name on NetPie for pressure
#define TEMP_NAME "temp" // Same feed name on NetPie for temperature
#define HUMID_NAME "hum" // Same feed name on NetPie for humidity
#define PM10_NAME "pm10" // Same feed name on NetPie for PM 10
#define PM25_NAME "pm2_5" // Same feed name on NetPie for PM 2.5
#define PM1_NAME "pm1_0" // Same feed name on NetPie for PM 1.0
```

- Application related keys

```
#define APPID "..." // from NetPie AppID
#define KEY "..." // Application Key from NetPie
#define SECRET "..." // Session Key from NetPie
#define ALIAS "PM25" // Alias of this device
```

## NETPIE2015

- Instantiate a Microgear object

```
MicroGear microgear(client); // client is WiFiClient object
```

- Configure microgear object with keys and connect to NETPIE2015

```
microgear.on(CONNECTED, onConnected); // Call onConnected() when NETPIE connection is established  
microgear.init(KEY, SECRET, ALIAS); // Initiate Microgear with KEY, SECRET and also set the ALIAS here  
microgear.connect(APPID); // connect to NETPIE with a specific APPID
```

- Method connected() can be used to check if the device currently connects to NETPIE or not

```
microgear.connected()
```



## NETPIE2015

- Create a string to be transmitted to NETPIE 2015's feed

```
sprintf(dataBuffer, "%s:%d,%s:%d,%s:%d,%s:%.2f,%s:%.2f,%s:%.2f",
    PM1_NAME, data.PM_AE_UG_1_0, PM25_NAME, data.PM_AE_UG_2_5, PM10_NAME,
    data.PM_AE_UG_10_0, PRES_NAME,pres,
    TEMP_NAME, temp, HUMID_NAME, hum);
```

- Write the created data string to feed

```
microgear.writeFeed(FEEDID, dataBuffer);
```

- Write each data value to NETPIE 2015's application

```
microgear.chat(PM1_NAME, data.PM_AE_UG_1_0); // PM 1.0
microgear.chat(PM25_NAME, data.PM_AE_UG_2_5); // PM 2.5
microgear.chat(PM10_NAME, data.PM_AE_UG_10_0); // PM 10
microgear.chat(PRES_NAME, pres); // Pressure
microgear.chat(TEMP_NAME, temp); // Temperature
microgear.chat(HUMID_NAME, hum); // Humidity
```

## NETPIE2015

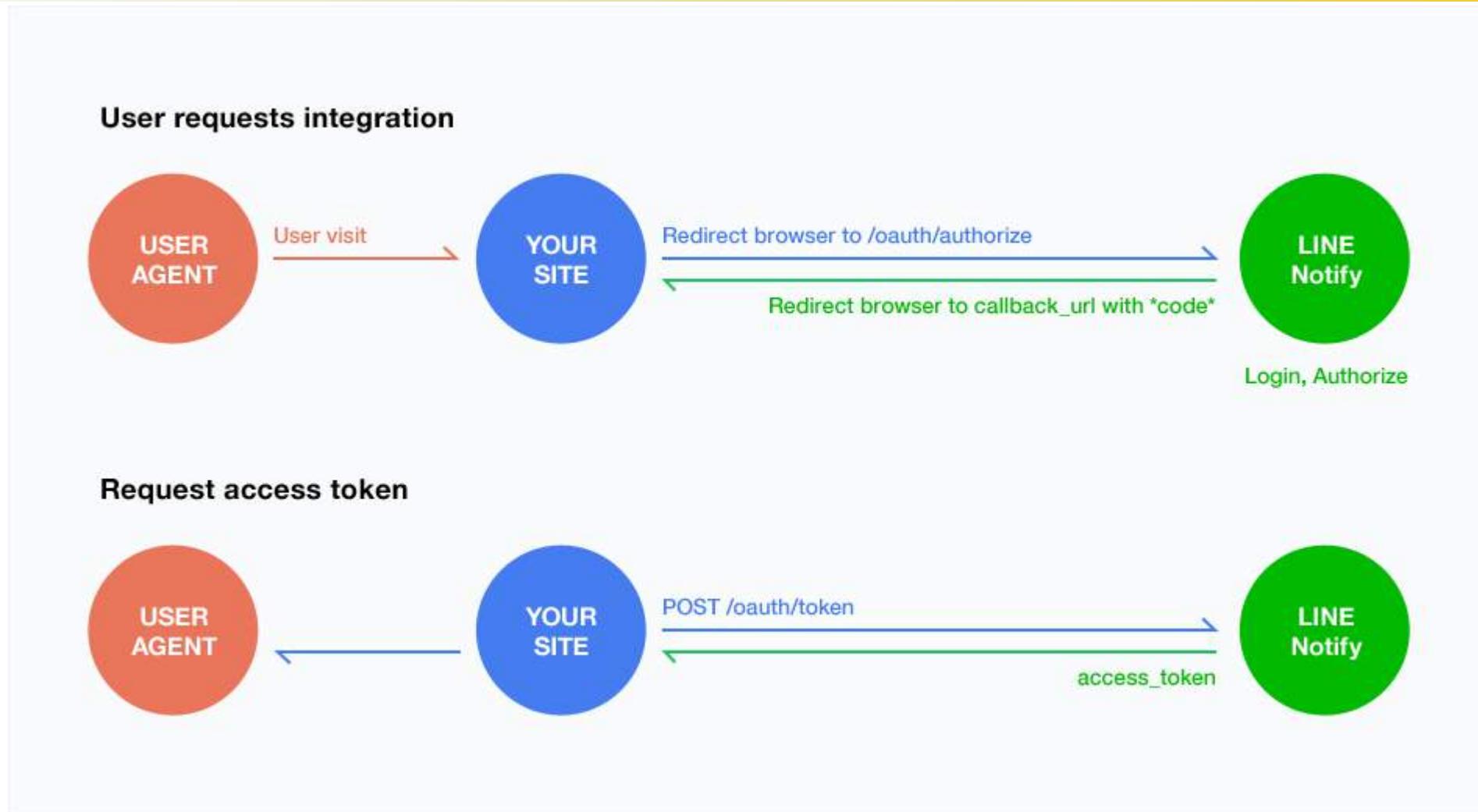
- **onConnect()** event handler function

```
void onConnected(char *attribute, uint8_t* msg, unsigned int msglen)
{
}

/* Hook onConnected() to the "on connect" event
(Automatically call onConnected() when the device connects to NETPIE) */
microgear.on(CONNECTED, onConnected);
```

- Make NETPIE know the alias of this device

```
microgear.setAlias(ALIAS); // Set the alias of this microgear ALIAS
```





- We can use one LINE account to notify and send a message to the other LINE accounts.
- We can Push Notification from a device or an application to a Line Account or Line Group
- First, we need to Log in at <https://notify-bot.line.me/en/> via our own Line Account
  - Then go to “My page” and to “Generate token”  
Note: This Token is linked to this Line Account or this Line Group which we will send a message to
  - Copy the Token for your program



## Token Generation

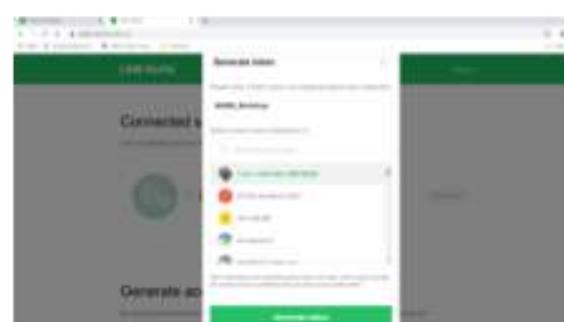


### Generate access token (For developers)

By using personal access tokens, you can configure notifications without having to add a web service.

Generate token

LINE Notify API Document



Your token is:

FnYc40Ysi6jZBMatwikN1HN9Gjqzz72aeVpQM!

If you leave this page, you will not be able to view your newly generated token again. Please copy the token before leaving this page.

Copy

Close



## From a Device

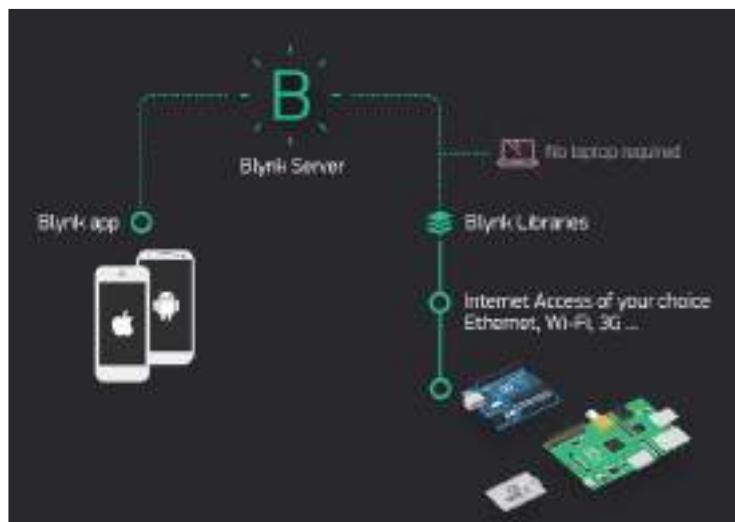
- Send a message to [notify-api.line.me/api/notify](https://notify-api.line.me/api/notify) with Token

```
POST /api/notify HTTP/1.1
Host: notify-api.line.me
Authorization: Bearer <Line Token>
Cache-Control: no-cache
User-Agent: ESP32
Content-Type: application/x-www-form-
urlencoded
Content-Length: <Length of Message>

message=<Message>
```

## <https://blynk.io>

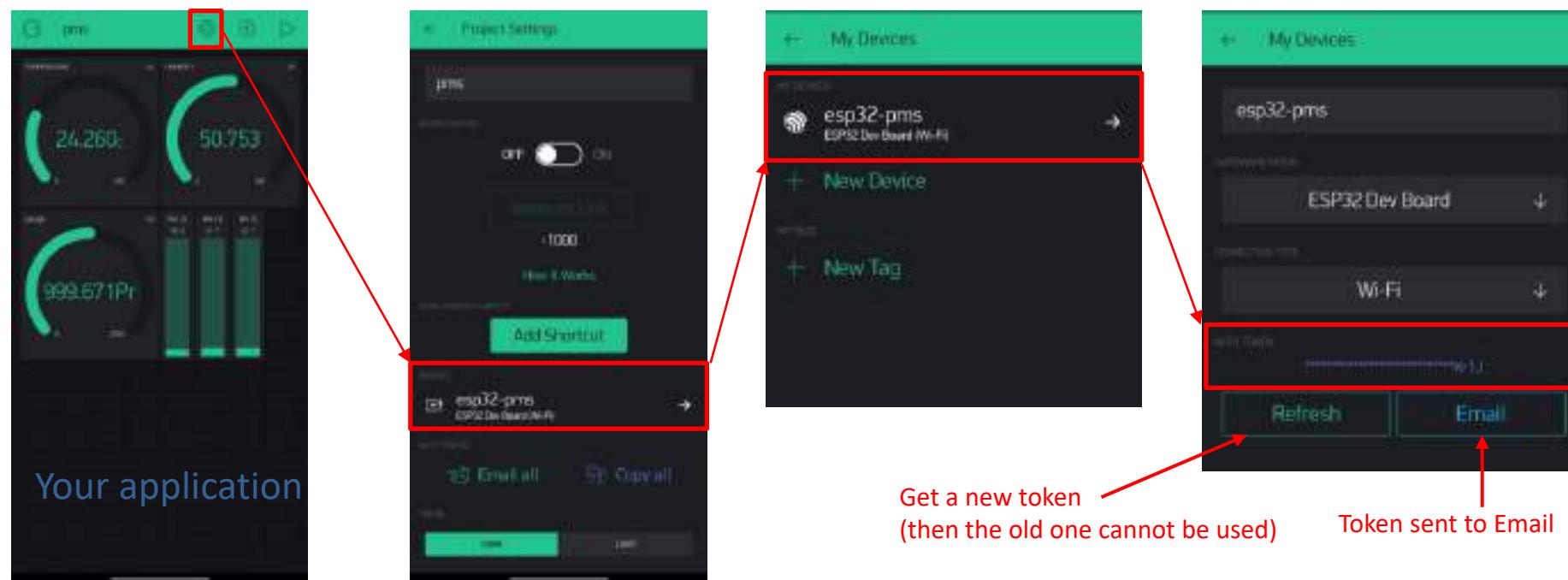
- Blynk is an APP for Android or iPhone (or tablets).
- It is a configurable control panel to read and control various GPIO devices on your hardware. It works across the internet and through firewalls. because it uses a Blynk Server out on the cloud.



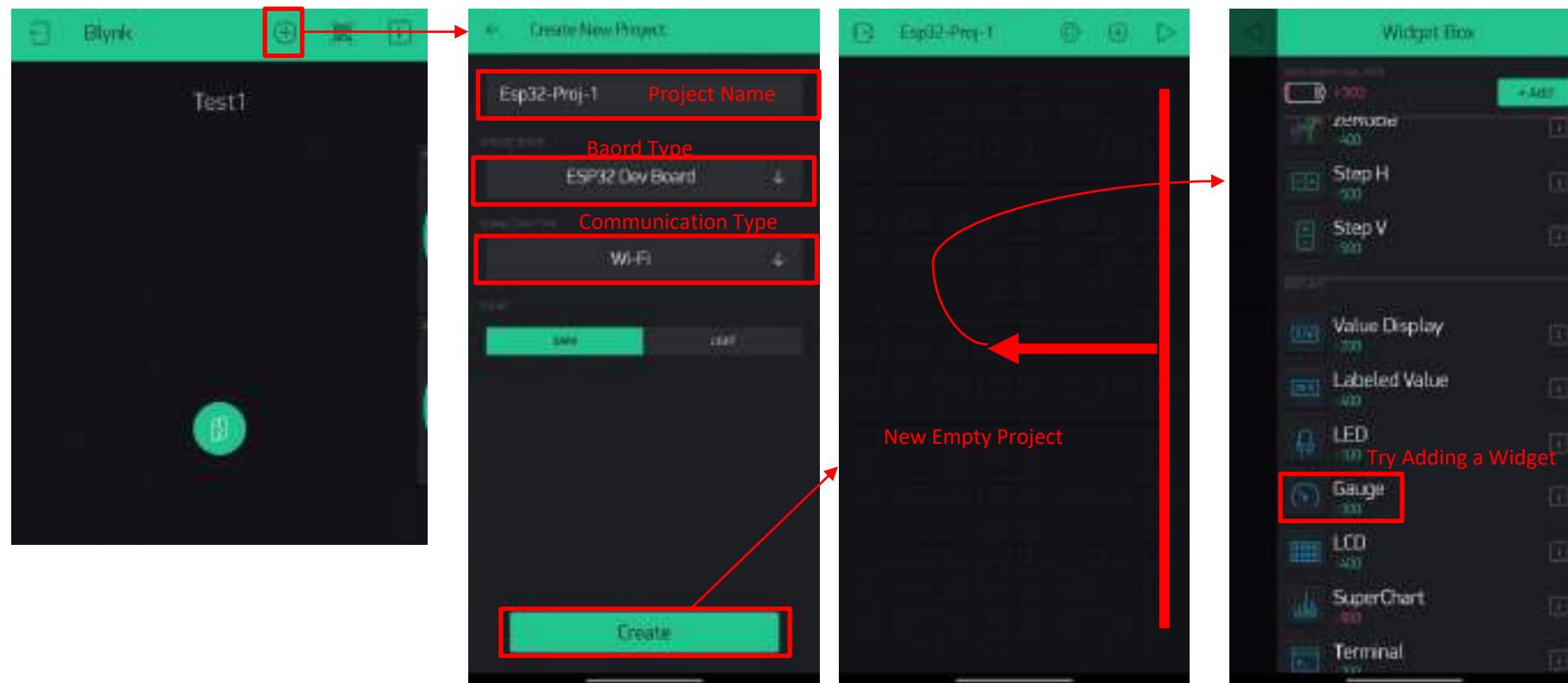
# Blynk on ESP 32 (1)



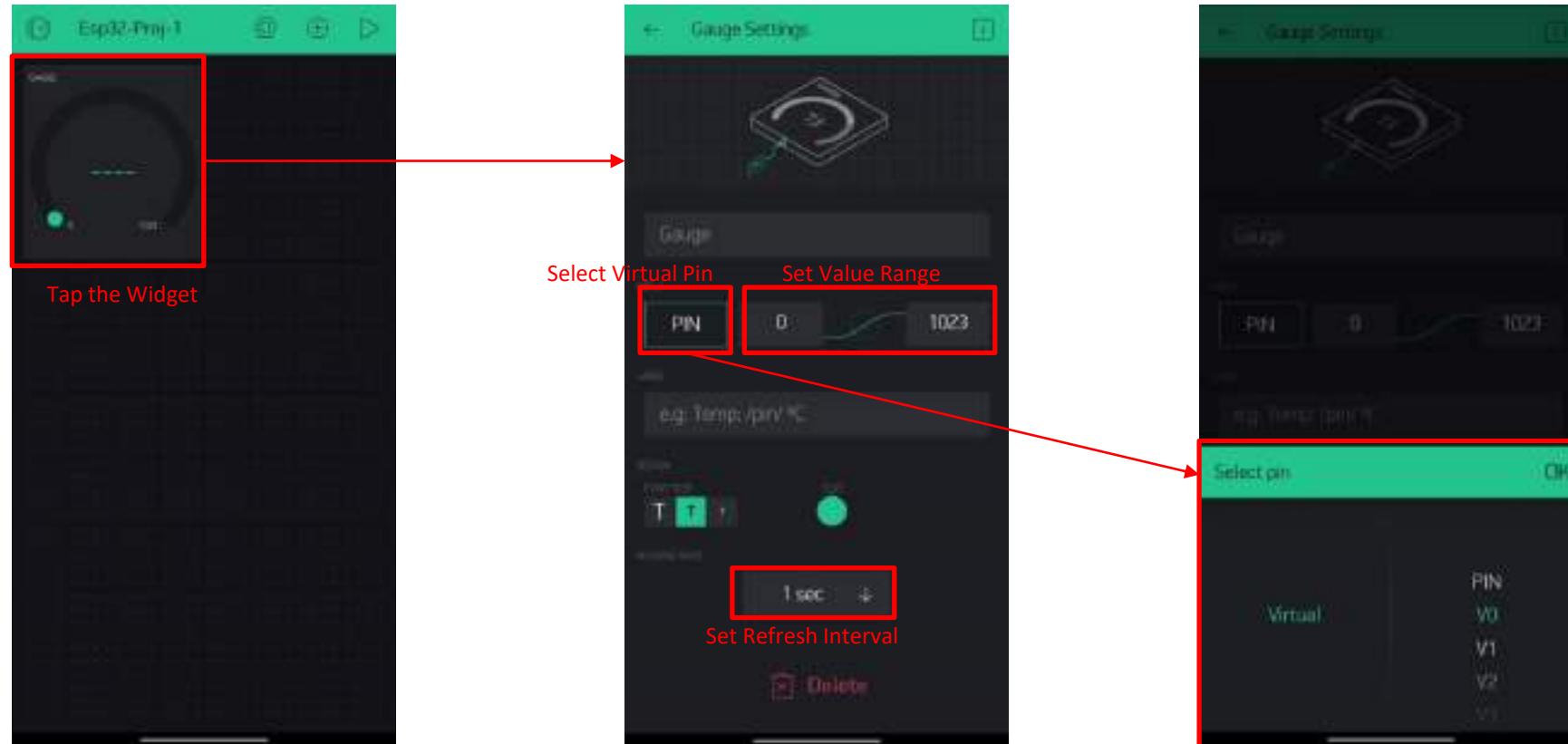
- You can connect your Blynk application to your ESP 32 device by using **Auth Token** which can be found by following the steps below.



# Creating Blynk Project (1)



# Creating Blynk Project (2)



# Blynk on ESP 32 (2)



- Example code on <https://examples.blynk.cc/>



```
/* Comment this out to disable prints and save space */
#define BLYNK_PRINT Serial

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <DHT.h>

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "YourAuthTokens";

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "YourNetworkName";
char pass[] = "YourPassword";

#define DHTPIN 2          // What digital pin we're connected to

// Uncomment whatever type you're using!
#define DHTTYPE DHT11    // DHT 11
//#define DHTTYPE DHT22   // DHT 22, AM2302, AM2321
//#define DHTTYPE DHT21   // DHT 21, AM2301

DHT dht(DHTPIN, DHTTYPE);
BlynkTimer timer;

// This function sends Arduino's up time every second to Virtual Pin #5.
// In the app, Widget's reading frequency should be set to PUSH. This means
// that you define how often to send data to Blynk App.
void sendSensor()
{
  float h = dht.readHumidity();
  float t = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  // You can send any value at any time.
}
```

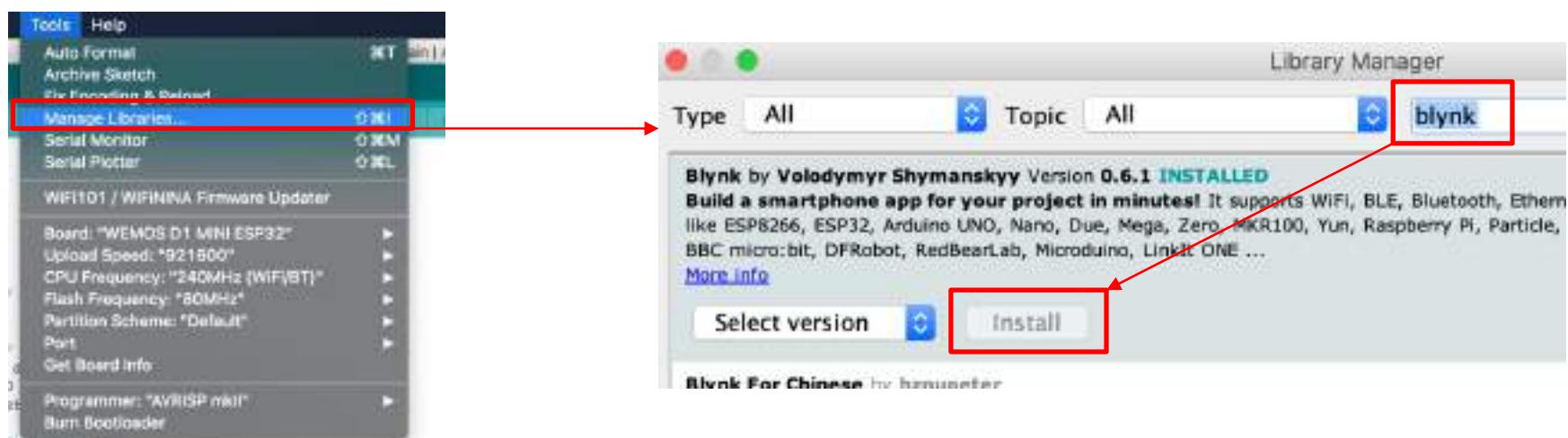
# Blynk on ESP 32 (3)



- Blynk library

```
#include <BlynkSimpleEsp32.h>
```

- Installing the library to Arduino IDE



- Configure the Blynk token and connect to Blynk server

```
bool blynkConnected = false; // Flag determining if this device can connect to Blynk
char auth[] = "..."; //Your Authentication Token here
Blynk.config(auth);
blynkConnected = Blynk.connect(); // connect to Blynk
```

- Write data to each widget in your Blynk application

```
Blynk.run();
Blynk.virtualWrite(V0, data.PM_AE_UG_1_0); // Write PM1.0 value to virtual pin V0
Blynk.virtualWrite(V1, data.PM_AE_UG_2_5); // Write PM2.5 value to virtual pin V1
Blynk.virtualWrite(V2, data.PM_AE_UG_10_0); // Write PM10 value to virtual pin V2
Blynk.virtualWrite(V3, pres); // Write pressure value to virtual pin V3
Blynk.virtualWrite(V4, temp); // Write temperature value to virtual pin V4
Blynk.virtualWrite(V5, hum); // Write humidity value to virtual pin V5
```

- This configuration option is used when you connect to WiFi separately using WiFi library.

```
bool blynkConnected = false; // Flag determining if this device can connect to Blynk  
char auth[] = "..."; //Your Authentication Token here  
Blynk.config(auth);  
blynkConnected = Blynk.connect(); // connect to Blynk
```

- Alternatively, Blynk can connect to WiFi and Blynk in one command in case you do not use a separate WiFi library for WiFi connection

```
char ssid[] = "..."; // Your WiFi SSID.  
char pass[] = "..."; // Set password to "" for open networks.  
char auth[] = "..."; //Your Authentication Token here  
Blynk.begin(auth, ssid, pass);
```

# References and Tutorials



- <https://nodered.org/docs/tutorials/>
- <https://github.com/thingsboard/thingsboard>
- <https://docs.aws.amazon.com/iot/latest/developerguide/iot-dg.pdf>
- <https://www.ibm.com/cloud/internet-of-things>
- <https://thingspeak.com/>
- <https://2015.netpie.io/tutorials#gsc.tab=0>
- <https://netpie.io/tutorials>
- <https://notify-bot.line.me/doc/en/>
- <https://examples.blynk.cc/>



# **ITCS447**

## **Lecture 10**

### **Introduction to NODE-RED**

Asst. Prof. Dr. Thitinan Tantidham



มหาวิทยาลัยมหิดล  
Mahidol University

ห้ามมิให้นักศึกษาทำข้า ดัดแปลง หรือใช้ประโยชน์จากการอันมีลิขสิทธิ์ปราภูอยู่ในระบบการศึกษาอิเล็กทรอนิกส์ (E-Learning) ของมหาวิทยาลัย ไม่ว่าจะทั้งหมดหรือบางส่วน โดยไม่ได้รับอนุญาตจากมหาวิทยาลัย นอกจากนี้จากการศึกษาส่วนบุคคล ทั้งนี้ การทำข้า ดัดแปลง หรือเผยแพร่ต่อสาธารณะชื่องานอันมีลิขสิทธิ์ จะมีโทษปรับตั้งแต่ 20,000 บาท ถึง 200,000 บาท และหากเป็นการกระทำเพื่อการค้า จะมีโทษจำคุกตั้งแต่ 6 เดือน ถึง 10 ปี หรือปรับตั้งแต่ 100,000 บาท ถึง 800,000 บาท หรือทั้งจำทั้งปรับ

Unless allowed by Mahidol University, the User shall not copy, modify, or exploit in part or in whole of the copyrighted materials on the Platform, other than for your own individual study. The copy, modify or communication to public of the copyrighted materials shall be inflicted with a fine from 20,000 Baht up to 200,000 Baht. If the offence is committed with the commercial purpose, the offender shall be inflicted with imprisonment for a term from 6 months up to 10 years or a fine from 100,000 Baht up to 800,000 Baht or both imprisonment and fine.



# Introduction to Node-RED

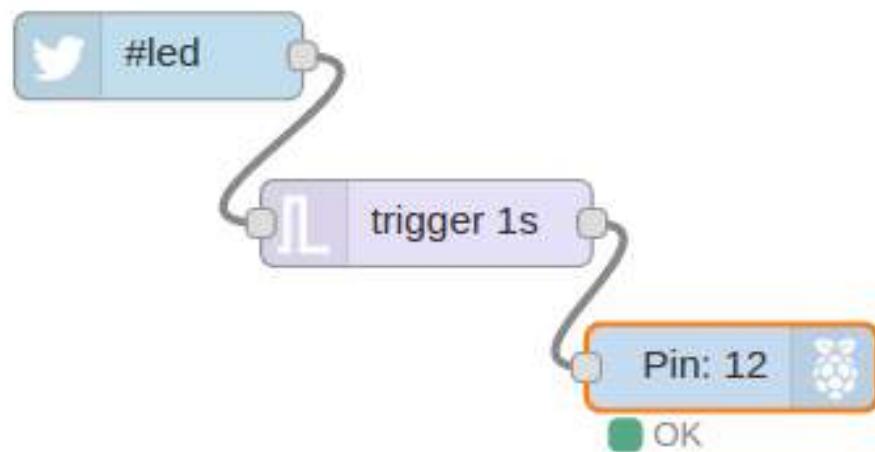


- Originally developed as an open source project at IBM in late 2013, to meet their need to quickly connect hardware and devices to web services and other software – as a sort of glue for the IoT – it has quickly evolved to be a general purpose an **IoT visual programming tool**.
- It allows developers to connect predefined code blocks, known as ‘nodes’, together to perform a task and to reuse Node-RED code for a wide variety of tasks.
- The connected nodes, usually a combination of input nodes, processing nodes and output nodes, when wired together, make up a ‘flows’.
- The flow can be represented in json.
- Many node can be functioned in Javascript.
- Node-RED as “a browser based on flow language” runs on pi or a server.

# Node-RED Functionalities on Raspberry Pi (RPi)



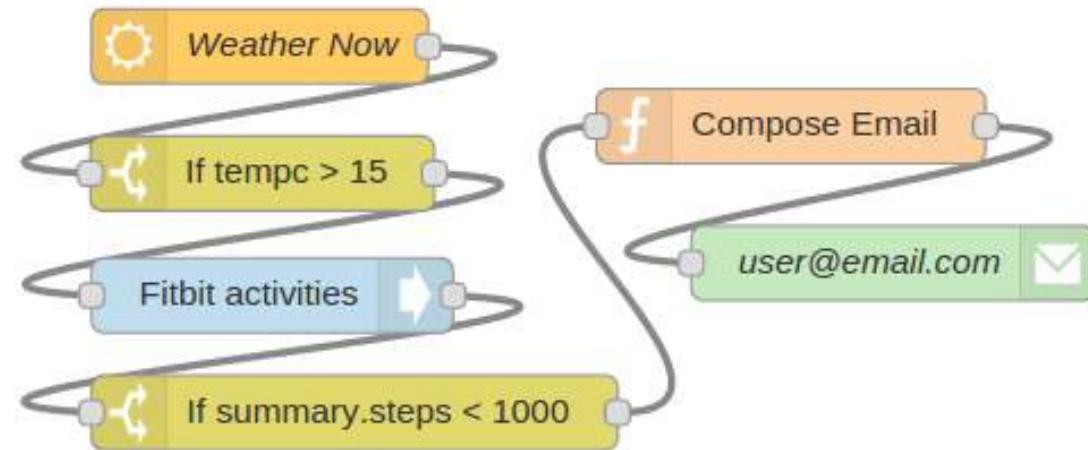
- Access your RPi GPIOs;



- The Twitter node is a built-in node in Node-RED and **hides all of the complexity of using the Twitter API**. It can be configured with a user's account credentials and a variety of search strings, in our case it's simply looking for hashtag '#led'.
- When the Twitter node sees the tag either in a tweet to the user, or the public tweet feed, it creates a new message with the details of the tweet, which is forwards to the next node in the flow.
- The trigger node is to wait for any message on its input. When it receives a message, it 'triggers', and sends a message on its output with the value "1" as the message body. It then waits 1 second and sends a second message with the value "0" in the message body.
- The trigger node controls the input/output on the RPi every 1 s.
- If RPi Pin12 is wired up with an LED connected to Pin 12, the gpionode going to on and off according to "1" and "0"

[https://raw.githubusercontent.com/SenseTecnic/nrbookflows/master/lesson1/1-1\\_twitter.json](https://raw.githubusercontent.com/SenseTecnic/nrbookflows/master/lesson1/1-1_twitter.json)

# Node-RED Functionalities



- Openweathermap
- Switch
- Fitbit

- Establish an MQTT connection with other devices (Arduino, ESP8266, ESP32 etc);
- Create a responsive graphical user interface for your projects;
- Communicate with third-party services (IFTTT.com, Adafruit.io, ThingSpeak, Home Assistant, [InfluxDB](#) etc);
- Retrieve data from the web (weather forecast, stock prices, emails. etc);
- Create time-triggered events;
- Store and retrieve data from a database.

[https://raw.githubusercontent.com/SenseTecnic/nrbookflows/master/lesson1/1-2\\_weatheralert.json](https://raw.githubusercontent.com/SenseTecnic/nrbookflows/master/lesson1/1-2_weatheralert.json)



## Installation (with npm)

- <https://nodered.org/docs/getting-started/>
- <https://nodered.org/docs/getting-started/local>
- <https://nodered.org/docs/faq/starting-node-red-on-boot>

(1) Install Node-RED

```
sudo npm install -g --unsafe-perm node-red
```

(2) Try to run node-red

The screenshot shows a terminal window titled 'node-red' with the following text:

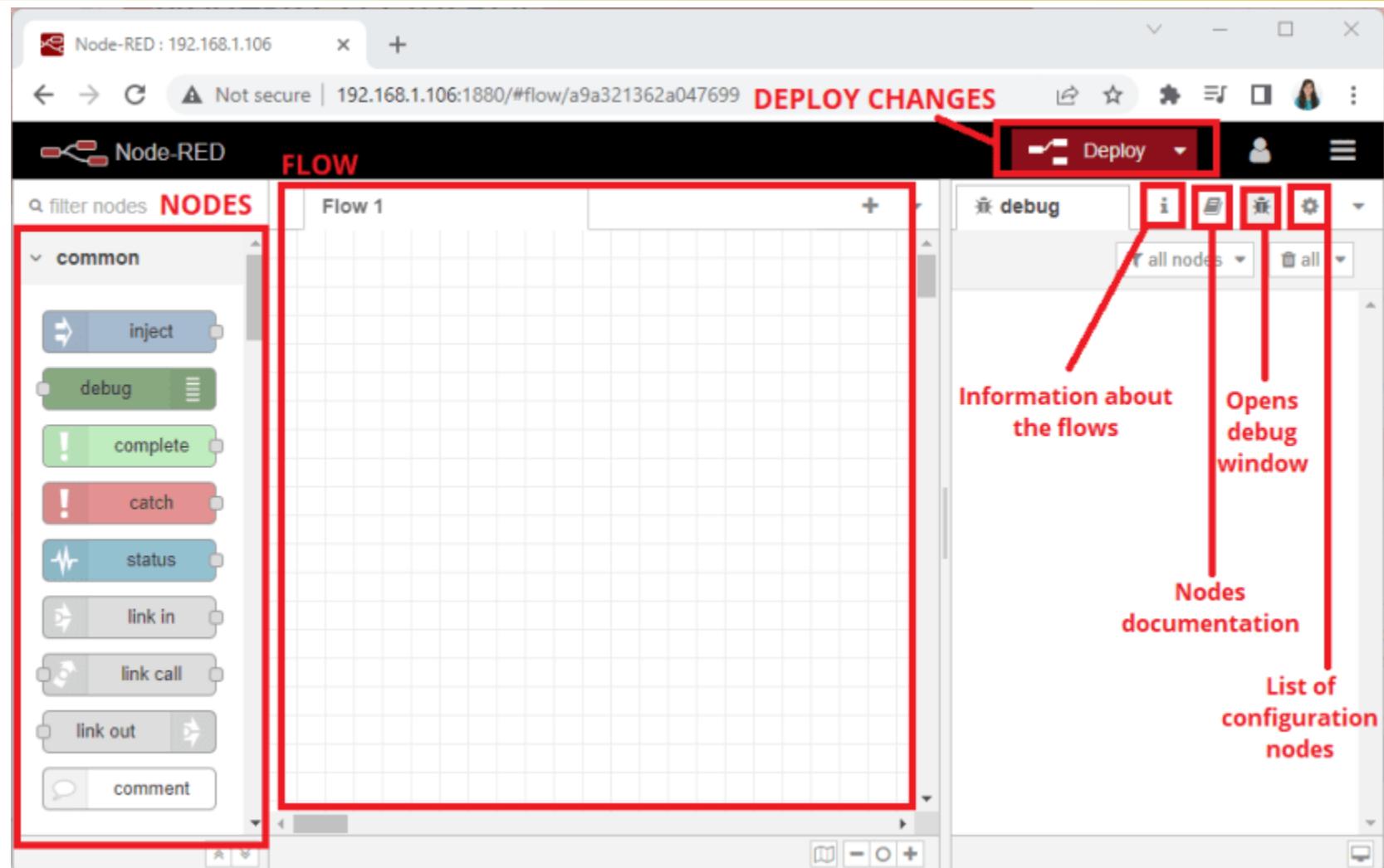
```
Welcome to Node-RED
-----
30 Jan 23:43:39 [info] Node-RED version: v1.5.9
30 Jan 23:43:39 [info] Node.js version: v14.7.0
30 Jan 23:43:39 [info] Darwin 18.6.0 x64 LE
30 Jan 23:43:39 [info] Loading palette nodes
30 Jan 23:43:44 [warn] npn-gpi - Raspberry Pi specific node set inactive
30 Jan 23:43:44 [info] Settings file: /Users/steve/.node-red/settings.js
30 Jan 23:43:44 [info] HEEF Status: /Users/steve/.node-red/heef
30 Jan 23:43:44 [info] Context store: 'default' [node:localfilesystem]
30 Jan 23:43:44 [info] User directory: /Users/steve/.node-red
30 Jan 23:43:44 [warn] Projects enabled: set editorTheme.projects.enabled=true to enable
30 Jan 23:43:44 [info] Creating new flow file: flow.voltage.json
30 Jan 23:43:44 [info] Starting flows
30 Jan 23:43:44 [info] Started flows
30 Jan 23:43:44 [info] Server now running on http://127.0.0.1:1880/red/
```

(3) Install PM2: process manager for node.js

```
sudo npm install -g pm2
pm2 start /usr/local/bin/node-red -v
```

Note: If you have done a global install of node-red, then on Linux/OS X the node-red command will probably be either: /usr/bin/node-red or /usr/local/bin/node-red. The command which node-red can be used to confirm the location.

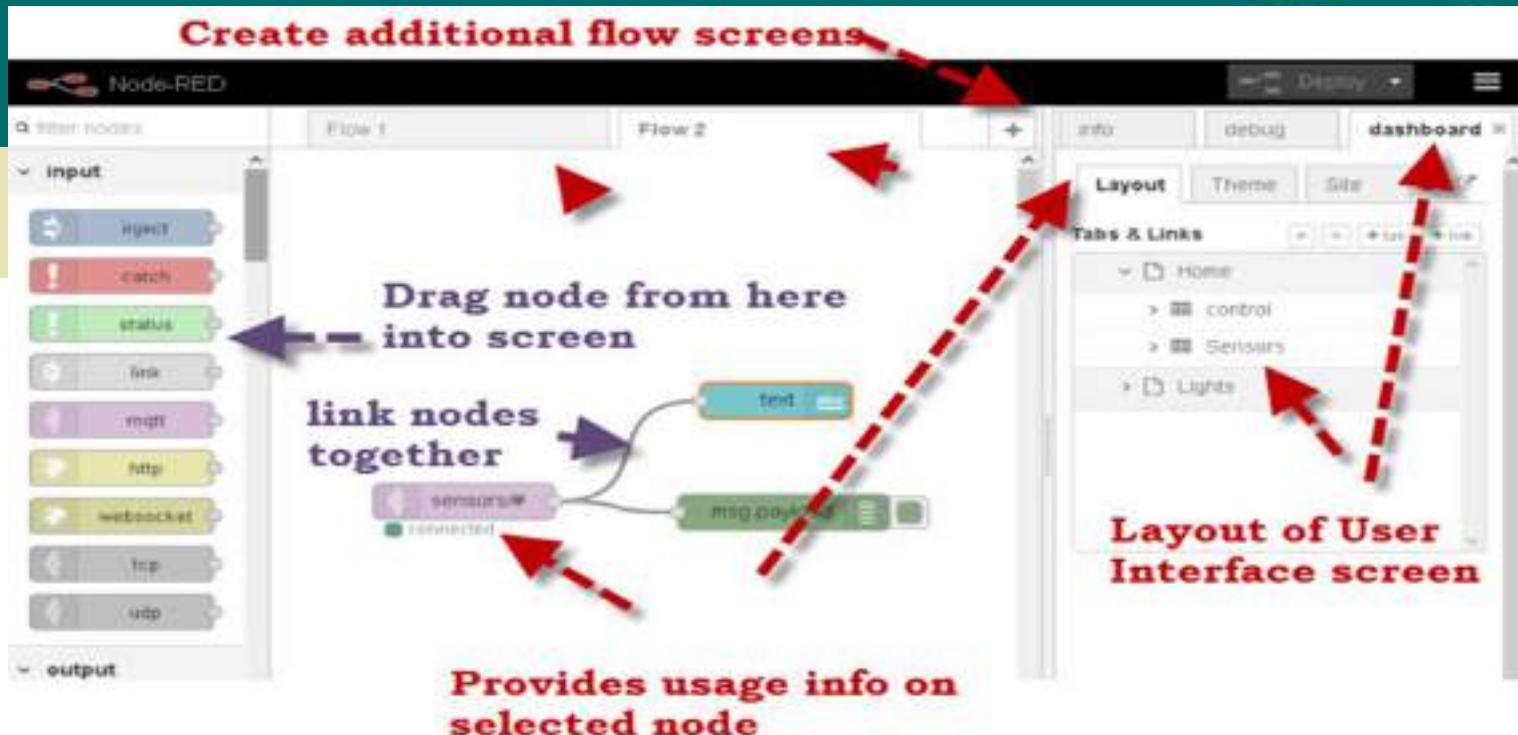
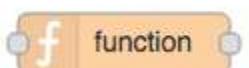
# Node-RED Interface Main Sections



# Node-RED

## Nodes

- Input Nodes (e.g. inject)
- Processing Nodes (e.g. function)
- Output Nodes (e.g. debug)





## Exercise: Simple Flow

The flow we'll create, simply prints a message to the debug console, when triggered.

Drag an **inject** node and a **debug** node to your flow and wire them together.



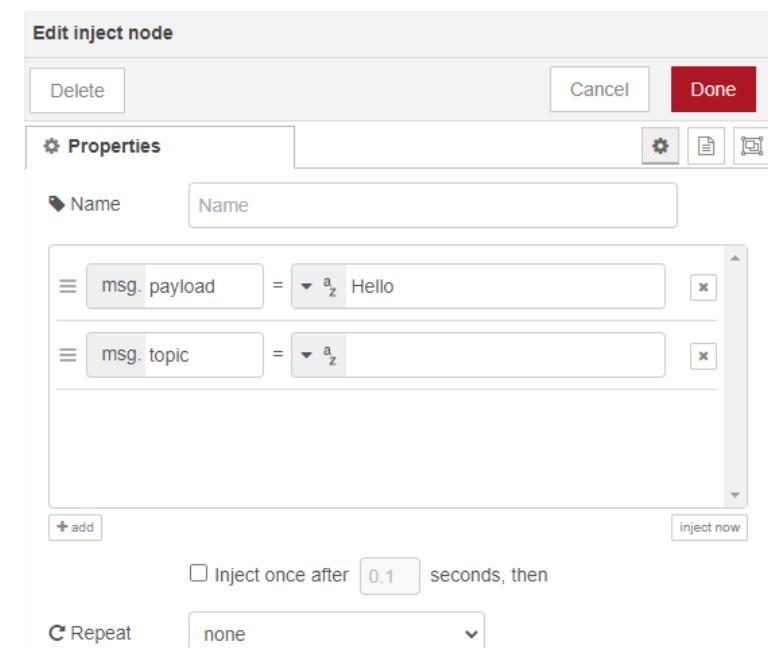
Now, let's edit the inject node. Double-click the node. In the figure below, you can see the different settings you can change.

On the `msg.payload` field, select string and type `Hello`. Then, click **Done**.

To save your application, you need to click the **Deploy** button in the top right corner.



Your application is saved

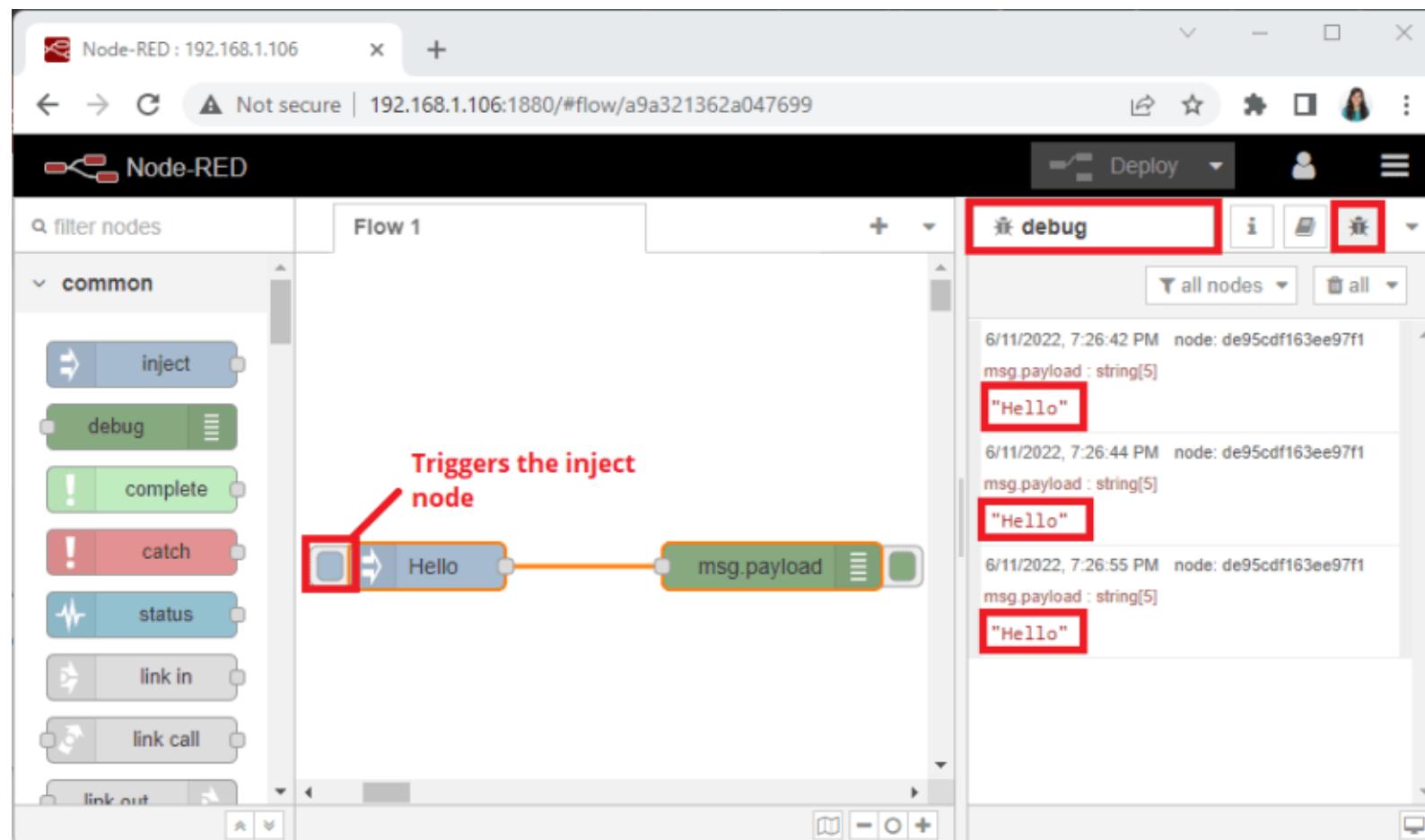


Ref: <https://randomnerdtutorials.com/getting-started-node-red-raspberry-pi/>



# Exercise: Simple Flow Test

Let's test our simple flow. Open the **debug** window and click the **inject** node to trigger the flow.

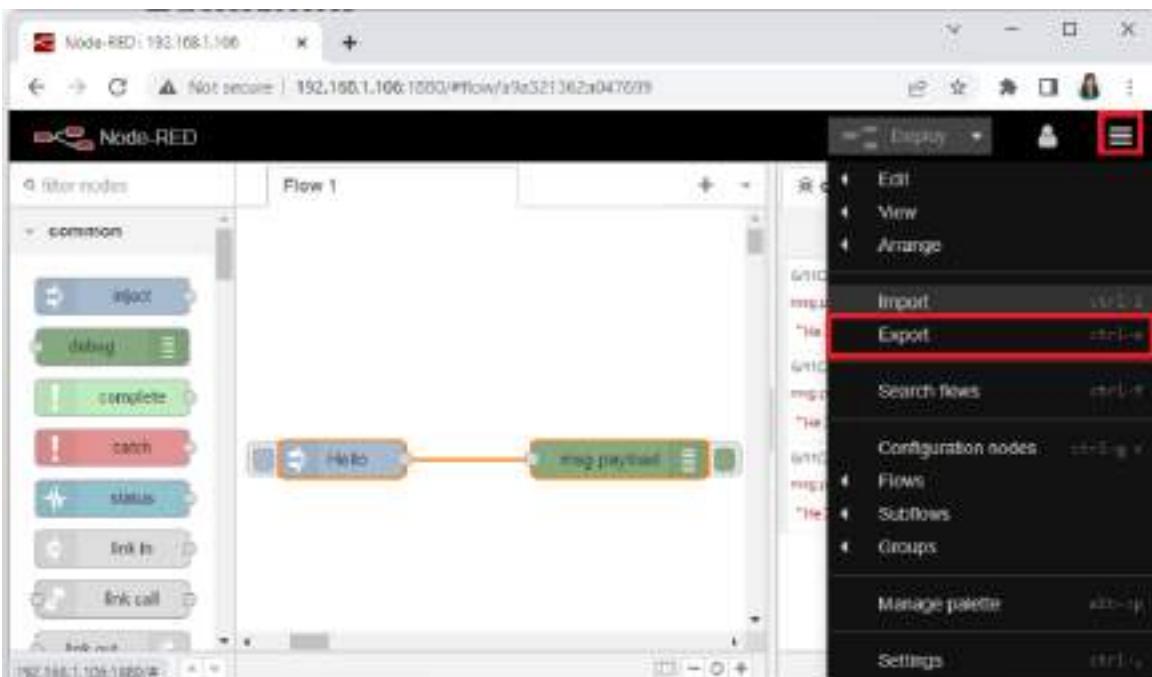


Ref: <https://randomnerdtutorials.com/getting-started-node-red-raspberry-pi/>

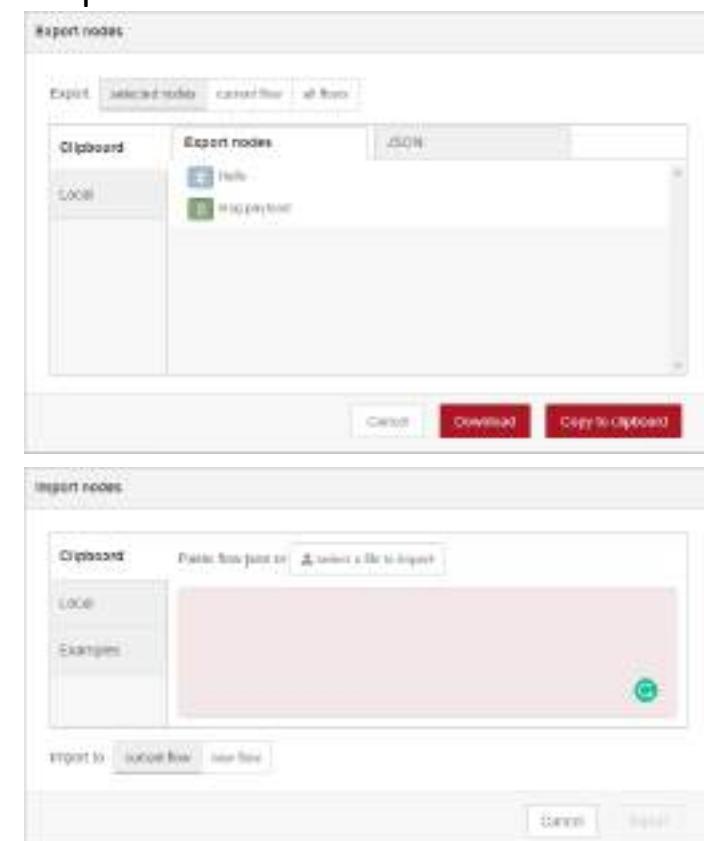


## Exporting and Importing Nodes

- Backup your Node-RED flow
- Move your flow to another Node-RED system
- Share your Node-RED project with others



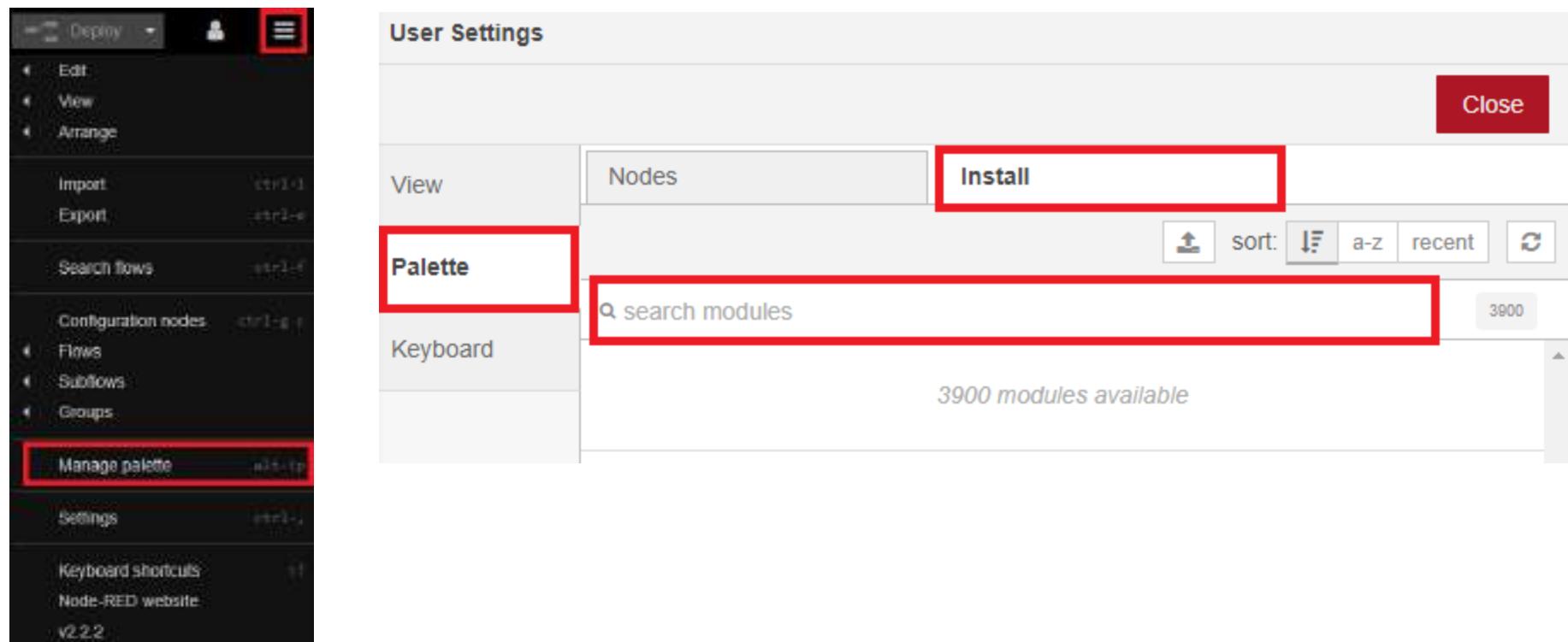
You can select if you want to save the selected nodes, the current flow, or all flows. You can also download the nodes as a JSON file or copy the JSON to the clipboard.



Ref: <https://randomnerdtutorials.com/getting-started-node-red-raspberry-pi/>

## Installing Palette Nodes

- There are many more nodes available that you can install and use for your projects. You can find them in the [Node-RED library](#). If you need some specific task for your project, there's probably already a node for that.



Ref:

<https://randomnerdtutorials.com/getting-started-node-red-raspberry-pi/>

<https://flows.nodered.org/>



## Dashboard

- Node-RED Dashboard is a module that provides a set of nodes in Node-RED to quickly create a live data dashboard. You can install those nodes using the **Menu > Manage Palette**. Then, search for node-red-dashboard and install it.

The screenshot shows the Node-RED 'Manage Palette' dialog. On the left, there's a sidebar with 'User Settings', 'View' (with 'Nodes' selected), and a red-bordered 'Palette' button. The main area has a search bar with 'q: node-red-dashboard' and a red-bordered 'Install' button. Below the search bar, the 'node-red-dashboard' package is listed with its version (0.1.7) and a red-bordered 'install' button. Another package, 'node-red-node-ui-list', is also listed below it.



Nodes from the dashboard section provide widgets that show up in your application user interface (UI). The user interface is accessible on the following URL:

**http://Your\_NodeRED\_IP\_address:1880/ui**

Ref:

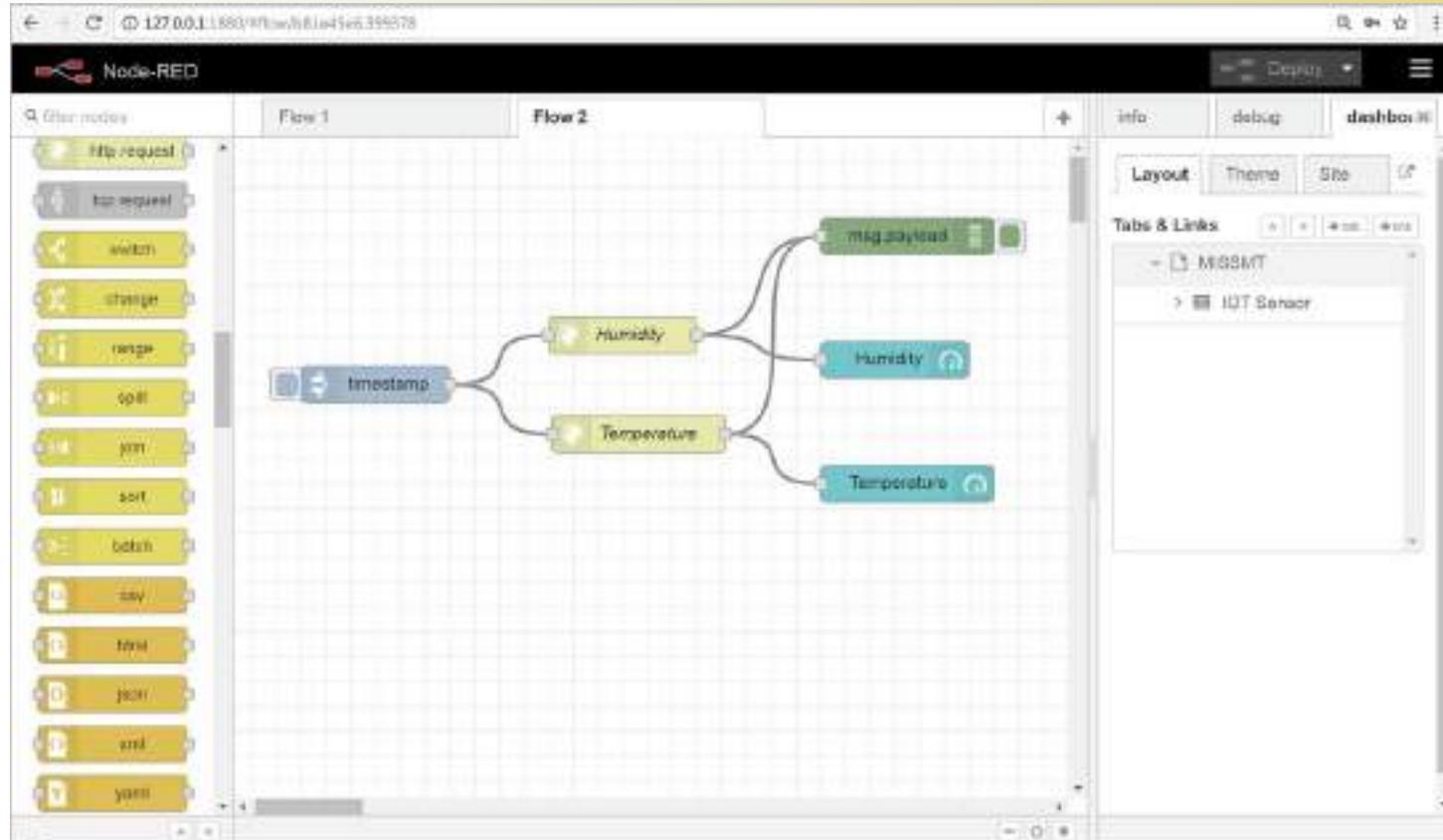
<https://randomnerdtutorials.com/getting-started-node-red-raspberry-pi/>

<https://flows.nodered.org/>

**After Installation**



## Example (RESTful)



Edit inject node

Done Cancel Done

node properties

Payload

Topic

Inject once after  seconds, then

Repeat  Interval  at specific time

every  seconds

Name

Note: "Interval between times" and "at a specific time" will use cron. "Interval" should be less than 240 hours. See intro text for details.

node settings

Edit http request node

Done Cancel Done

node properties

Method

URL

Enable secure (SSL/TLS) connection

Use basic authentication

Return  parsed JSON object  raw string

Name

Tip: If the JSON parse fails the fetched string is returned as-is

node settings

Edit http request node

Done Cancel Done

node properties

Method

URL

Enable secure (SSL/TLS) connection

Use basic authentication

Return  parsed JSON object  raw string

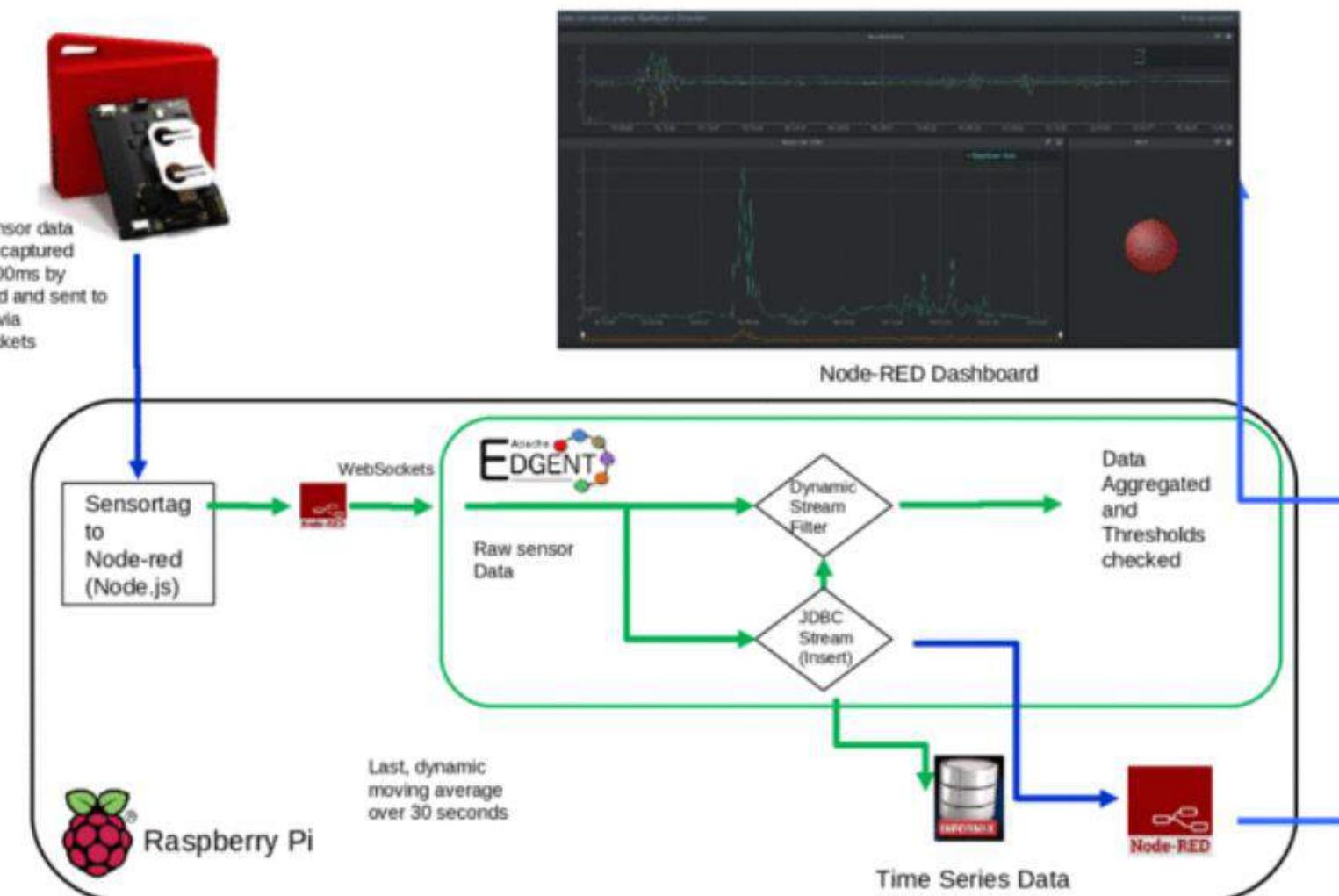
Name

Tip: If the JSON parse fails the fetched string is returned as-is

node settings



## Raspberry Pi

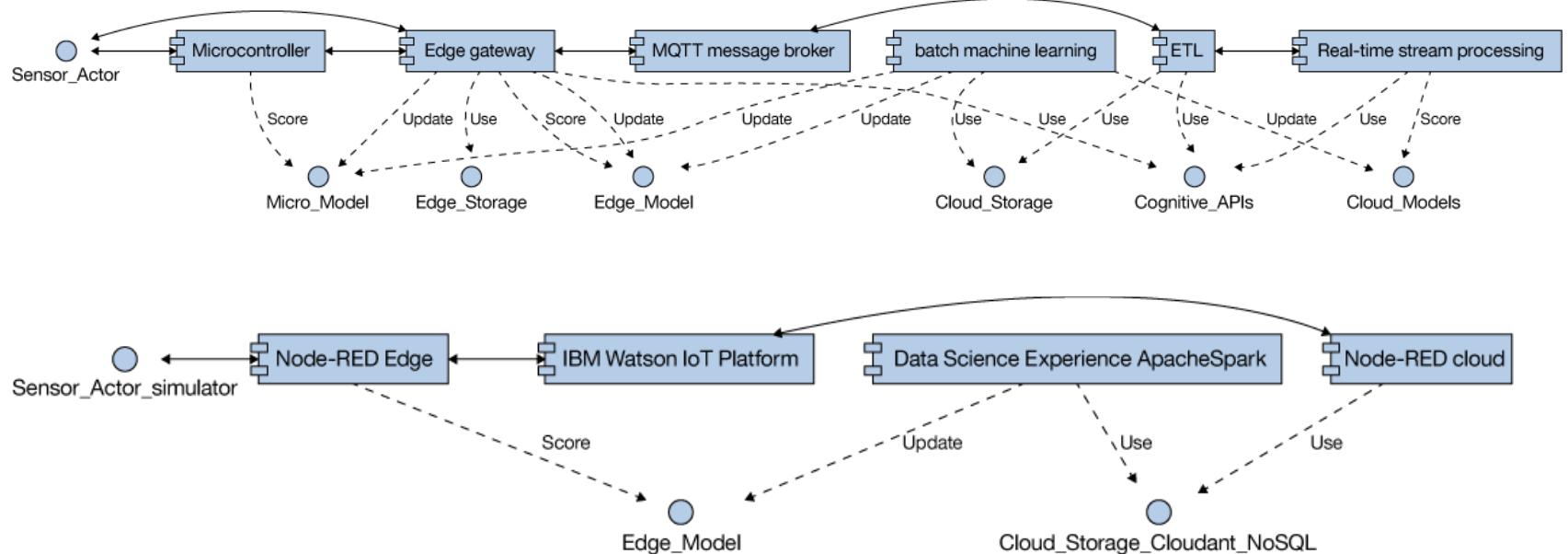


<https://vdocuments.site/iot-analytics-from-edge-to-cloud-using-ibm-informix.html>

# Node-RED Case Studies



IBM



| Component Name               | Component Type                                  | UML Stereotype                       | Component Description                                                                                                                                                                              |
|------------------------------|-------------------------------------------------|--------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Node-RED Edge                | Node-RED                                        | Edge Gateway                         | Used to create data flow application.<br><br>Node-RED is an open source data flow editor written in JavaScript and running on Node.js. IBM created it and donated it to the JavaScript Foundation. |
| Sensor_Actor_simulator       | Node-RED node                                   | Sensor or Actor                      | Used to simulate a sensor or actor in the absence of physical IoT systems.                                                                                                                         |
| Watson Studio                | Apache Spark and Jupyter notebooks as a service | Batch Machine Learning               | Used to detect anomalies in real time on an IoT sensor time-series stream.                                                                                                                         |
| IBM Watson IoT Platform      | IBM Watson IoT Platform                         | MQTT message broker                  | Acts as asynchronous glue between all components in the IoT operational model.                                                                                                                     |
| Node-RED Cloud               | Node-RED                                        | ETL plus real-time stream processing | Used for streaming IoT sensor data in cloud storage.                                                                                                                                               |
| Cloud_Storage_Cloudant_NoSQL | Cloudant                                        | Cloud_Storage                        | Used to store IoT sensor data.<br>Cloudant is an Apache CouchDB as a service. We can also use SQL databases or OpenStack Swift Object Storage (which is the most cost-effective option).           |
| Edge_Model                   | Node-RED                                        | Edge_Model                           | Holds a simple threshold value that gets populated by the Batch Machine Learning component dynamically.                                                                                            |

<https://developer.ibm.com/technologies/iot/tutorials/iot-cognitive-iot-app-machine-learning/>



## Operational Model Components

| Component Name               | Component Type                                  | UML Stereotype                       | Component Description                                                                                                                                                                               |
|------------------------------|-------------------------------------------------|--------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Node-RED Edge                | Node-RED                                        | Edge-Gateway                         | Used to create data flow application.<br><br>Node-RED is an open source data flow editor, written in JavaScript and running on Node.js. IBM created it and donated it to the JavaScript foundation. |
| Sensor_Actor_simulator       | Node-RED node                                   | Sensor or Actor                      | Used to simulate a sensor or actor in the absence of a physical IoT system.                                                                                                                         |
| Watson Studio                | Apache Spark and Jupyter notebooks as a service | Batch Machine Learning               | Used to detect anomalies in real time on an IoT sensor time-series stream.                                                                                                                          |
| IBM Watson IoT Platform      | IBM Watson IoT Platform                         | MQTT message broker                  | Acts as asynchronous glue between all components in the IoT operational model.                                                                                                                      |
| Node-RED Cloud               | Node-RED                                        | ETL plus real-time stream processing | Used for streaming IoT sensor data to cloud storage.                                                                                                                                                |
| Cloud_Storage_Cloudant_NoSQL | Cloudant                                        | Cloud_Storage                        | Used to store IoT sensor data.<br>Cloudant is an Apache CouchDB as a service. We can also use SQL databases or OpenStack Swift Object Storage (which is the most cost-effective option).            |
| Edge_Model                   | Node-RED                                        | Edge_Model                           | Holds a simple threshold value that gets populated by the Batch Machine Learning component dynamically.                                                                                             |

<https://developer.ibm.com/technologies/iot/tutorials/iot-cognitive-iot-app-machine-learning/>

# References & Tutorials



<https://nodered.org/>

<https://flows.nodered.org/>

<https://developer.ibm.com/learningpaths/get-started-node-red/>

<https://stevesnoderedguide.com/node-red-dashboard>

<https://randomnerdtutorials.com/getting-started-node-red-dashboard/>

[https://www.youtube.com/results?search\\_query=node-red](https://www.youtube.com/results?search_query=node-red)

- Node-RED Essential: <https://www.youtube.com/watch?v=ksGeUD26Mw0&list=PLyNBB9VCLmo1hyO-4fIz08gqFcXBkHy-6>
- Understanding Node-RED: <https://www.youtube.com/watch?v=raV5NFInPio>



# ITCS447

## Lecture 10

# IoT Platforms and Infrastructures Edge/Fog/Cloud Computing and Technologies

Asst. Prof. Dr. Thitinan Tantidham



ห้ามมิให้นักศึกษาทำข้า ดัดแปลง หรือใช้ประโยชน์จากการอันมีลิขสิทธิ์ปราการอยู่ในระบบการศึกษาอิเล็กทรอนิกส์ (E-Learning) ของมหาวิทยาลัย ไม่ว่าจะทั้งหมดหรือบางส่วน โดยไม่ได้รับอนุญาตจากมหาวิทยาลัย นอกเหนือจากการศึกษาส่วนบุคคล ทั้งนี้ การทำข้า ดัดแปลง หรือเผยแพร่ต่อสาธารณะชื่องานอันมีลิขสิทธิ์ จะมีโทษปรับตั้งแต่ 20,000 บาท ถึง 200,000 บาท และหากเป็นการกระทำเพื่อการค้า จะมีโทษจำคุกตั้งแต่ 6 เดือน ถึง 10 ปี หรือปรับตั้งแต่ 100,000 บาท ถึง 800,000 บาท หรือทั้งจำทั้งปรับ

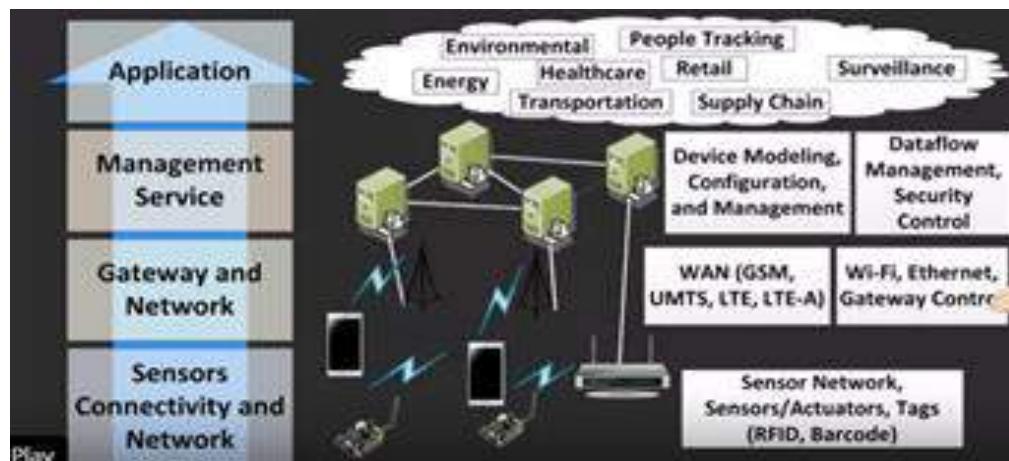
Unless allowed by Mahidol University, the User shall not copy, modify, or exploit in part or in whole of the copyrighted materials on the Platform, other than for your own individual study. The copy, modify or communication to public of the copyrighted materials shall be inflicted with a fine from 20,000 Baht up to 200,000 Baht. If the offence is committed with the commercial purpose, the offender shall be inflicted with imprisonment for a term from 6 months up to 10 years or a fine from 100,000 Baht up to 800,000 Baht or both imprisonment and fine.



- Introduction: IoT Architecture Layers vs TCP/IP Layers
- Multi-Tier Computing
- Fog and Edge Computing
- Edge Computing



## IoT Architecture Layers vs TCP/IP Layers: IoT & Web



### IoT Architecture Layers

### TCP/IP Protocol Stack Layers

#### WEB STACK

*Web applications*

*HTML, XML, JSON*

*HTTP, DHCP, DNS, TLS/SSL*

*TCP, UDP*

*IPv6, IPv4, IPSec*

*Ethernet (IEEE 802.3),  
DSL, ISDN, Wireless LAN  
(IEEE 802.11), Wi-Fi*

#### IOT STACK

*IoT applications | Device Management*

*Binary, JSON, CBOR*

*CoAP, MQTT, XMPP, AMPQ*

*UDP, DTLS*

*IPv6/IP Routing*

*6LOWPAN*

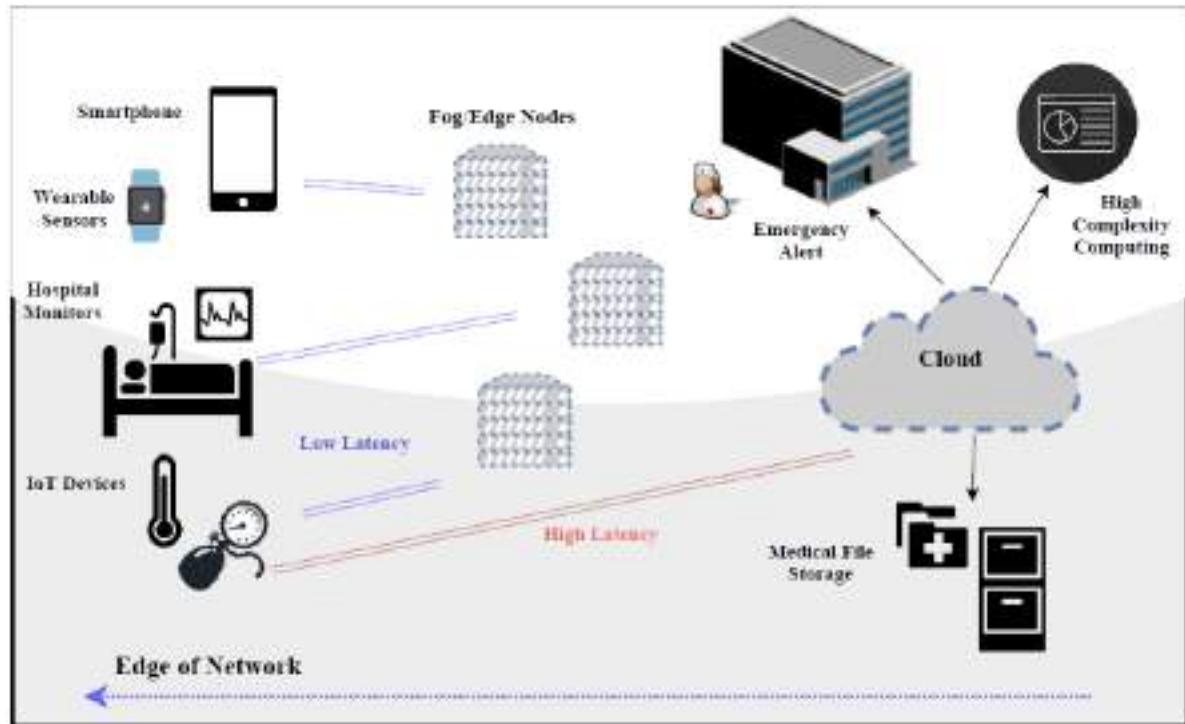
*IEEE 802.15.4 MAC*

*IEEE 802.15.4 PHY / Physical Radio*

Go to:

Coursera, Prof.Jong-Moon Chung Yonsei University – \*0,a,b video files

## Comparison of Edge-Enabled and Legacy Healthcare

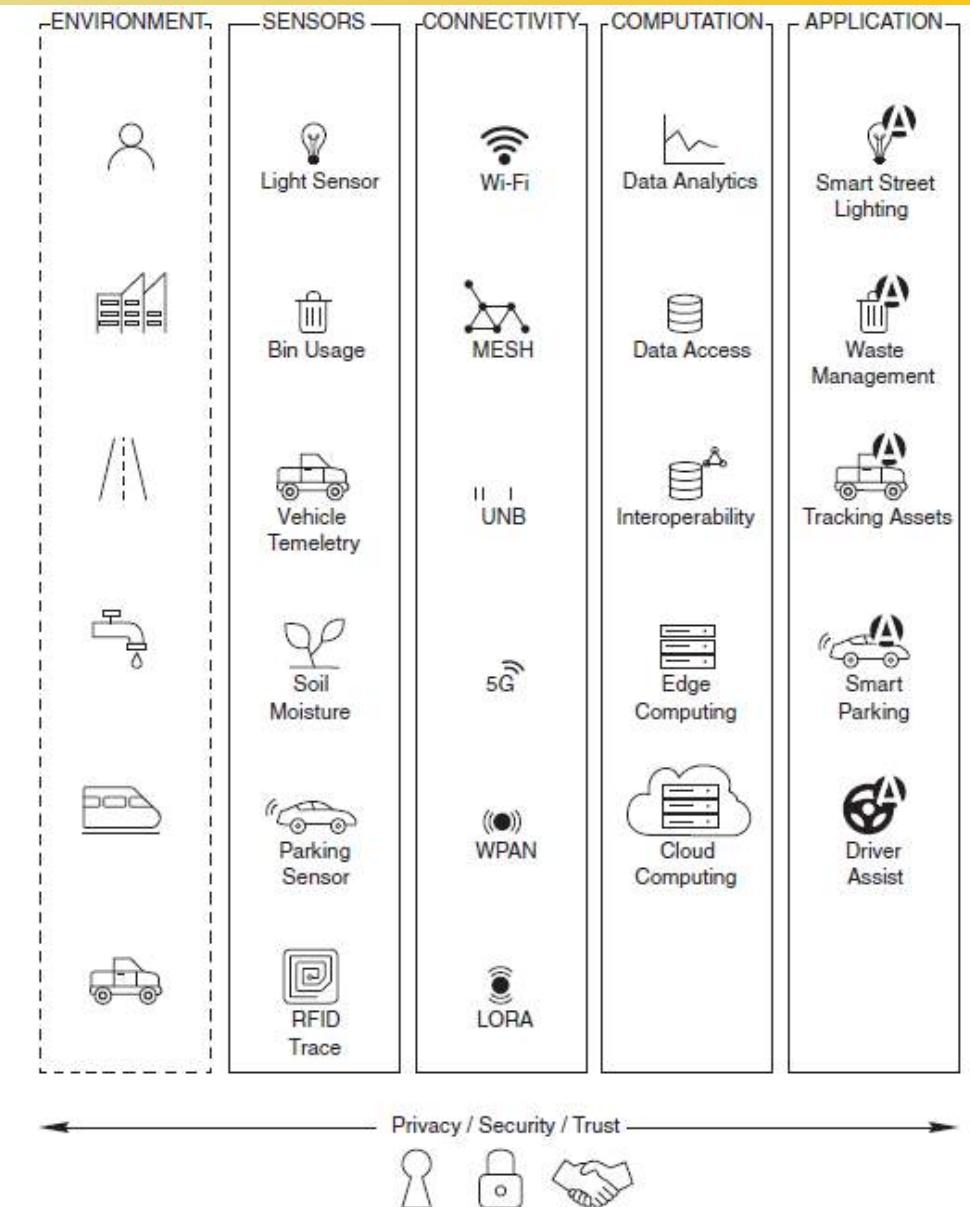


Ref: M. Hartmann, U. S. Hashmi, A. Imran, "Edge Computing in Smart Health Care Systems: Review, Challenges, and Research Directions, 2017

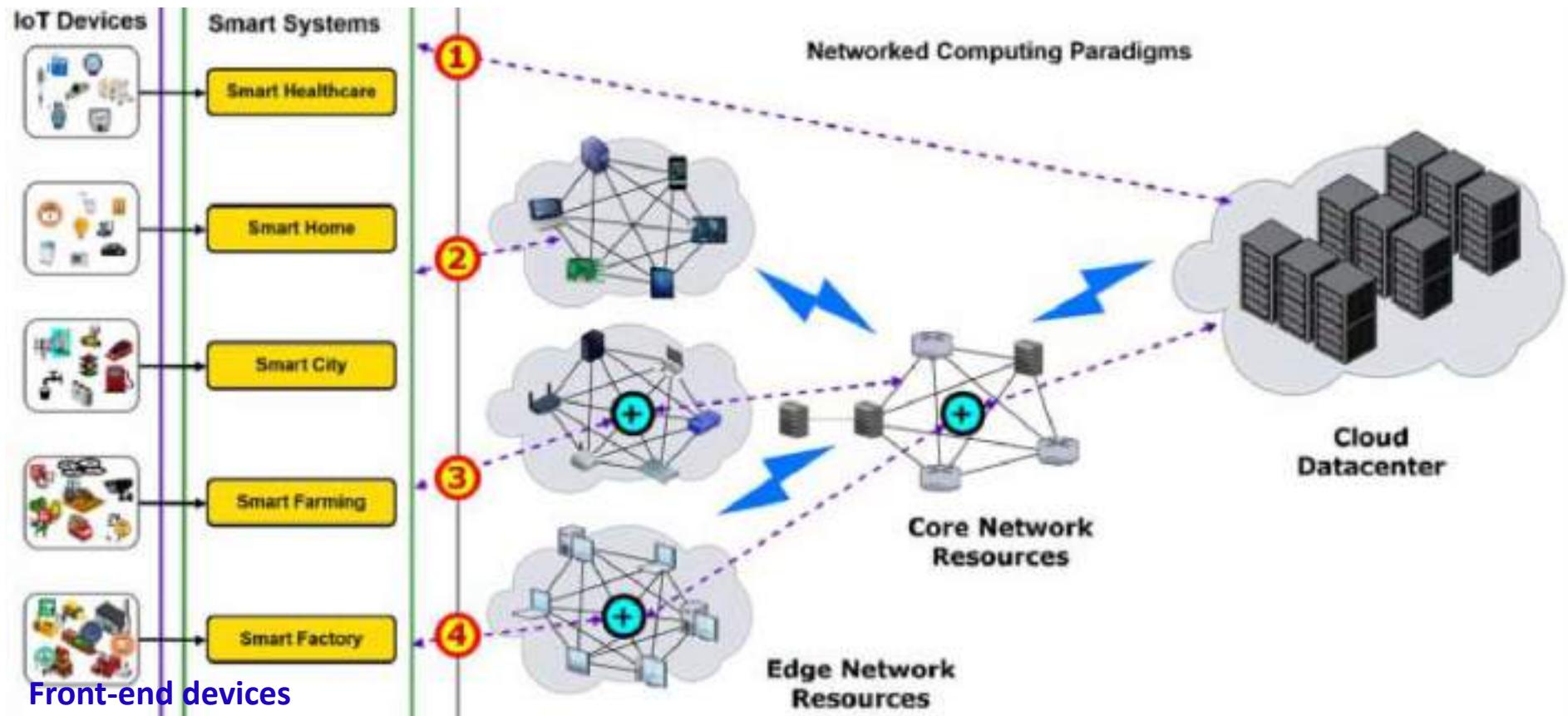


## IoT Ecosystem [DF20]

- IoT Ecosystem consists of multiple components that allow business, governments, and consumers to connect their **interdependent** IoT devices and technologies, such as a smart city environment.
- Stakeholders may play one or more different roles. These include multiple providers and developers for end users such as sensors, connectivities, information, applications, analytics, services, and platforms.
  - ✓ Sensor/Actuators
  - ✓ Device connectivity
  - ✓ Application in the smart device
  - ✓ The Access Network and the Internet (Cellular: \*G: 4G/5G, NB-IoT/Sigfox/LoRaWAN)
  - ✓ The application (and processing) on the cloud
  - ✓ Data Analytics
  - ✓ Security



## IoT and Cloud-centric IoT





## IoT and Cloud-centric IoT

- Commonly, an IoT system follows the architecture of the **Cloud-centric Internet of Things (CIoT)** in which the physical objects are represented in the form of **Web resources** that are managed by the servers in the **global Internet**.
- CiOT faces challenges in **BLURS**
  - **Bandwidth**: the increasing large and high-frequent rate produced in objects in IoT will exceed the bandwidth availability, e.g. a self-driving vehicle can generate gigabytes of data per second due to the need for real-time video streaming.
  - **Latency**: The requirement of controlling the end-to-end latency within tens of milliseconds.
  - **Uninterrupted**: The long distance between cloud and the front-end IoT devices can face issues derived from the unstable and intermittent network connectivity.
  - **Resource-constrained**: CiOT systems usually require front-end devices to continuously stream their data to the cloud.
  - **Security**: rarely updated software and less resource capacity of front-end devices. The attacker may also damage or control the front-end device and send false data to the cloud.

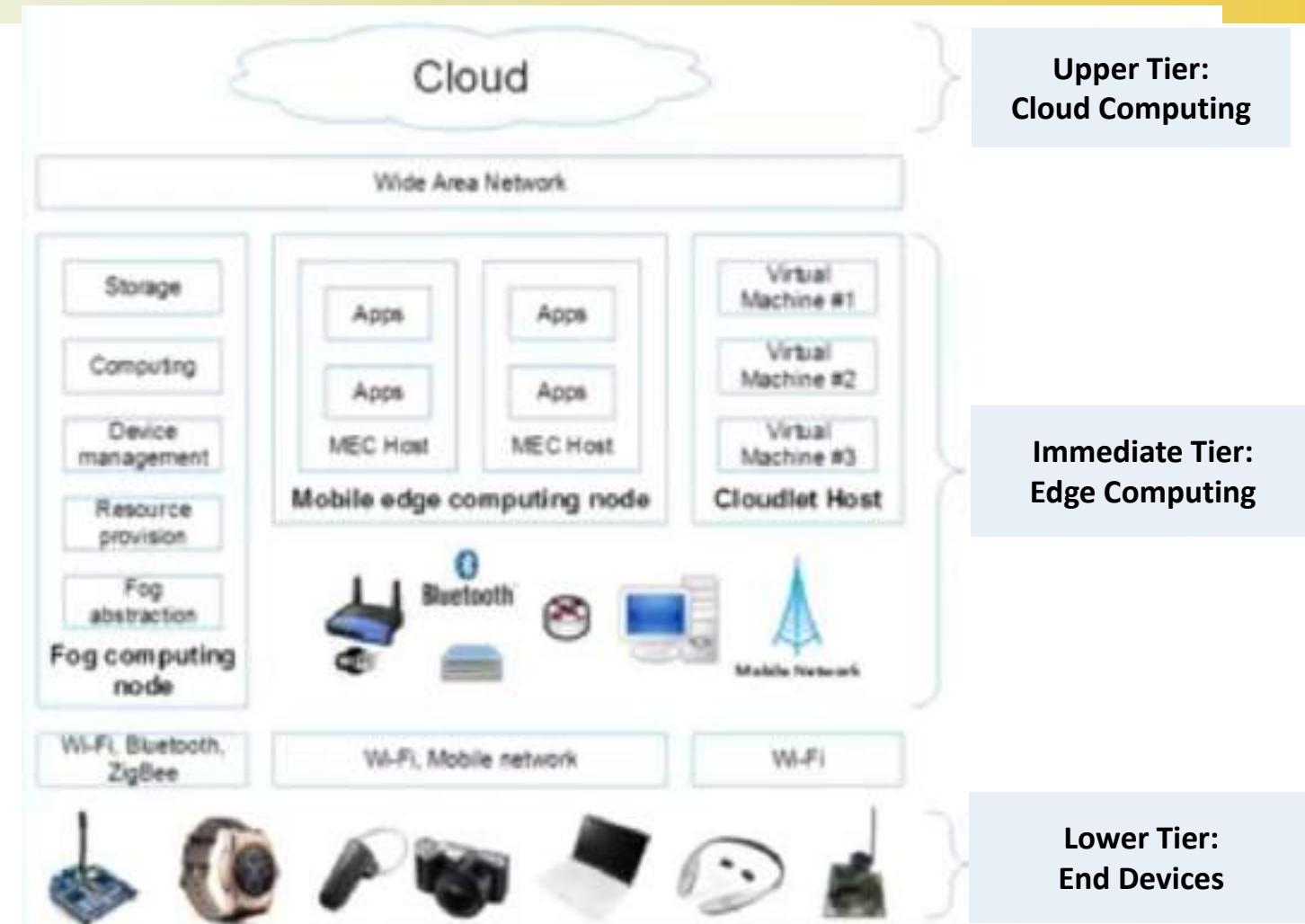
The Edge (Fog) of the Cloud is becoming a global platform for the computation and interaction between machines and smart objects, in real-time applications.



# Multi-Tier Computing



## Example 1

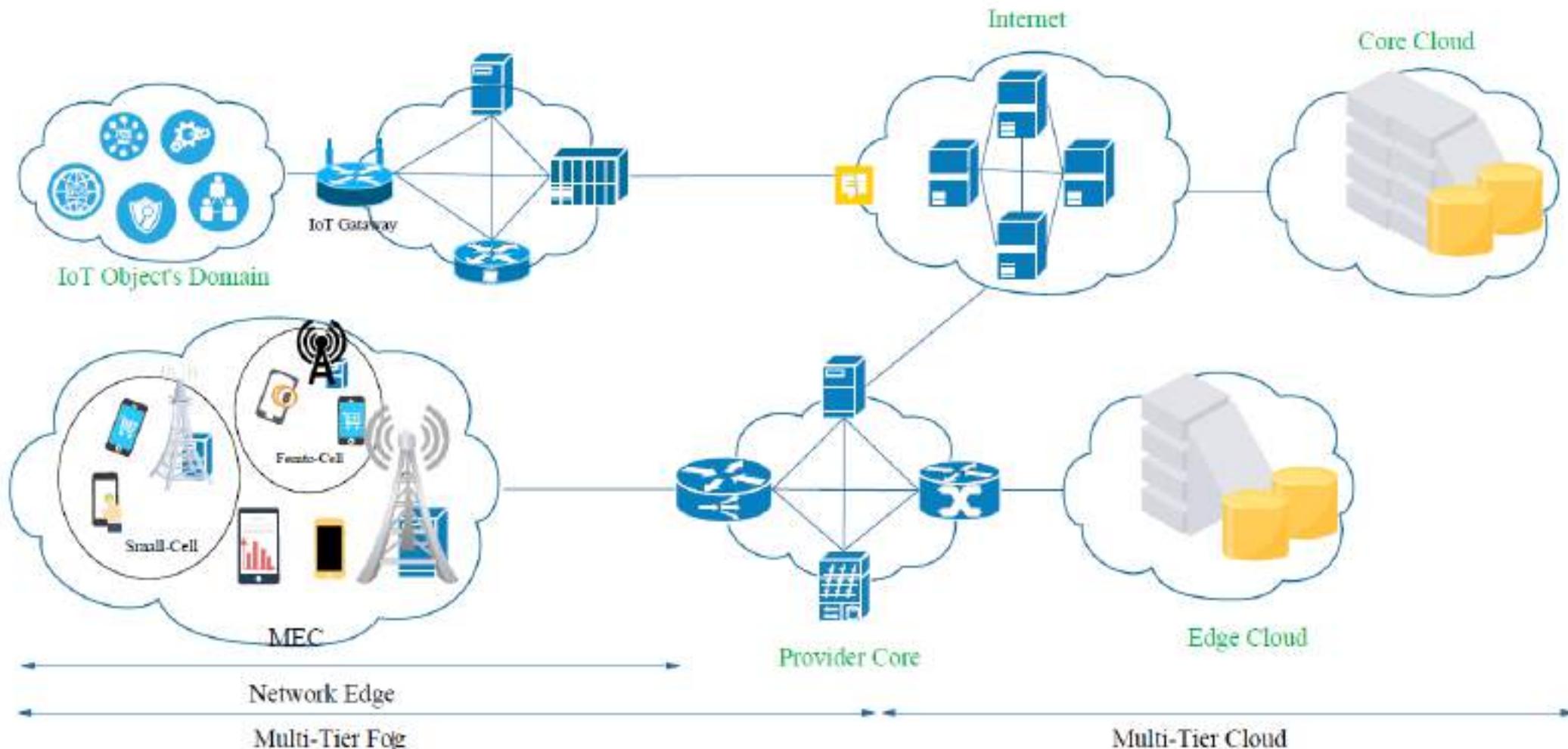


MEC: Multi-access or Mobile Edge Computing

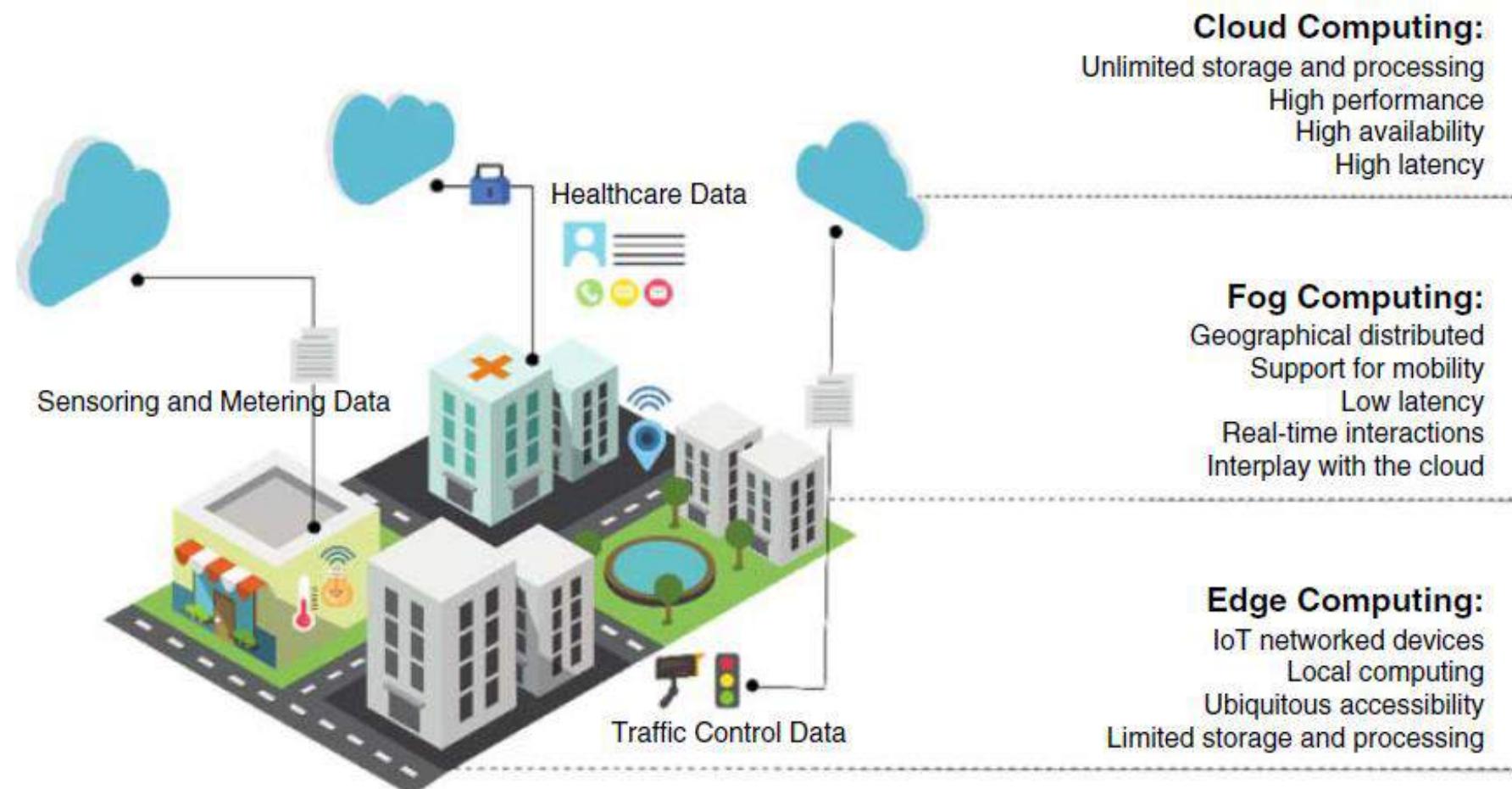
# Multi-Tier Computing



## Example 2



## Example 3



# Multi-Tier Computing



## Example 4

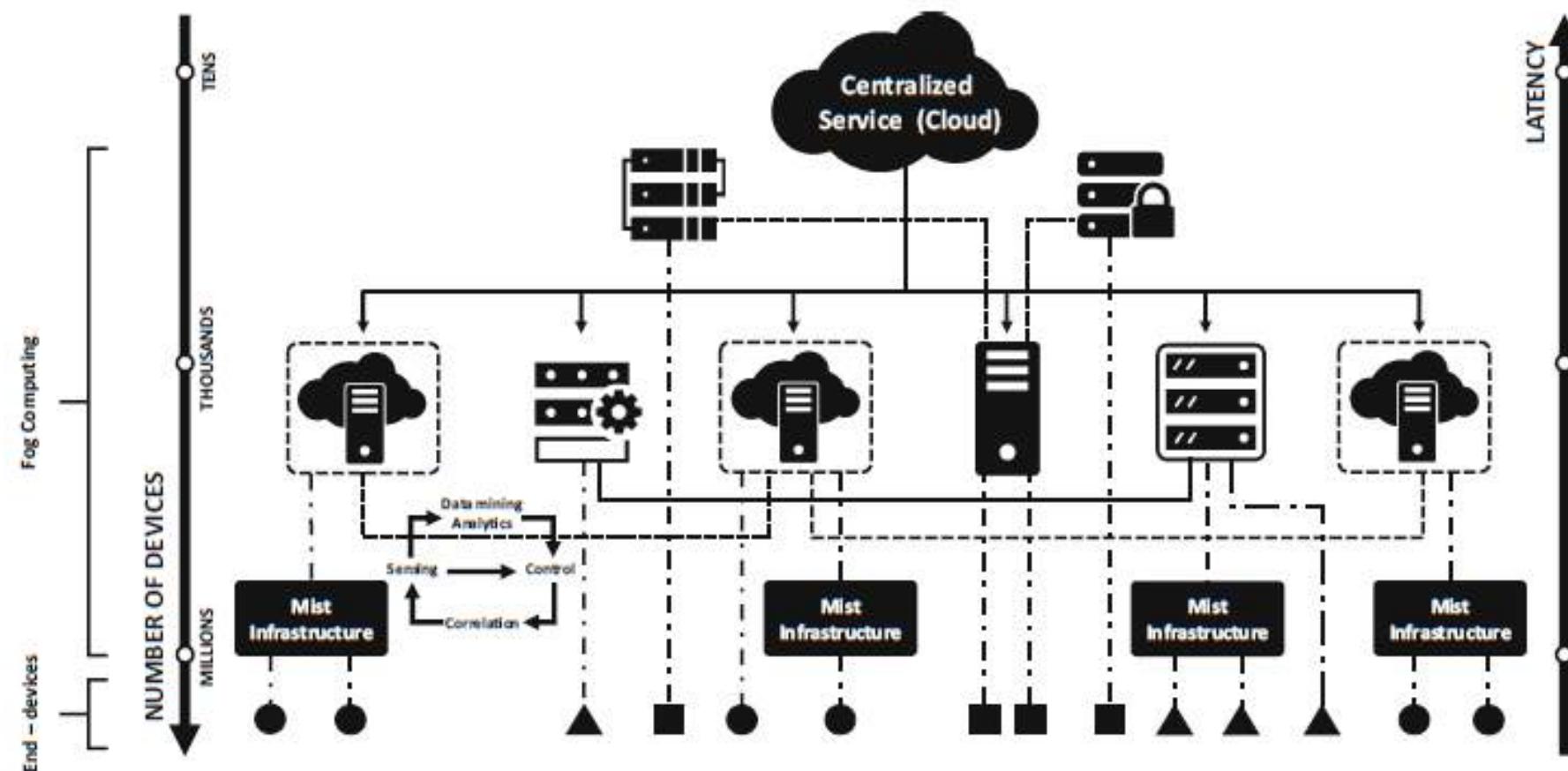
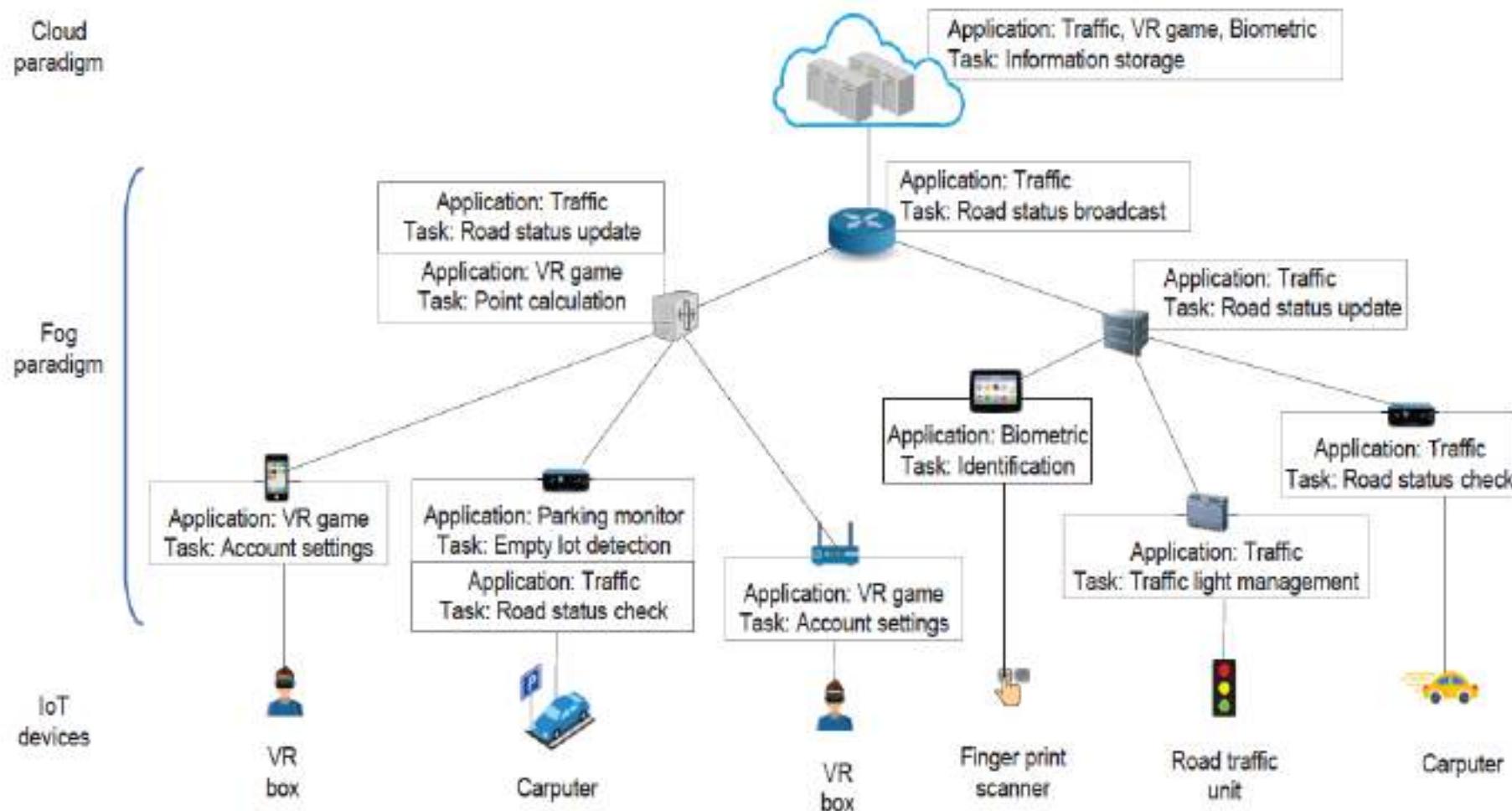


Fig. 8.1 Fog Computing Model (adapted from Iorga et al. (2017))

# Multi-Tier Computing



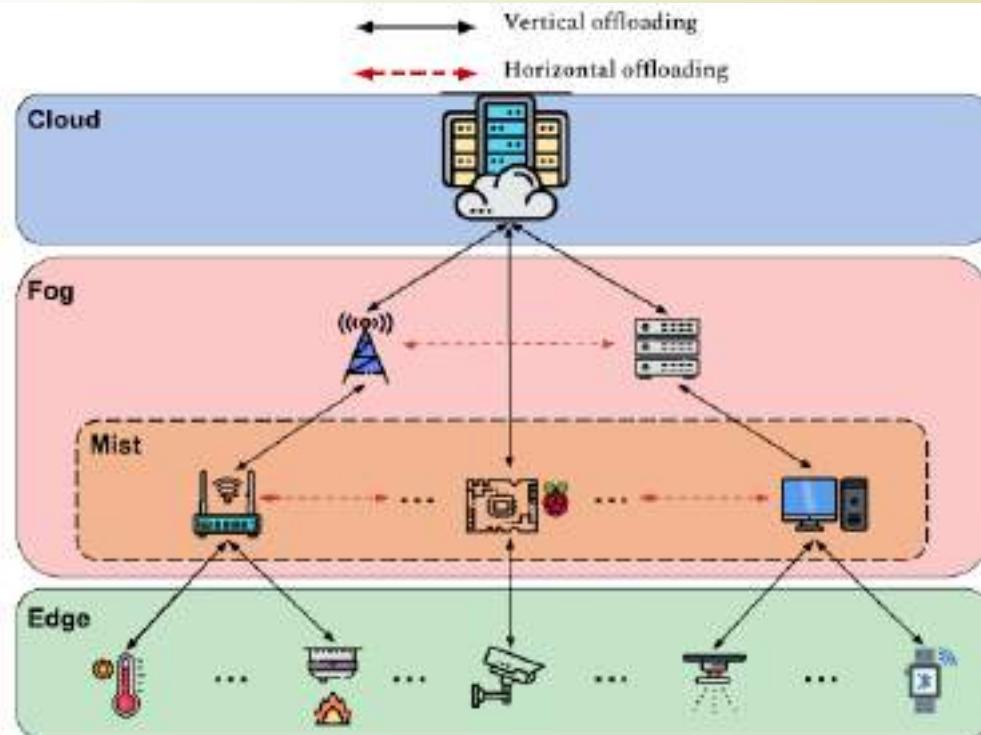
## Example 5



# Multi-Tier Computing

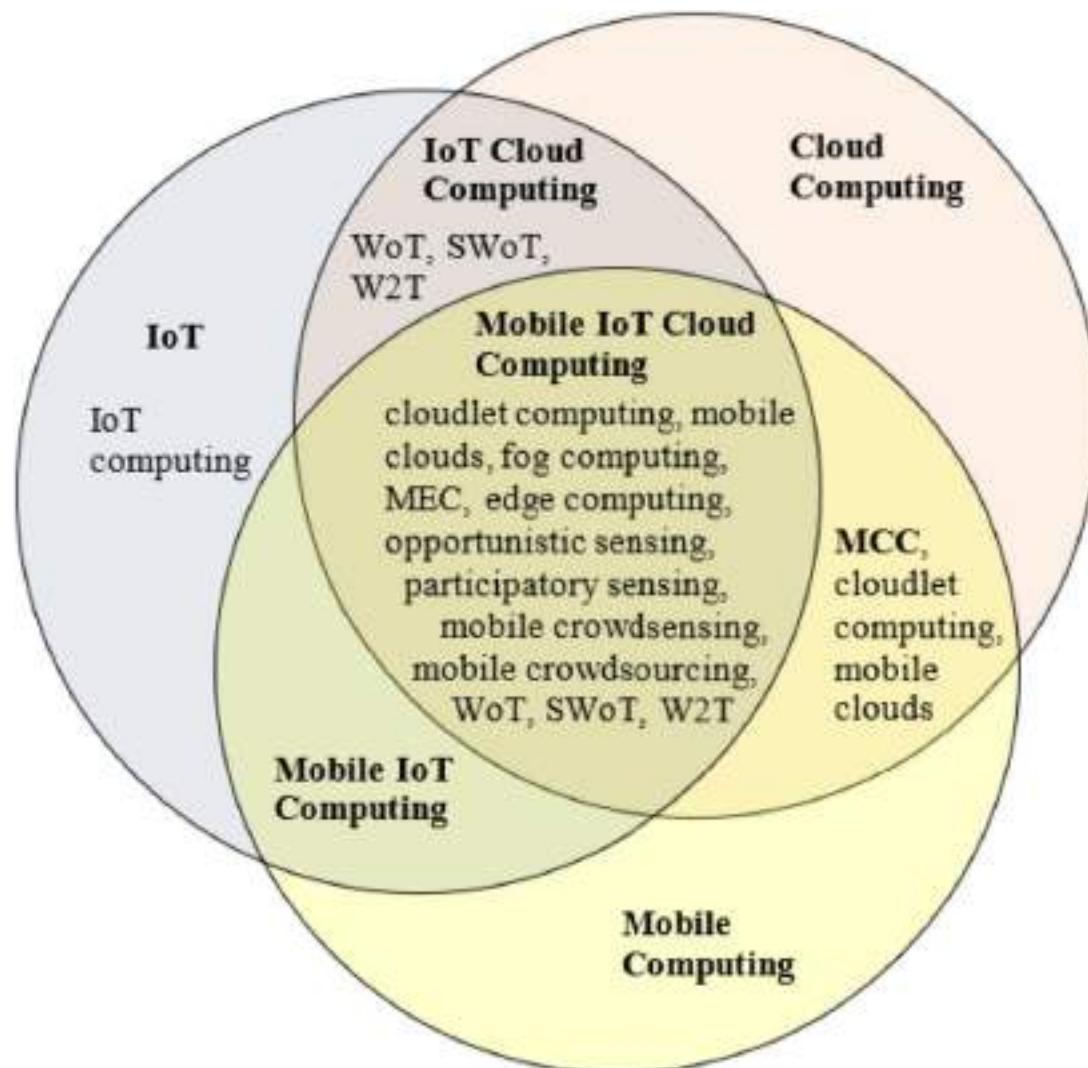


## Concept



- **Edge:** data processing is held immediately on the Edge IoT devices, reducing network latency and overheads.
- **Mist:** considers that the processing capabilities are close to data sources (e.g. access points, single-board computers). They can cooperate using clustering based on peer-to-peer communications.
- **Fog:** Compute resources are “relatively” close to data sources. Indeed, Fog nodes could be part of the access networks (e.g., servers, cloudlets, routers). Similarly to Mist nodes, Fog nodes can collaborate through clustering.
- **Cloud** provides huge computational/storage resources for the lower levels (Fog/Mist and Edge) when their capabilities are not sufficient..

# IoT Computing Paradigms





## Key Differences: Cloud Computing vs. Fog and Edge Computing

|                          | Cloud Computing                          | Fog and Edge Computing                  |
|--------------------------|------------------------------------------|-----------------------------------------|
| Architecture             | Centralized<br>(servers in data centers) | Distributed<br>(via fog or edge nodes)  |
| Location of server nodes | On the Internet (global)                 | At the edge of the network (local)      |
| Location awareness       | No                                       | Yes                                     |
| Latency                  | Medium                                   | Low                                     |
| Amount of nodes          | Few (large servers)                      | Millions (of fog or edge nodes)         |
| Support for mobility     | Limited                                  | Supported                               |
| Computing capabilities   | Higher                                   | Lower                                   |
| Analysis                 | Long-term/deep analysis                  | Short-term                              |
| Connection               | Internet                                 | Various protocols and standards         |
| Risk of failure          | Medium<br>(depending on the Internet)    | Low<br>(based on different connections) |
| Security                 | Medium                                   | High (due to distributed architecture)  |

## Advantages: SCALE

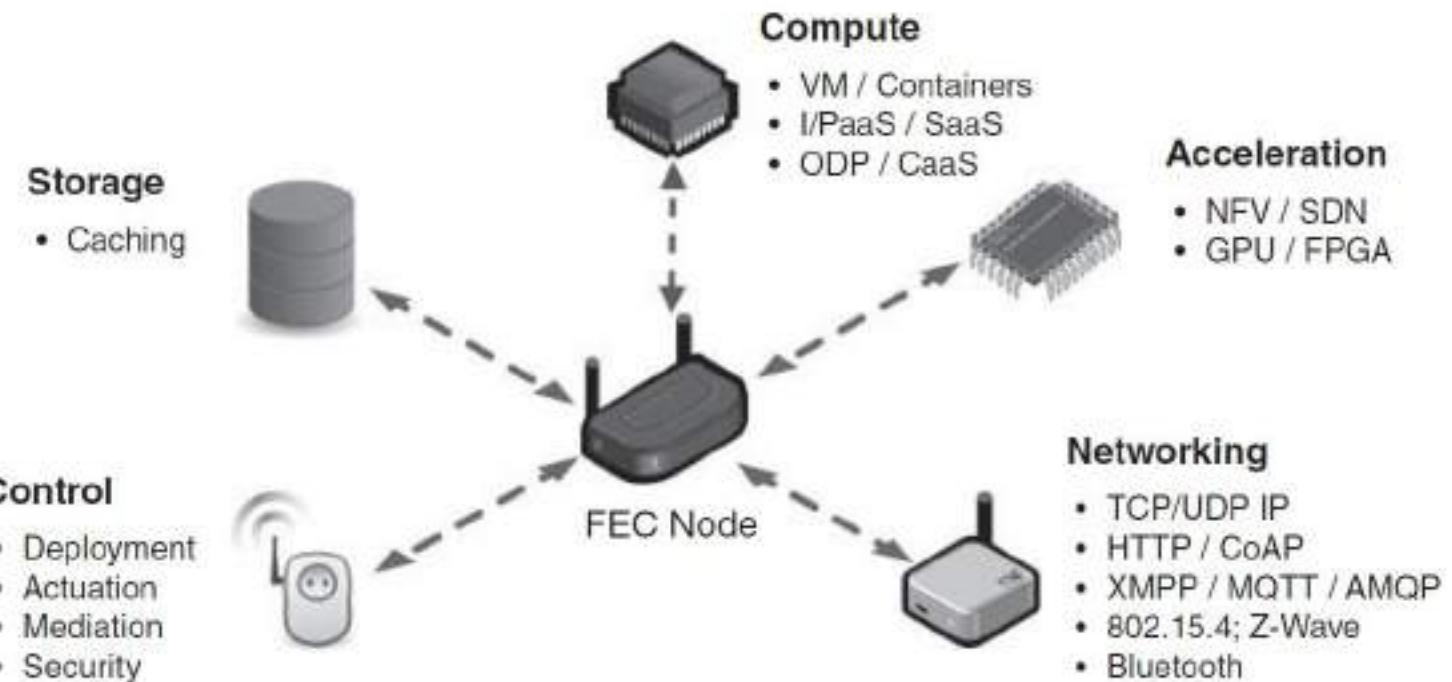
- Security
- Cognition
- Agility
- Latency
- Efficiency

# Fog and Edge Computing (FEC)



## To achieve SCALE with SCANC

- Storage
- Compute:
  - 2 Models: I/PaaS and SaaS
  - 2 Approaches:
    - Hypervisor VMs
    - Container engines
  - SaaS:
    - on-demand processing
    - Context as a service (CaaS)
- Acceleration:
  - Networking
  - Computing
- Networking:
  - Vertical
  - Horizontal (wireless domain)
- Control:
  - Deployment
  - Actuation
  - Mediation (interaction)
  - Security





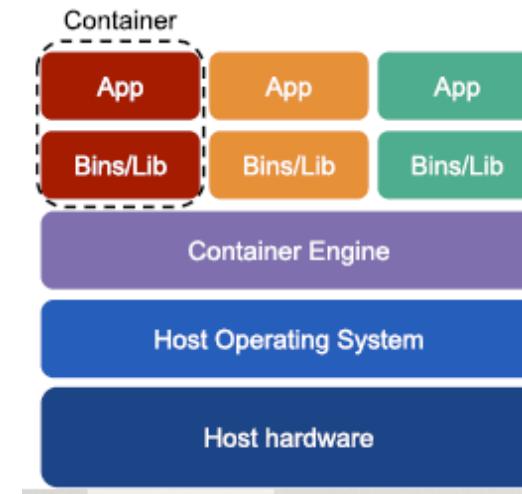
## Virtualization vs Containerization



Virtualization paradigm (i.e., hardware virtualization) is characterized by higher complexity compared to the former as it makes use of a fully abstracted environment including the emulation of low-level (software-based) hardware management.

Learn more:

<https://www.technicolor.com/sites/default/files/whitepapers/2018-edge-compute-and-software-life-cycle-management.pdf>

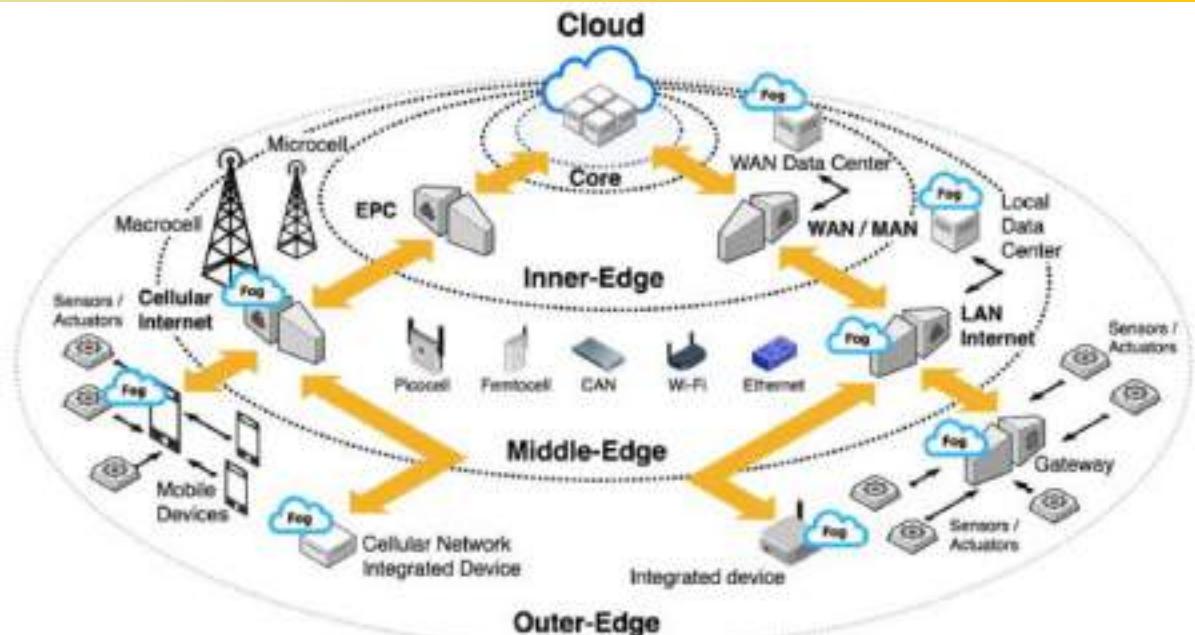


Containers or containerization paradigm, the abstraction of resources occurs at the host kernel whereas the hardware virtualization is below that level (i.e., the host kernel).

# Fog and Edge Computing (FEC)

## Hierarchy of FEC

- Inner-Edge: the WAN-based cloud data center
- Middle-Edge
  - Local Area Network
  - Cellular Network
- Outer-Edge
  - Constraint Devices
  - Integrated Devices
  - IP Gateway Devices



EPC: evolved packet core

Ref: Fog and Edge Computing: Principles and Paradigms

## Networking and Management Challenges

Edge refers to a collection of technologies as decentralizing data center resources for bringing computational resources closer to the end user.



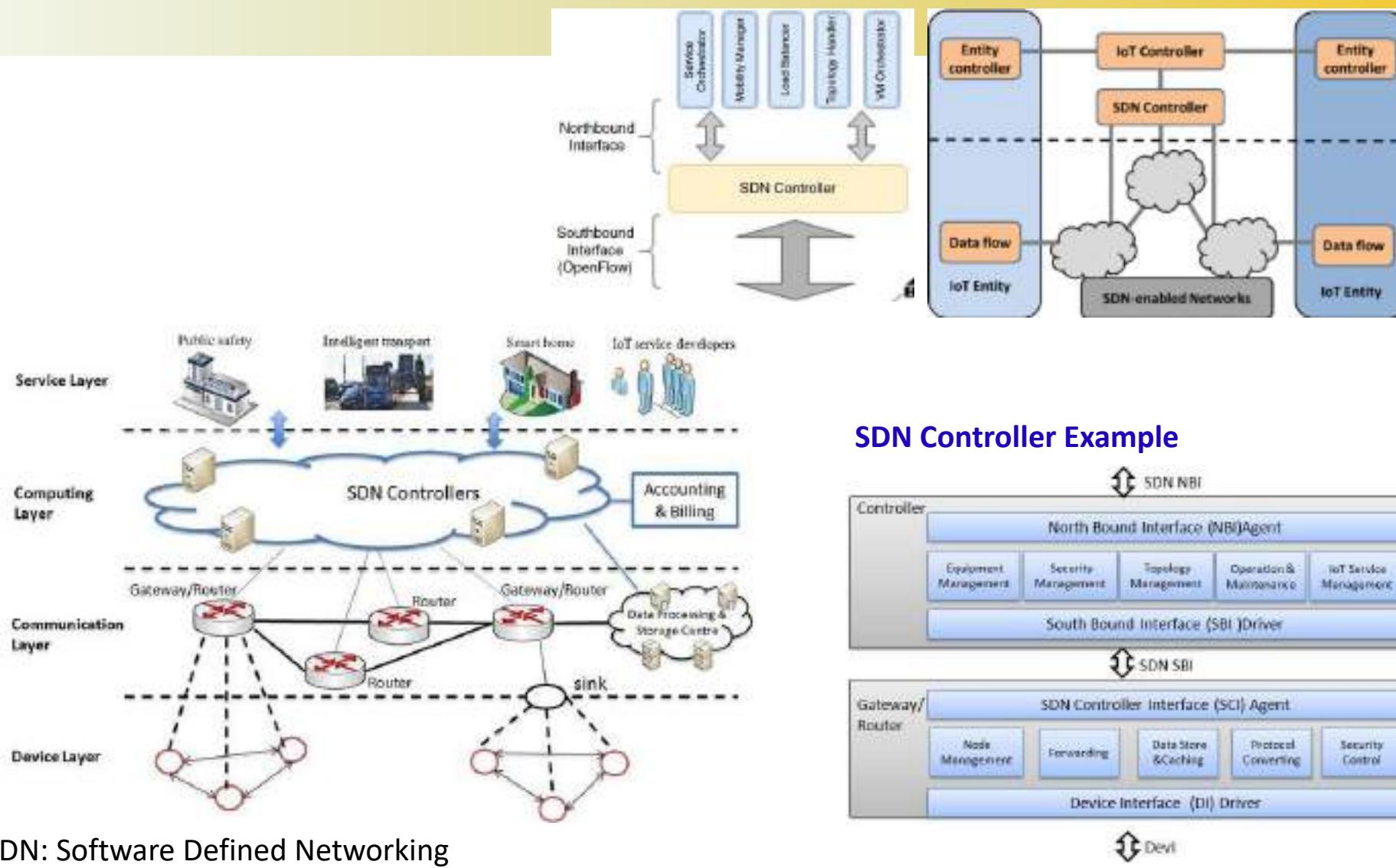
The Instances of Edge Computing:  
Mobile cloud computing (MCC),  
Cloudlets,  
Fog computing,  
Multi-access Edge Computing (MEC)

SDN: Software Defined Networking

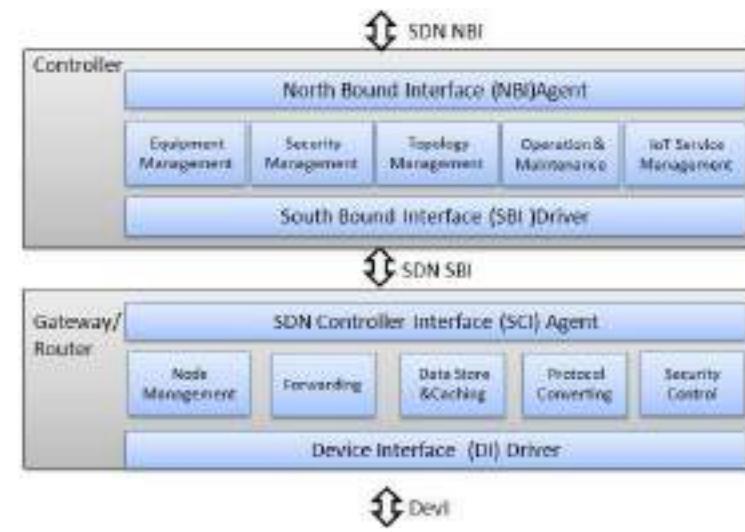
# Edge Resources



## SDN in IoT



### SDN Controller Example





## Networking Challenges

- We need to handle application layer handovers from one edge node to another.
- It's upon where the users are located and how request patterns are formed, the location of a service may change at any time under QoS satisfaction.

| Networking challenge                      | Why does it occur?                                                                           | What is required?                                                                                        |
|-------------------------------------------|----------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|
| User mobility                             | Keeping track of different mobility patterns                                                 | Mechanisms for application layer handover                                                                |
| QoS in a dynamic environment              | Latency-intolerant services, dynamic state of the network                                    | Reactive behavior of the network                                                                         |
| Achieving a service-centric model         | Enormous number of services with replications                                                | Network mechanisms focusing on "what" instead of "where"                                                 |
| Ensuring reliability and service mobility | Devices and nodes joining the network (or leaving)                                           | Frequent topology update, monitoring the servers and services                                            |
| Managing multiple administrative domains  | Heterogeneity, separate internal operations and characteristics, different service providers | Logically centralized, physically distributed control plane, vendor independency, global synchronization |

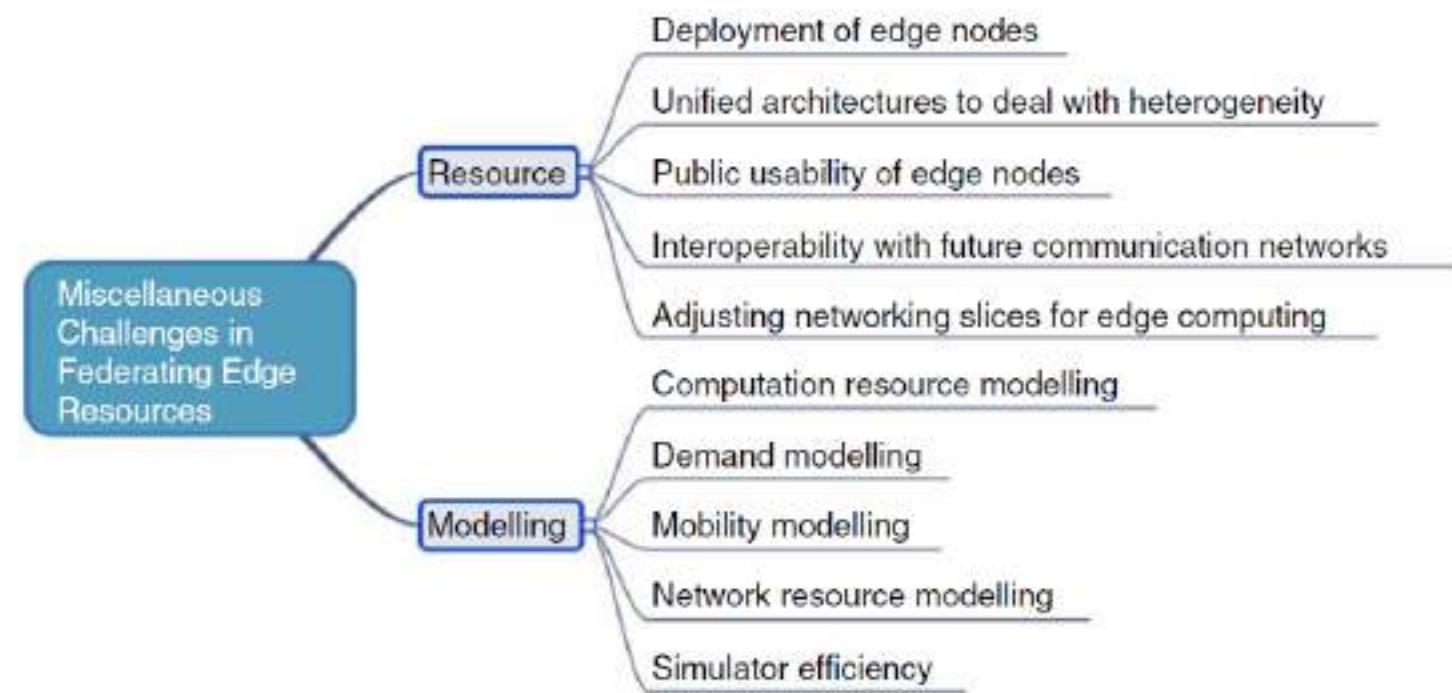


## Management Challenges

| Management challenge                   | Why should it be addressed?                                                 | What is required?                                   |
|----------------------------------------|-----------------------------------------------------------------------------|-----------------------------------------------------|
| Discovery of edge nodes                | To select resources when they are geographically spread and loosely coupled | Lightweight protocols and handshaking               |
| Deployment of service and applications | Provide isolation for multiple services and applications                    | Monitoring and benchmarking mechanisms in real-time |
| Migrating services                     | User mobility, workload balancing                                           | Low overhead virtualization                         |
| Load balancing                         | To avoid heavy subscription on individual nodes                             | Auto-scaling mechanisms                             |

## Research Challenges

- In order to support multiple use cases and demands, various communication and resource models, and heterogeneous environments.



See more details ch3 [BS19] for research topics in:  
energy consumption, performance, resource consumption, costs, QoS, Security



## Microsoft Azure IoT Edge

<https://azure.microsoft.com/en-us/services/iot-edge/>

## AWS Greengrass

<https://aws.amazon.com/greengrass/>

## IBM Watson IoT

<https://www.ibm.com/cloud/internet-of-things>

## Google

<https://cloud.google.com/iot-core/>



- [VB17] Ovidiu Vermesan and Joel Bacquet, “**Cognitive Hyperconnected Digital Transformation: Internet of Things Intelligence Evolution**”, River Publishers, 2017.
- [BHC+18] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, “**The Industrial Internet of Things (IIoT): An Analysis Framework**”, Computers in Industry 101, October 2018.
- [BS19] Rajkumar Buyya and Satish Narayana Srirama, “**Fog and Edge Computing: Principles and Paradigms**”, Wiley Series on Parallel and Distributed Computing, 2019.
- [LEA20] Perry Lea, “**IoT and Edge Computing for Architects: Implementing edge and IoT systems from sensors to clouds with communication systems, analytics, and security**”, 2<sup>nd</sup> Edition, Packt Publishing, 2020.

## Simplilearn

<https://www.youtube.com/watch?v=KeaeuUcw02Q>

## Google

<https://www.youtube.com/watch?v=-T9MNR-BI8I>

## Azure IoT Edge:

<https://www.youtube.com/watch?v=01kHKNN3z-0>

<https://www.youtube.com/watch?v=rjfCNakISBM> (Implement)

## Eseye

<https://www.youtube.com/watch?v=knQbAEhipxM> (IoT Ecosystem)

## Atos

<https://www.youtube.com/watch?v=cxFJWxQRRDY> (IoT Ecosystem Security)

## Oracle:

<https://www.youtube.com/watch?v=HoGV35vpTEY>

## NGIOT:

<https://horizon-europe-cloud-edge-iot.b2match.io/>  
<https://www.ngiot.eu/from-cloud-to-edge-to-iot-for-european-data/>

[https://www.youtube.com/playlist?list=PLBrivHE6\\_rsdWf9F9KN-MyFZ9b57RjcDm](https://www.youtube.com/playlist?list=PLBrivHE6_rsdWf9F9KN-MyFZ9b57RjcDm)



# More References: IoT + ML

- Creating a Machine Learning IoT application on Raspberry Pi with Node-RED and TensorFlow.js  
<https://www.youtube.com/watch?v=6sFrQaDtK5Q>
- IoT Edge examples  
<https://adv-corp-test.azurewebsites.net/resources/?solution=iot-edge-intelligence-software-solutions>
- TinyML study group  
<https://github.com/scaledown-team/study-group>
- IBM  
<https://developer.ibm.com/technologies/iot/tutorials/iot-cognitive-iot-app-machine-learning/>
- From Cloud to Edge to IoT  
[https://www.ngiot.eu/wp-content/uploads/sites/73/2021/03/NGIOT-Workshop\\_25Feb2021-RRiemenschneider.pdf](https://www.ngiot.eu/wp-content/uploads/sites/73/2021/03/NGIOT-Workshop_25Feb2021-RRiemenschneider.pdf)
- AI + Sensor  
<https://www.electropages.com/blog/2021/02/how-ai-taking-sensors>
- Edge Intelligence  
<https://rvce.edu.in/sites/default/files/RVJSTEAM-ISSUE1.pdf>
- Computing at the Edge of IoT – Google Developers – Medium  
<https://www.epanorama.net/blog/2018/02/18/computing-at-the-edge-of-iot-google-developers-medium/>



# Internet of Things & NETPIE2020 - Workshop

[For 29 Oct'21]

**Mr. Piyawat Jomsathan [Tae]**  
**Cyber-Physical Systems [CPS]**  
**National Electronics and Computer Technology Center [NECTEC], Thailand**

# Download Document & Software

Link --> <https://bit.ly/3BXJzeh>



# Main Topic

1

What is IoT Platform and NETPIE ?

2

Structure of NETPIE2020

3

Type of Communication in NETPIE2020

4

Data Management in NETPIE2020

5

Freeboard in NETPIE2020

6

Workshop



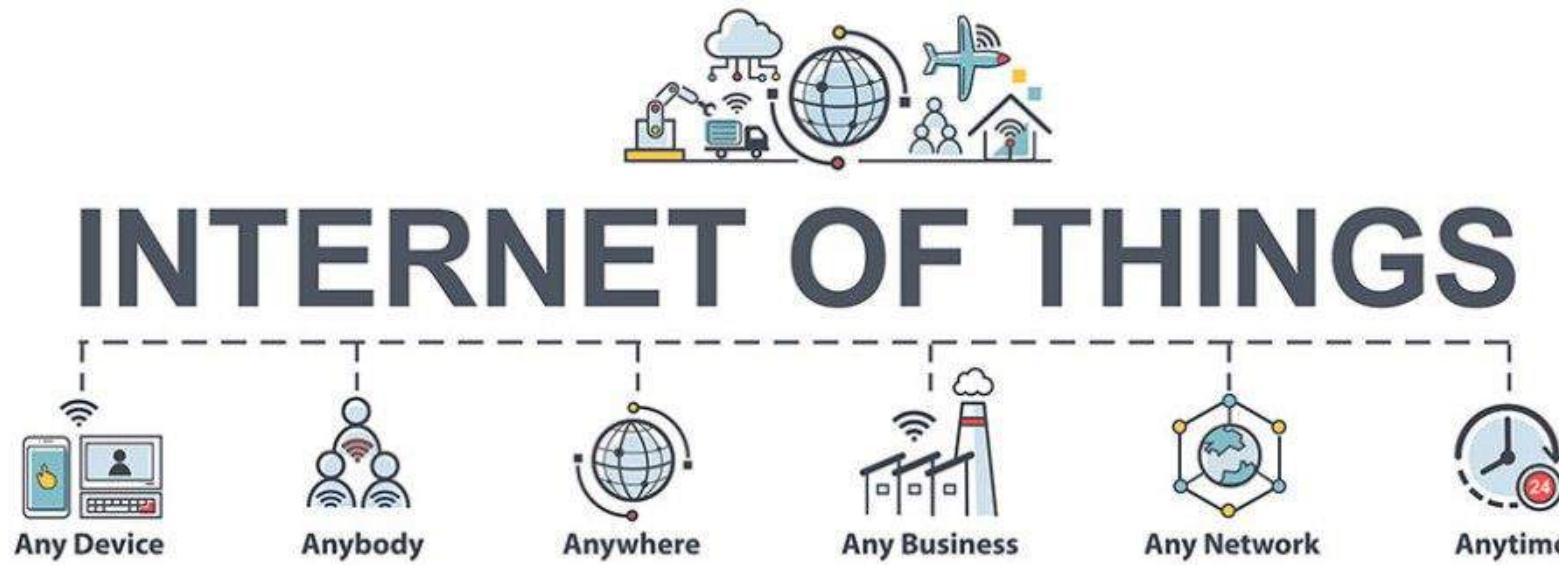
1

# What is IoT Platform and NETPIE ?



# 1 - What is IoT Platform and NETPIE ?

---



# What is IoT??

# 1 - What is IoT Platform and NETPIE ?

## What is IoT??

IoT (Internet of Things) refers to how things are connected to the Internet, enabling humans to control or command various devices via the Internet, such as turning electrical equipments on/off, operating communication devices, modern office, agricultural and industrial machines, and other appliances in our daily life.



# 1 - What is IoT Platform and NETPIE ?

---

**What is IoT Platform??**

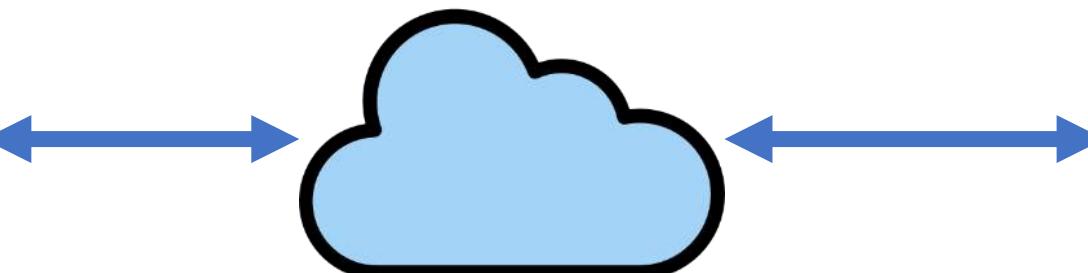
# 1 - What is IoT Platform and NETPIE ?

## Simple IoT example

What we want from a rice cooker

1. temperature
2. Control [ON/OFF]

Temperature can be monitored via  
Mobile phone



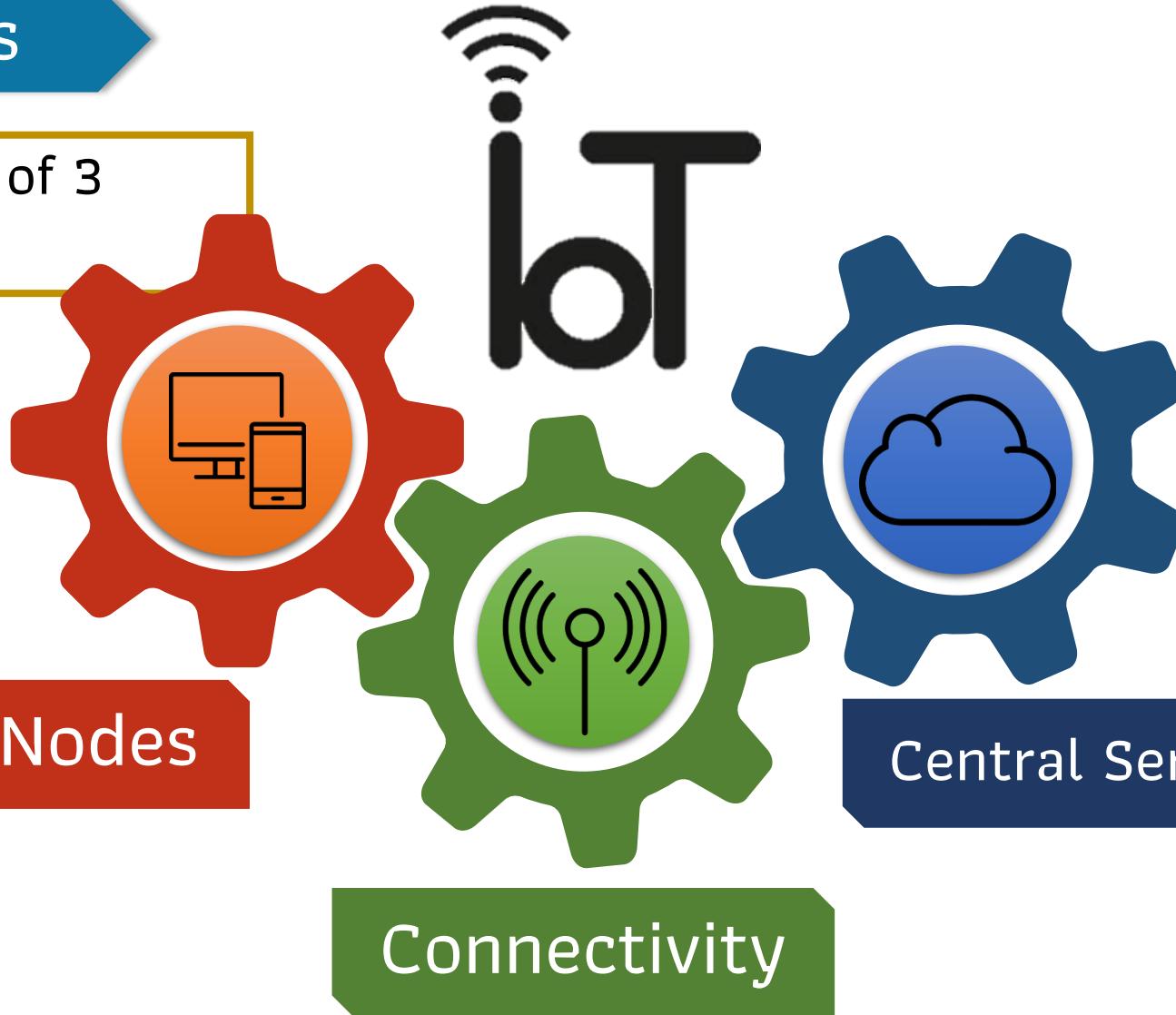
Data sent to the cloud via internet



# 1 - What is IoT Platform and NETPIE ?

## IoT Components

An IoT system consists of 3 important components



# 1 - What is IoT Platform and NETPIE ?

## IoT components

### End Nodes



End nodes are parts of objects (bulbs, motors, sensors) and devices that are used to control things or detect and measure parameters of interest such as temperature, humidity, lighting, etc. What are the things for IoT? However, it is only necessary to install an embedded device in order to connect to the Internet. With the currently popular devices including Microcontroller, Single Board, etc.

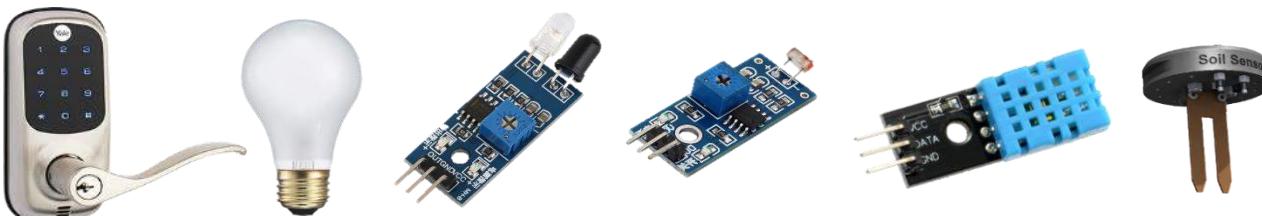
#### Microcontrollers



#### Single Board



#### Actuators and Sensors



# 1 - What is IoT Platform and NETPIE ?

## IoT Components

### Connectivity



Connectivity is a system that allows devices and things to connect to the Internet. To send and receive data between devices to the Cloud, there are various types of Internet connection systems, either wired or wireless depending on the suitability of use.



# 1 - What is IoT Platform and NETPIE ?

## IoT Components

### Central Servers/Cloud



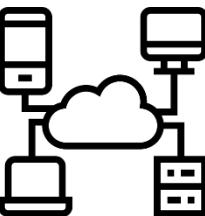
Central Servers / Cloud is a storage area to be used to create a Web Server, Application Server, etc. At present, for IoT systems, the cloud is used as a storage area. The differences between Server and Cloud are

#### Server

- :::::
- :::::
- :::::

Server is a machine or computer program that provides services in the network to clients. The computers serving as this server should be high-performance, stable, able to serve a large number of users.

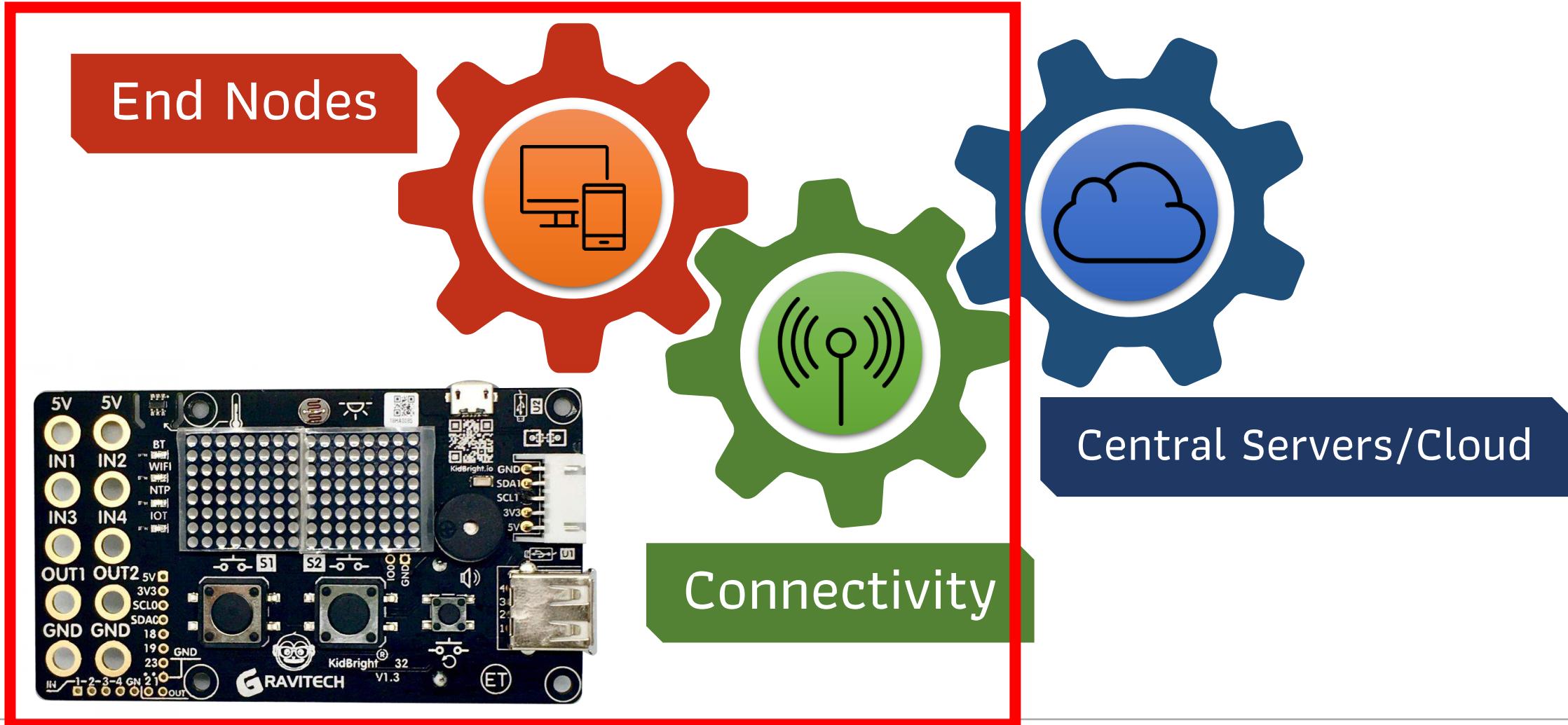
#### Cloud



Cloud is to bring multiple servers to work together with a high level of computing power. Cloud can build services to perform a variety of tasks.

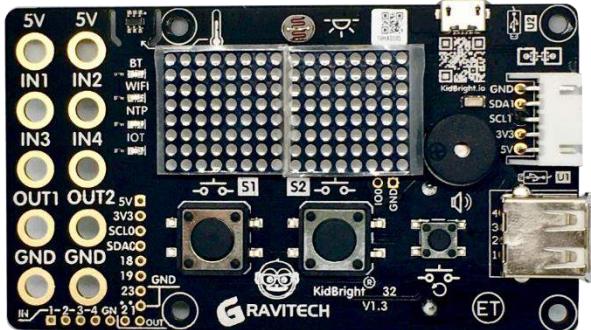
# 1 - What is IoT Platform and NETPIE ?

## IoT Components



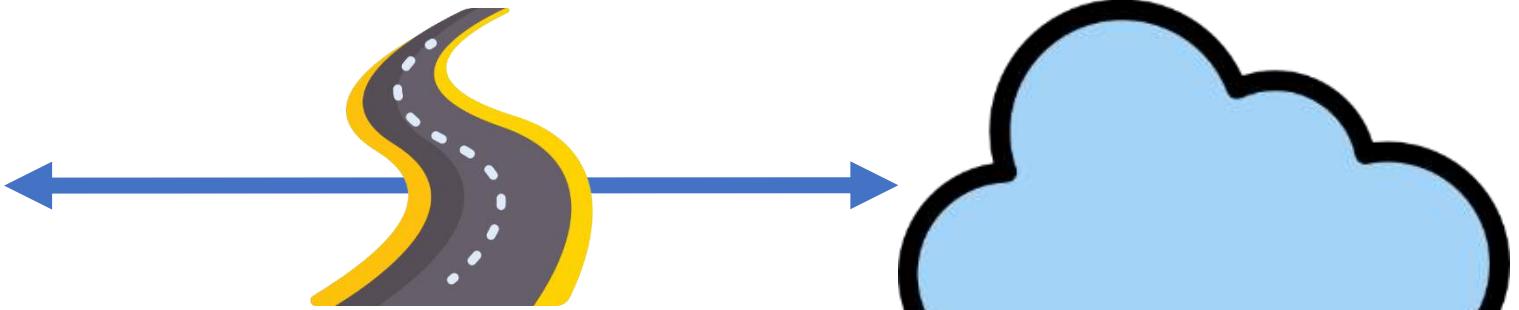
# 1 - What is IoT Platform and NETPIE ?

## IoT Components



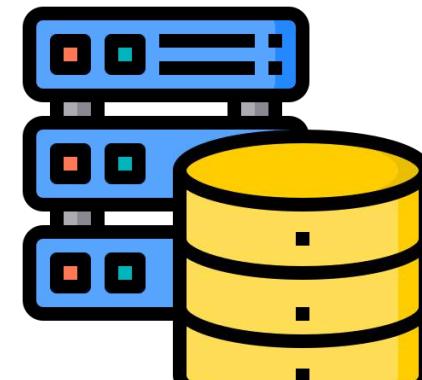
KidBright

Data that KidBright wants to send to Cloud  
"Temperature" = 25 °C  
"Light" = 80 Lux  
"LED Status" = "OFF"



Way of transmitting  
data / path of  
transmission

Cloud

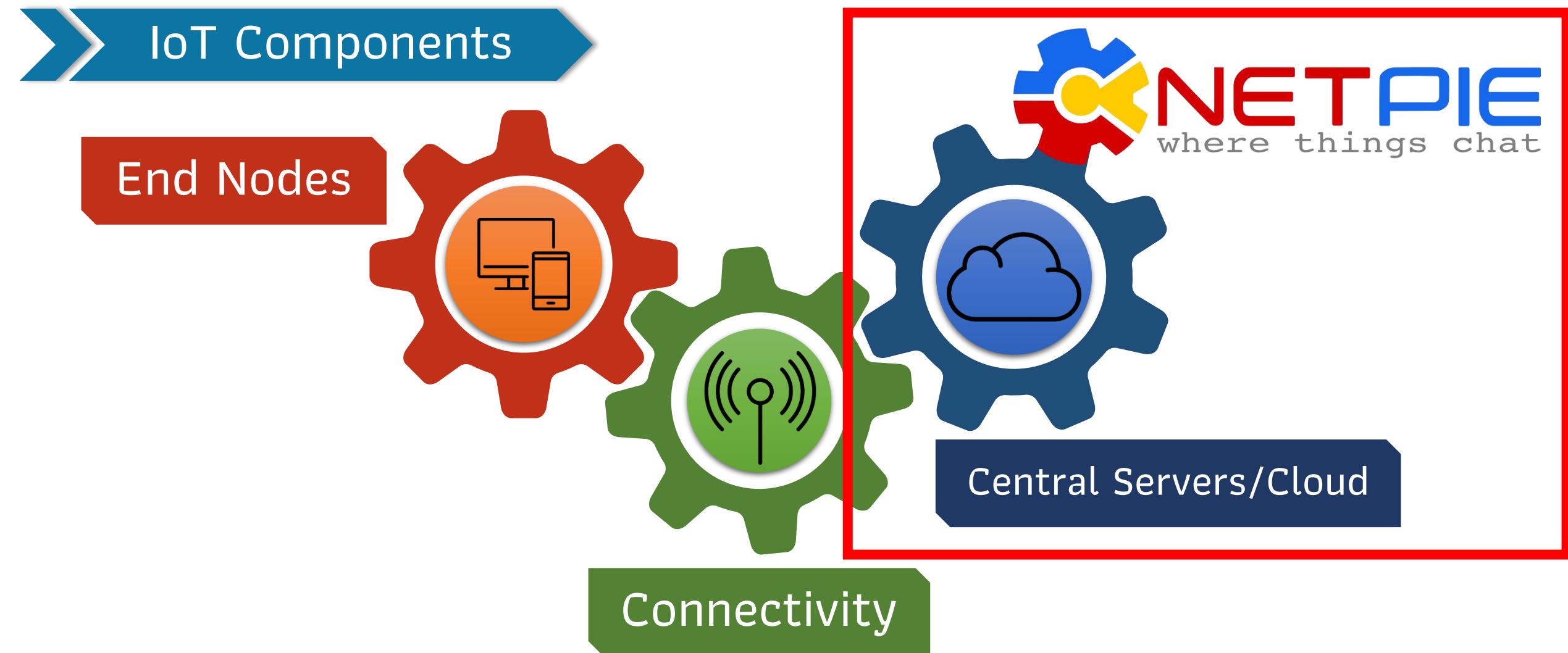


Data Storage



Dashboard IoT

# 1 - What is IoT Platform and NETPIE ?



# 1 - What is IoT Platform and NETPIE ?

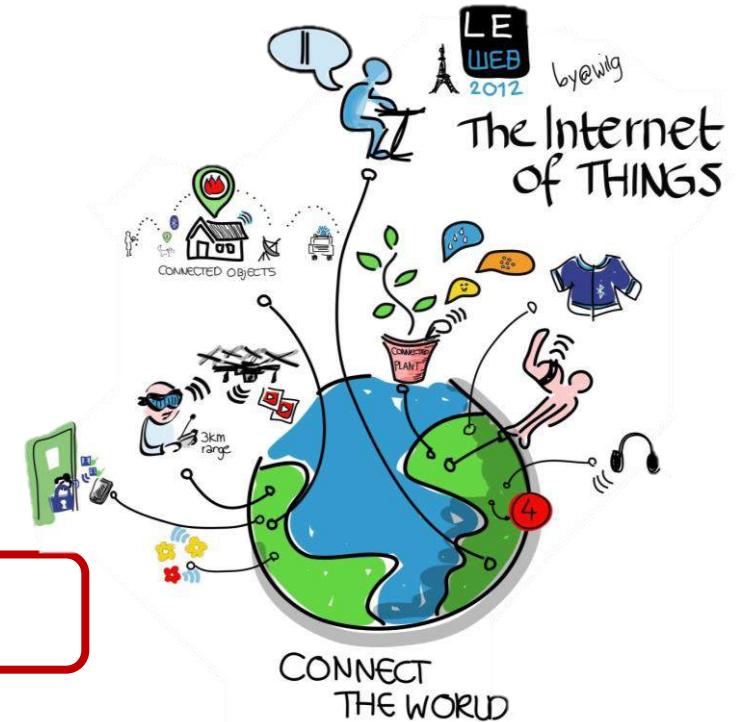


## What is NETPIE?

NETPIE is an IoT cloud platform that is open to the public. The platform will help devices able to communicate with each other, receive - transmit data between devices in real-time, allowing users to know the information of the device at that time, no matter where the user is.

## Main features of NETPIE

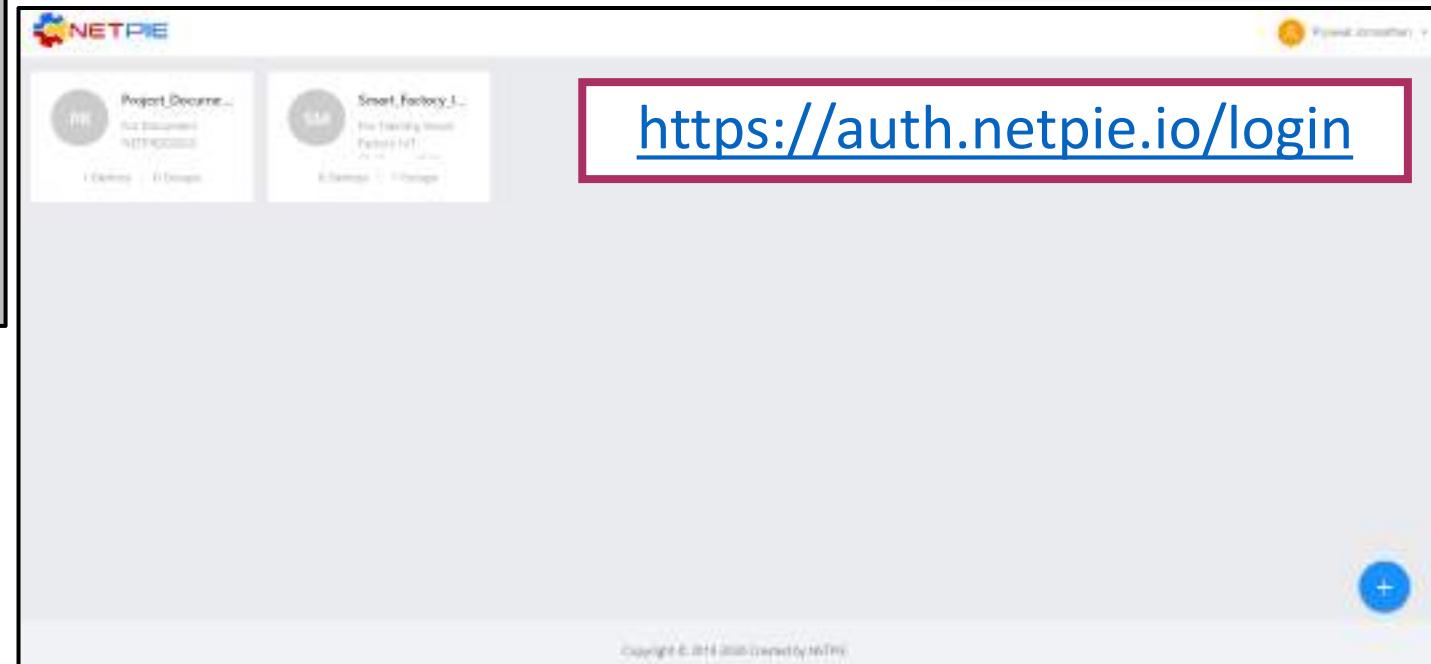
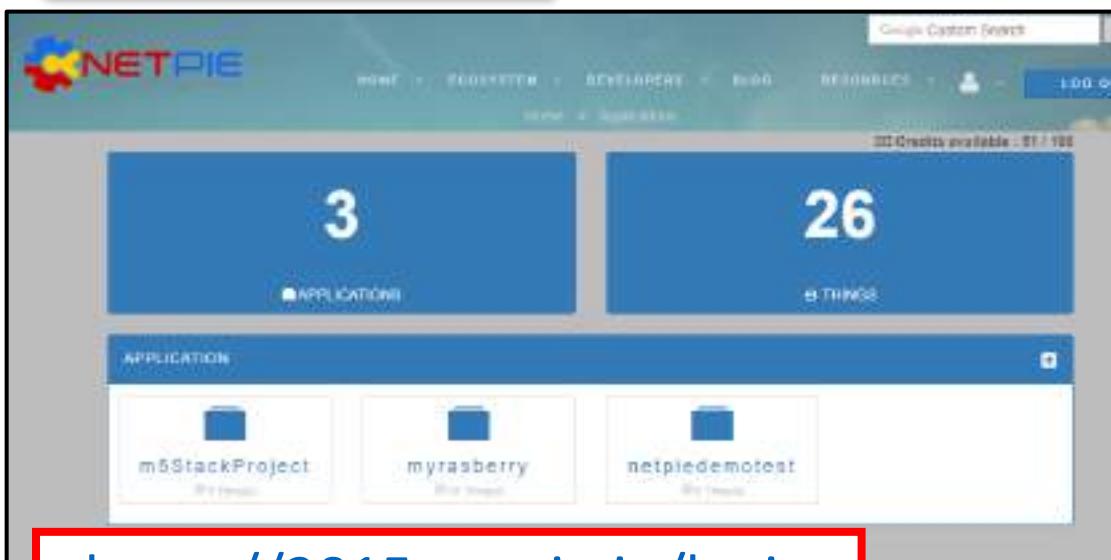
1. Monitoring : to display device or sensor data in real time
2. Controlling : to control the operation of devices via the cloud platform
3. Data Storage : to collect data from sensors or devices
4. Notification : to alert the user when sensor malfunction occurs



# 1 - What is IoT Platform and NETPIE ?

NETPIE 2015

At present, NETPIE is developed in 2 versions  
NETPIE 2015 and NETPIE 2020



# 1 - What is IoT Platform and NETPIE ?

## NETPIE 2020 and NETPIE 2015 Comparison

|                           | NETPIE 2020                                                                        | NETPIE 2015                                                          |
|---------------------------|------------------------------------------------------------------------------------|----------------------------------------------------------------------|
| Design Philosophy         | Platform - Centric                                                                 | Device – Centric                                                     |
| Commercial Ability        | Commercial – Ready                                                                 | Need to re-program Hardware migrate to commercial platform           |
| Suitable Usage            | Mass production, Project-based                                                     | Project-based                                                        |
| Target User               | IoT consumer product makers, Hobbyists, Students                                   | Makers, hobbyists, students                                          |
| Communication Protocol    | MQTT, HTTP                                                                         | Microgear                                                            |
| Programming Language      | Any languages with MQTT library support                                            | Limited to Microgear library                                         |
| Hardware Support          | Unlimited as long as it supports MQTT                                              | Limited to those with Microgear support                              |
| Device Identity and Group | No APPID Device identity and group can be adjusted after product is sold/installed | Use APPID Device identity and group must be programmed into firmware |
| Rate Limit                | Allow burst                                                                        | Everyone is subject to the same rate limit                           |
| Trigger                   | Can set trigger action in cloud platform                                           | Set trigger action inside IoT devices                                |

2

# Structure of NETPIE2020



## 2 - Structure of NETPIE2020

### Getting started with NETPIE2020

Create NETPIE2020 user account at  
<https://auth.netpie.io/signup>



The form consists of several input fields and a sign-up button. At the top right is the NETPIE 2020 logo. Below it are five input fields: 'EMAIL' (with a required indicator), 'NAME' (with a required indicator), 'ORGANIZATION' (with a required indicator), 'COUNTRY CODE' (set to Thailand (+66)), and 'MOBILE PHONE NUMBER\*' (NO COUNTRY CODE) (with a required and number only indicator). Below these fields is a checkbox labeled 'I agree to the Privacy Statement and Terms of Use'. At the bottom is a large red 'SIGN UP' button. A note at the bottom states: '\*Password will be sent to your mobile phone number.'

EMAIL

NAME

ORGANIZATION

COUNTRY CODE

Thailand (+66)

MOBILE PHONE NUMBER\* (NO COUNTRY CODE)

I agree to the Privacy Statement and Terms of Use

SIGN UP

\*Password will be sent to your mobile phone number.

## 2 - Structure of NETPIE2020

Getting started with NETPIE2020

After sign up NETPIE2020, login  
With Username [email] and Password

Or login at this link

<https://auth.netpie.io/login>



Connect Everything

Username (Email Address)

Password

SIGN IN

FORGET PASSWORD?

SIGN UP

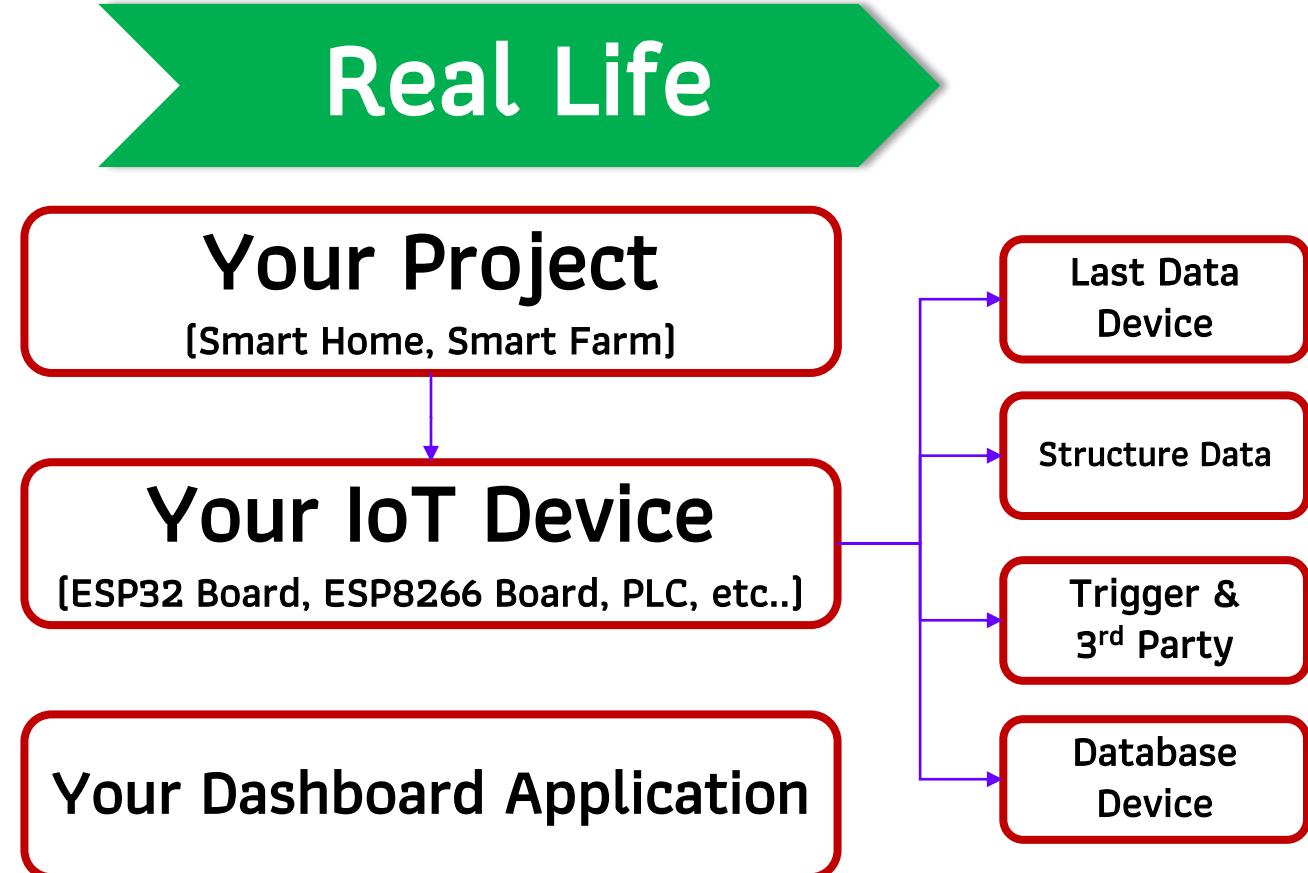
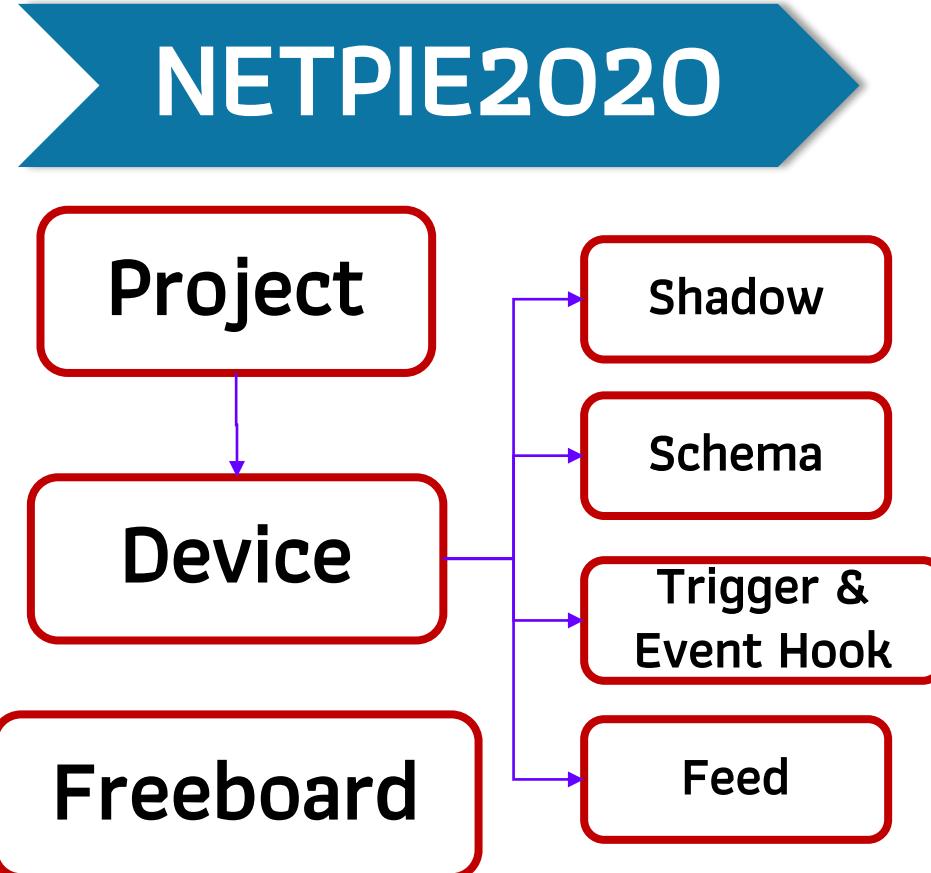
# 2 - Structure of NETPIE2020

## Getting started with NETPIE2020



## 2 - Structure of NETPIE2020

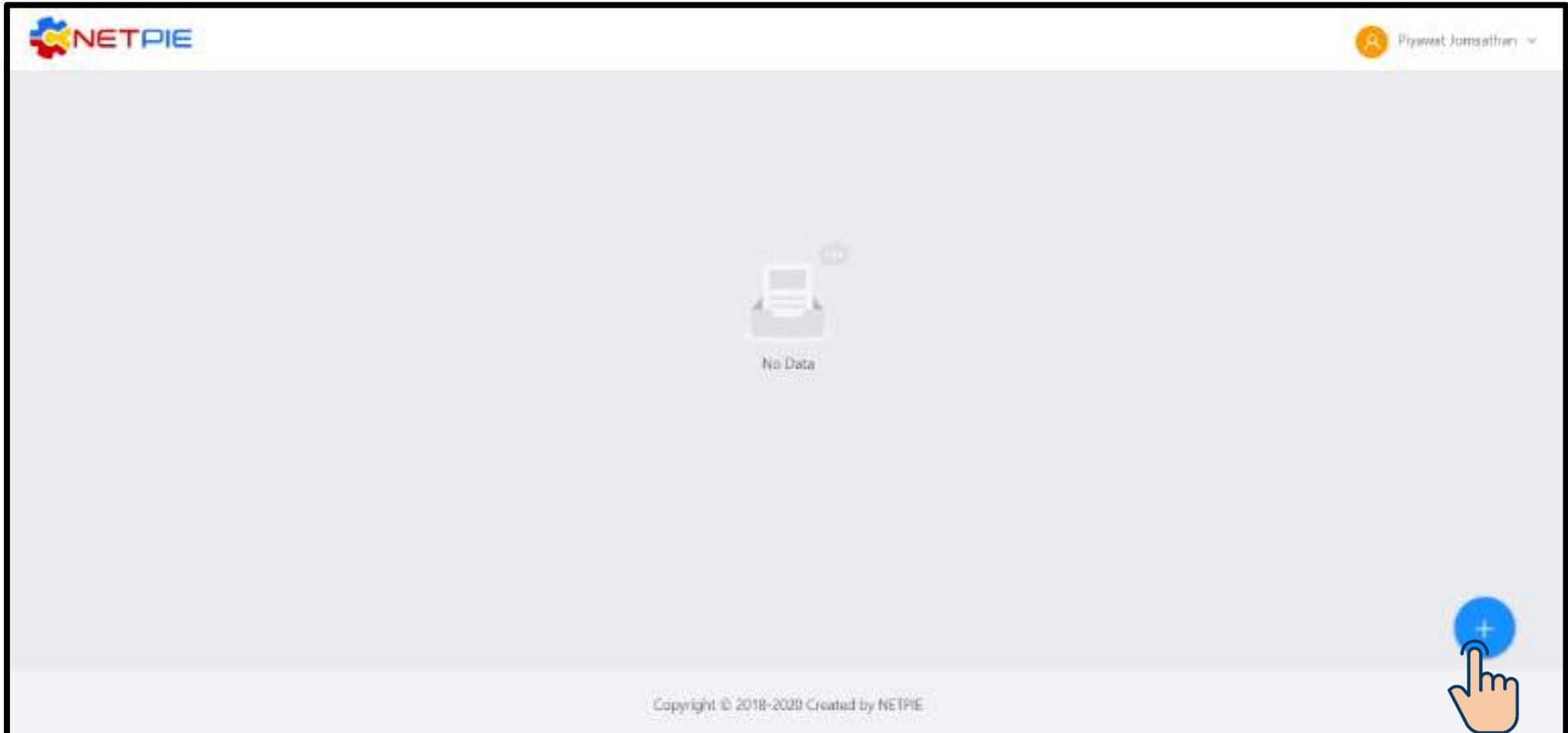
### Structure of NETPIE2020



## 2 - Structure of NETPIE2020

Create a Project

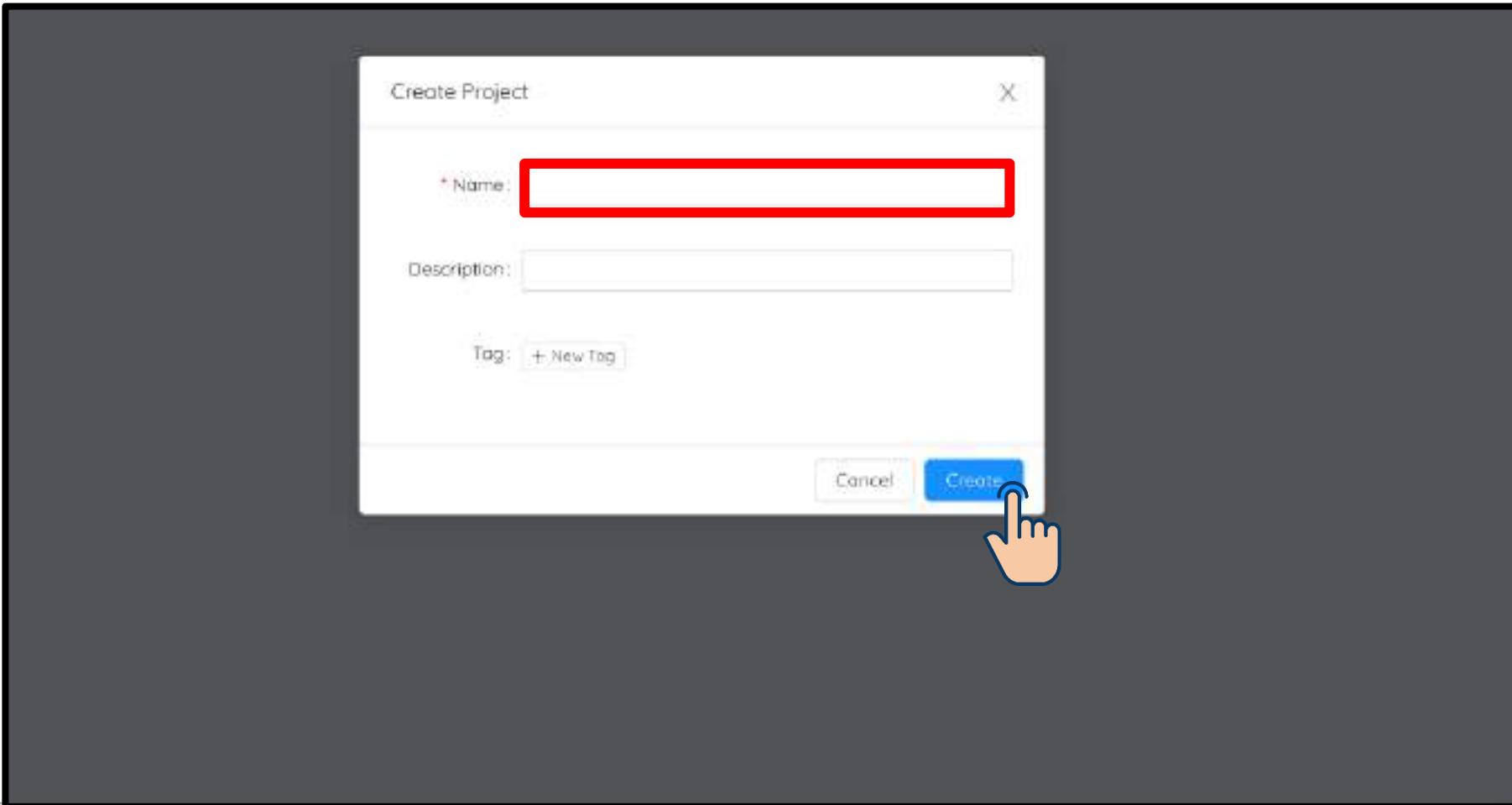
To create a project, click at [+] as shown in the picture



## 2 - Structure of NETPIE2020

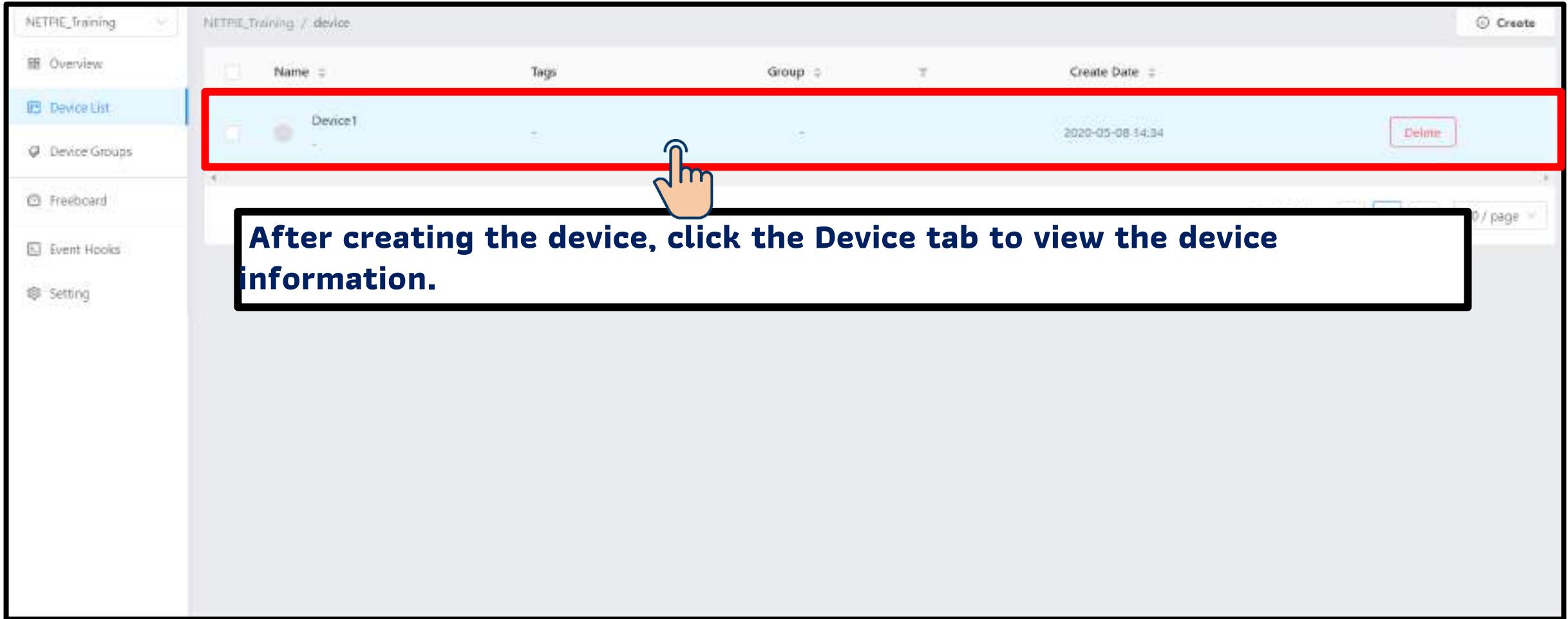
Create a Project

Fill out the information for creating the project. The field marked with \* is required.



# 2 - Structure of NETPIE2020

## Device Creation



NETPIE\_Training / device

Overview Device List Device Groups Freeboard Event Hooks Setting

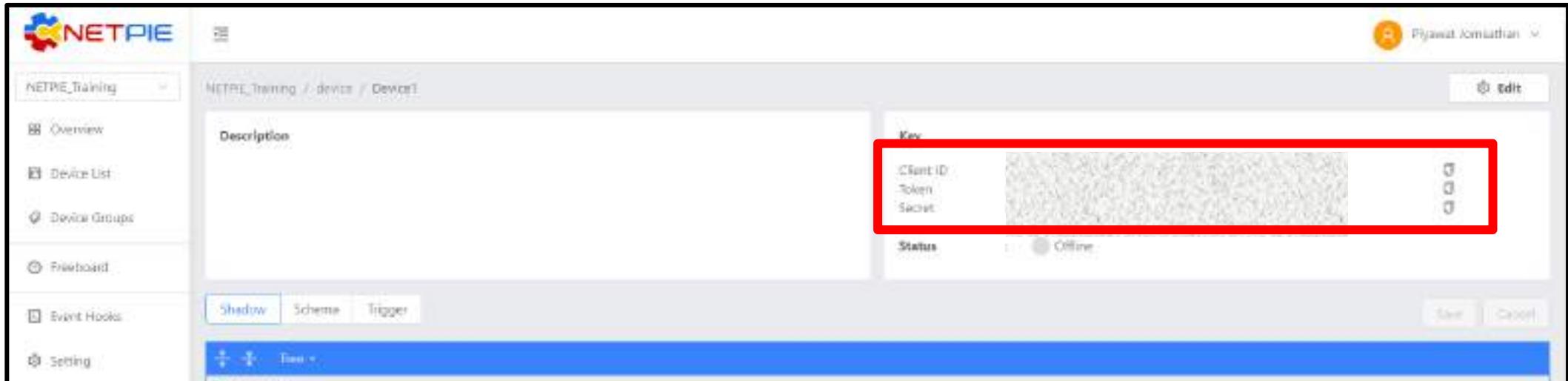
Name Tags Group Create Date

Device1 2020-05-08 14:34 Define

After creating the device, click the Device tab to view the device information.

## 2 - Structure of NETPIE2020

### Device Creation



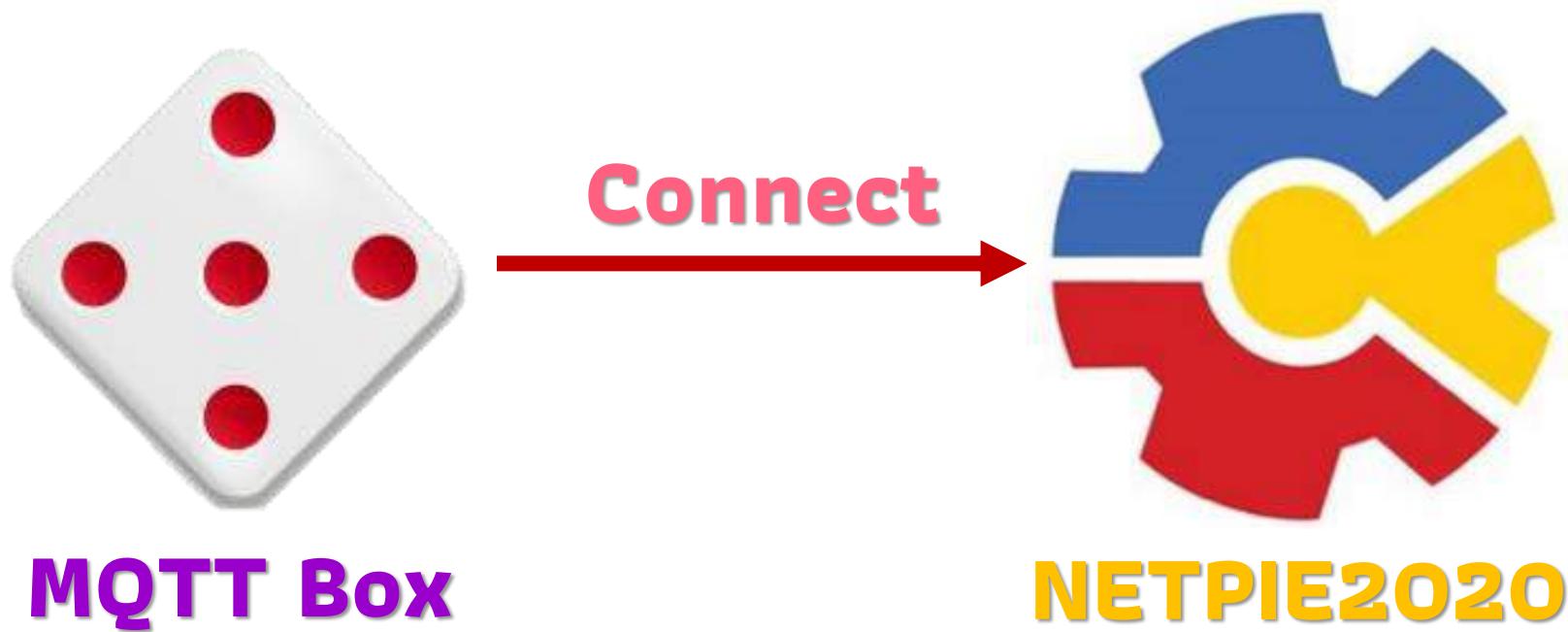
After entering the Device, you will find the details of the Device, that is

1. Client ID
2. Token
3. Secret

These 3 parameters are important to connect the device to NETPIE2020 using various protocols.

## 2 - Structure of NETPIE2020

**Exercise 1 : Connecting the NETPIE2020 with MQTT Box**

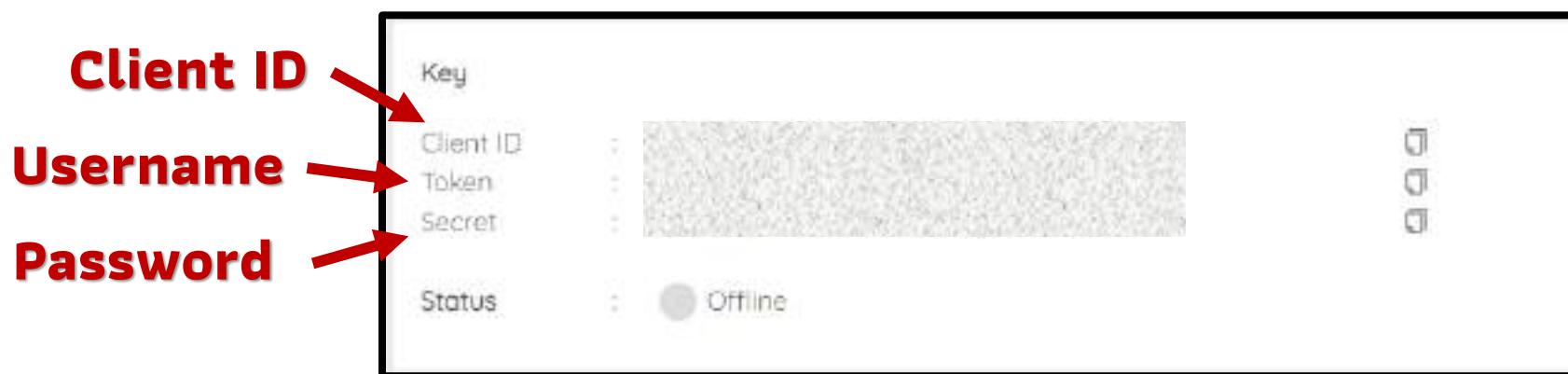


## 2 - Structure of NETPIE2020

### Exercise 1 : Connecting the NETPIE2020 with MQTT Box

**NETPIE2020 connection requires 4 parameters as follows:**

1. Host : broker.netpie.io
2. Client ID : Client ID of Device created in NETPIE2020 portal.
3. Username : Token of Device created in NETPIE2020 portal.
4. Password : Secret of Device created n NETPIE2020 portal.  
[used for added security]



## 2 - Structure of NETPIE2020

### Exercise 1 : Connecting the NETPIE2020 with MQTT Box

#### 1. Launch MQTTBox



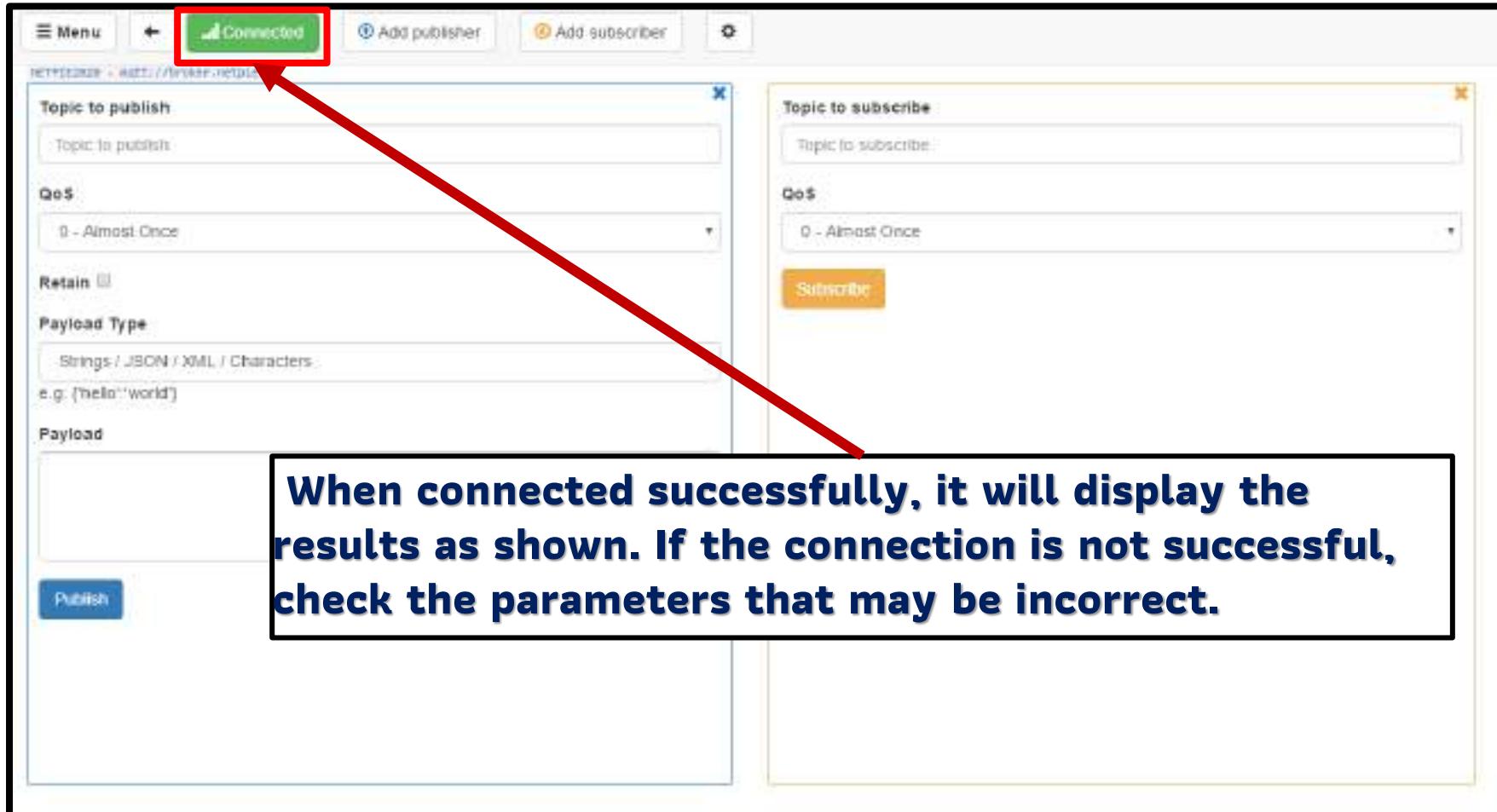
## 2 - Structure of NETPIE2020

### Exercise 1 : Connecting the NETPIE2020 with MQTT Box



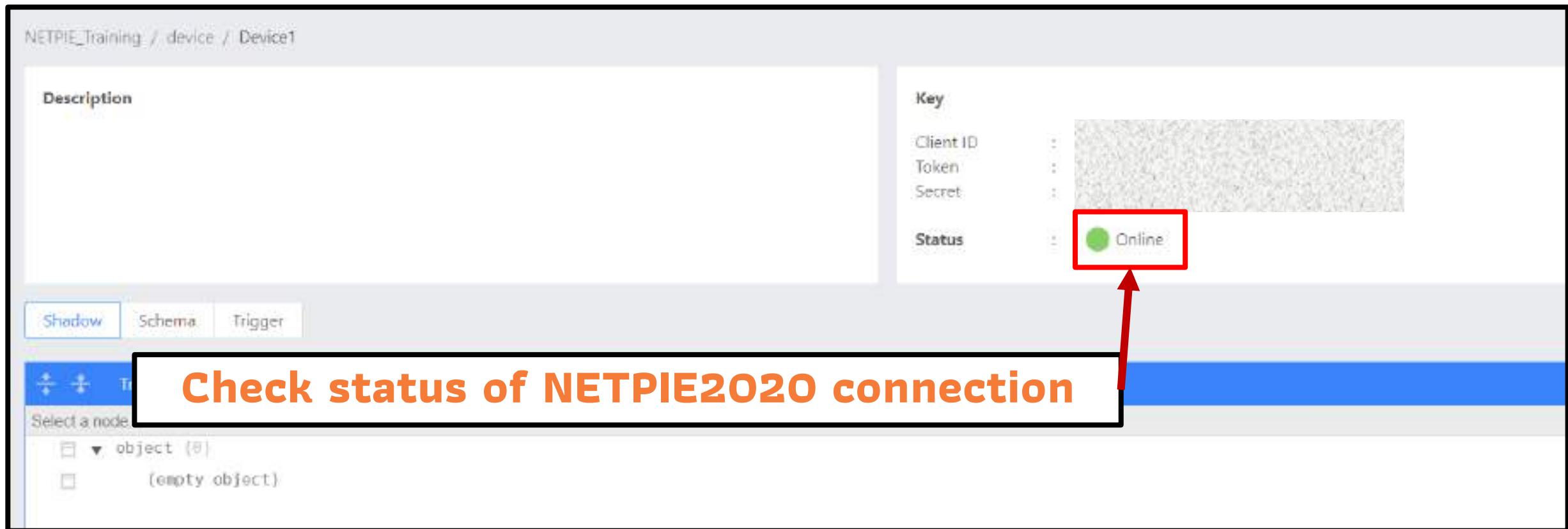
## 2 - Structure of NETPIE2020

### Exercise 1 : Connecting the NETPIE2020 with MQTT Box



## 2 - Structure of NETPIE2020

### Exercise 1 : Connecting the NETPIE2020 with MQTT Box



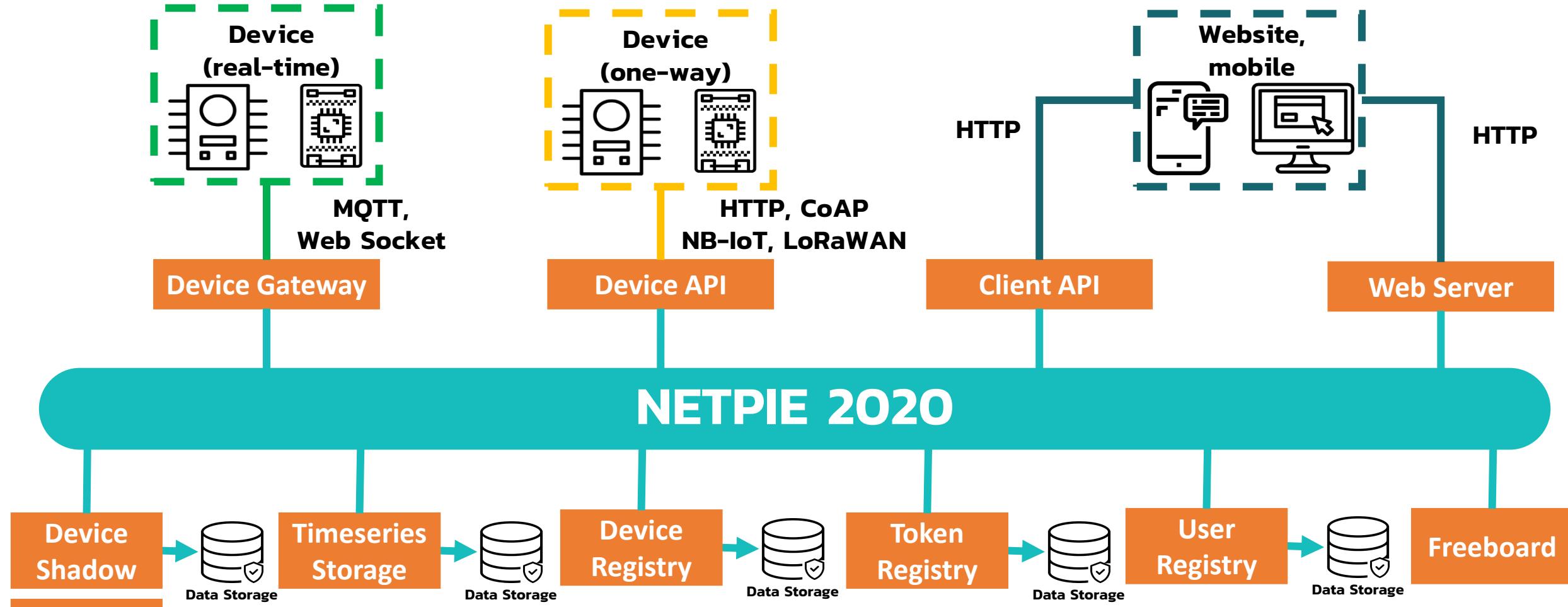
3

## Type of Communication in NETPIE2020

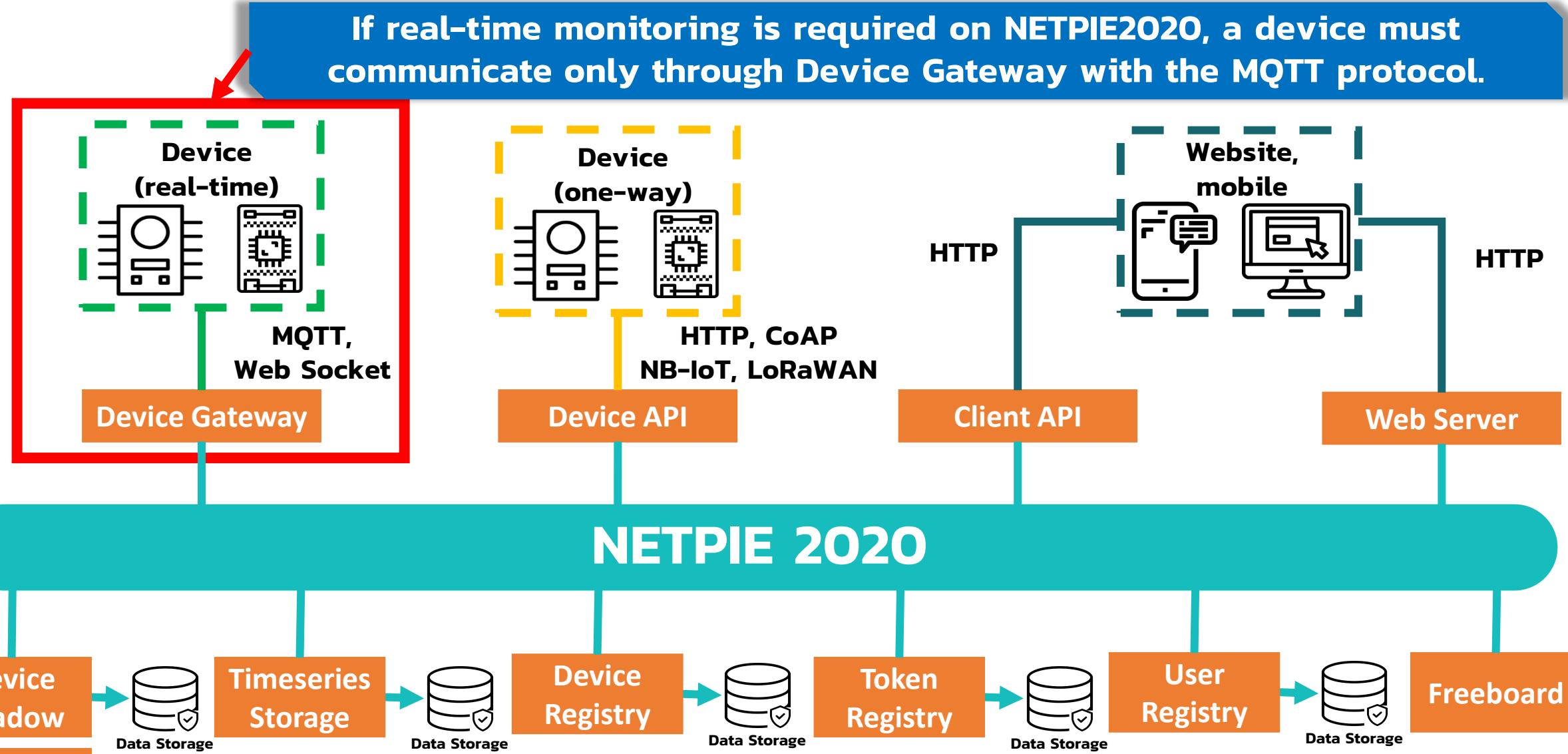


# 3 - Type of Communication in NETPIE2020

## NETPIE 2020 Architecture



# 3 - Type of Communication in NETPIE2020

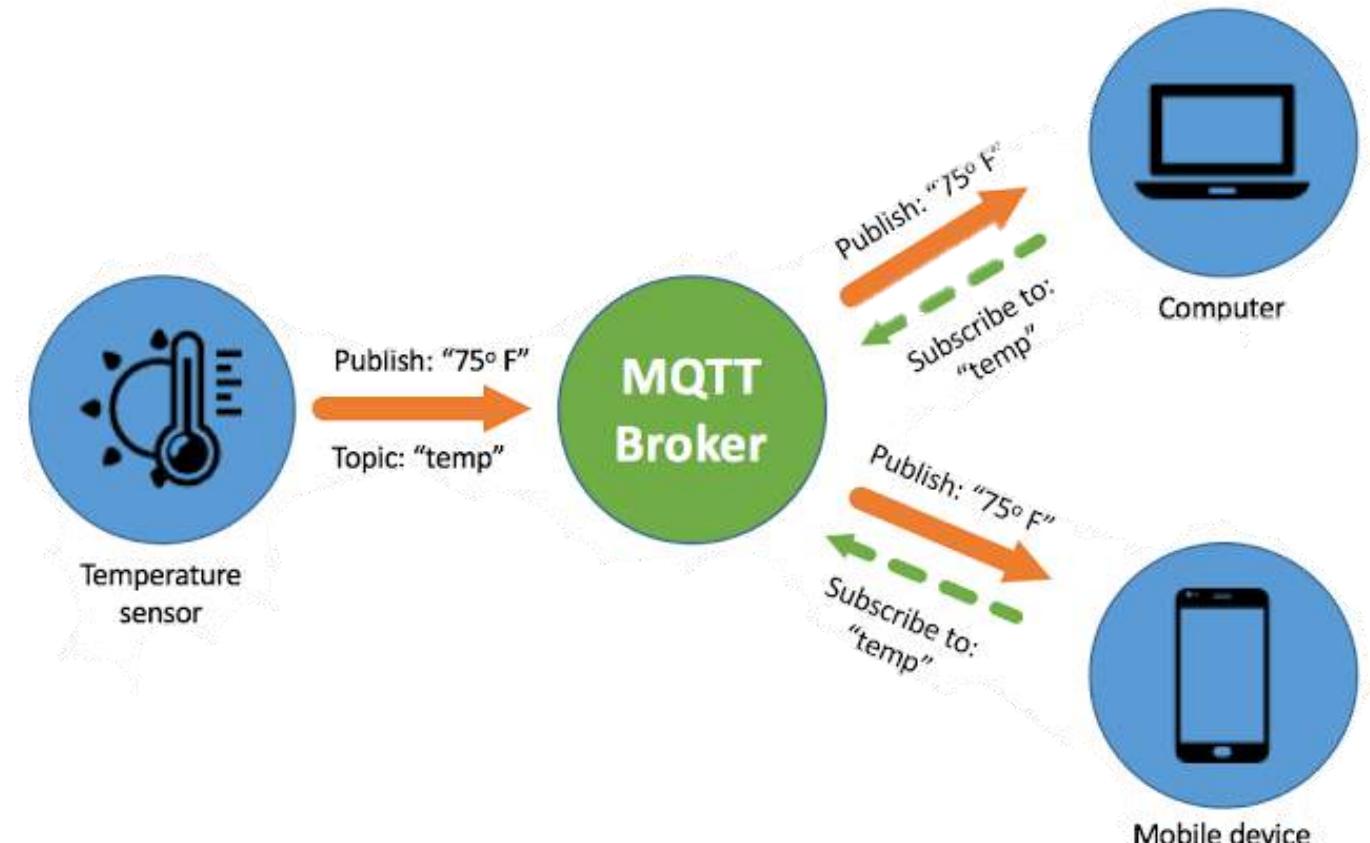


# 3 - Type of Communication in NETPIE2020

## NETPIE2020 communication [MQTT]

### What is MQTT ??

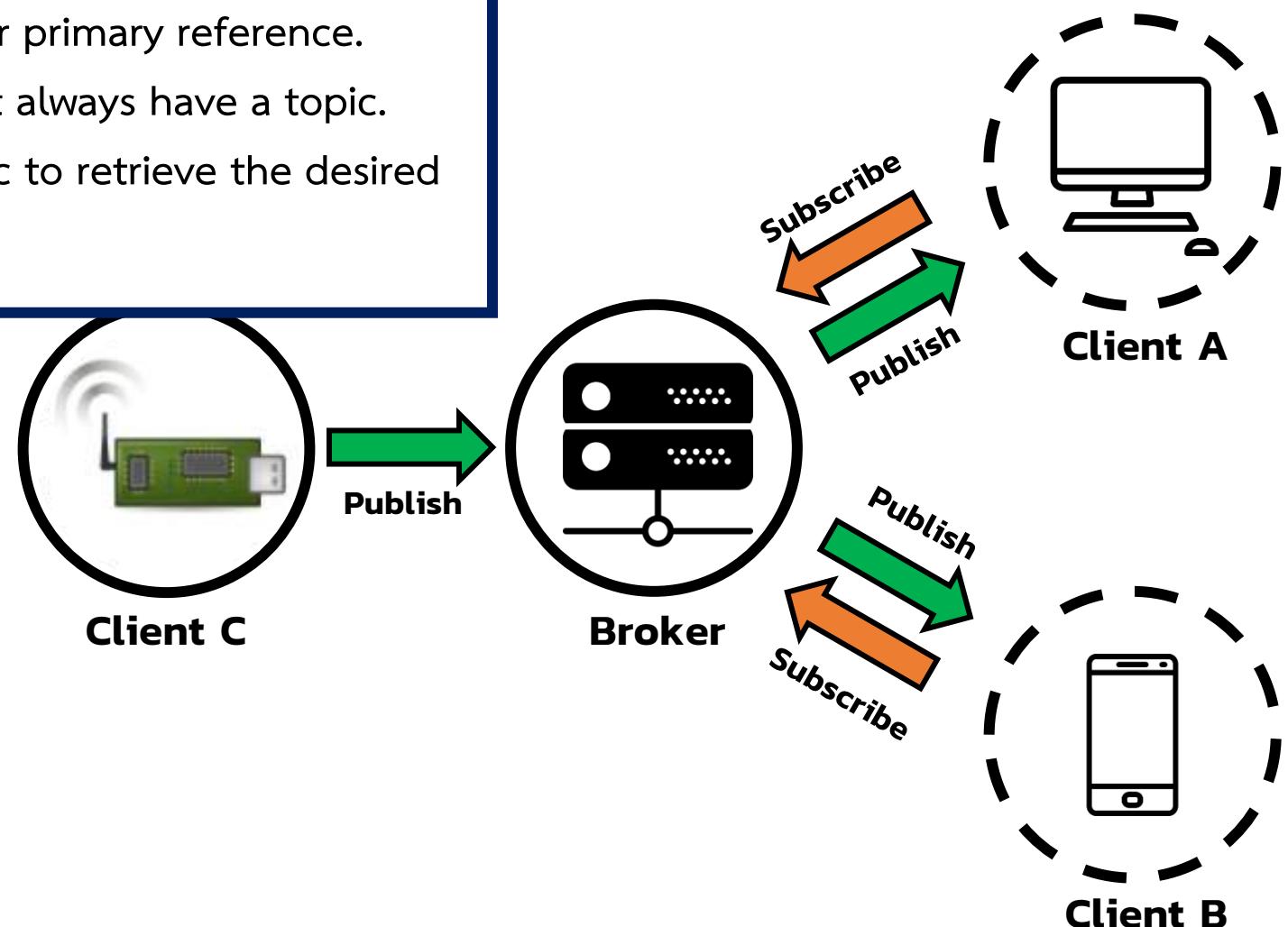
- MQTT is a protocol designed to connect M2M or device to device, which is supported by IoT
- Use the principle of data transmission, Publish / Subscribe, similar to the principles used in the Web Service that requires a Web Server as an intermediary between users' computers.
- But MQTT uses an intermediary called Broker that manages the transmission order between devices and Publish / Subscribe mechanisms.



# 3 - Type of Communication in NETPIE2020

transmissions on the MQTT have a topic as their primary reference.

- The data to be published to the Broker must always have a topic.
- On the subscribe end, it must refer to a topic to retrieve the desired information.



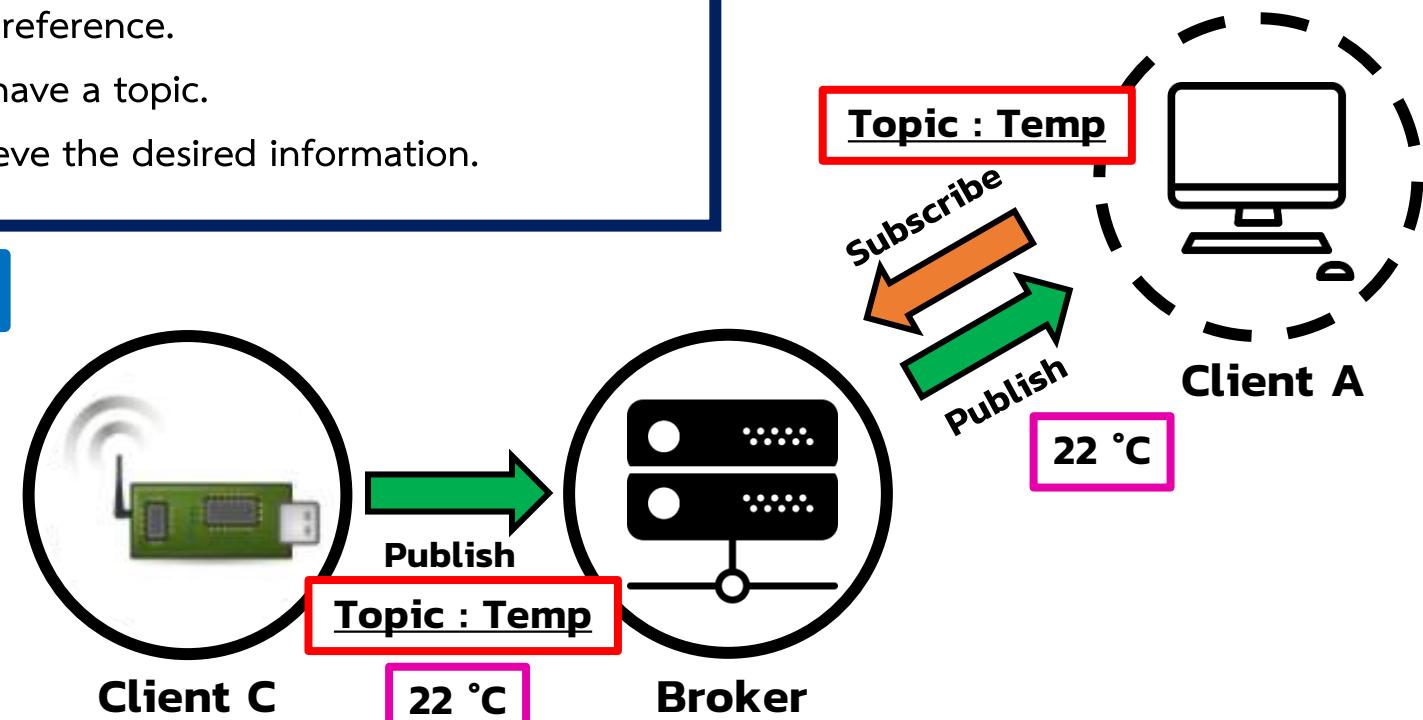
# 3 - Type of Communication in NETPIE2020

transmissions on the MQTT have a topic as their primary reference.

- The data to be published to the Broker must always have a topic.
- On the subscribe end, it must refer to a topic to retrieve the desired information.

## MQTT communication example

- Client C publishes information on a topic named **Temp**, while Client A subscribes to topic **Temp**.
- The data from Client C is a temperature with values equal to  $22^{\circ}\text{C}$ , so Client A receives the information.



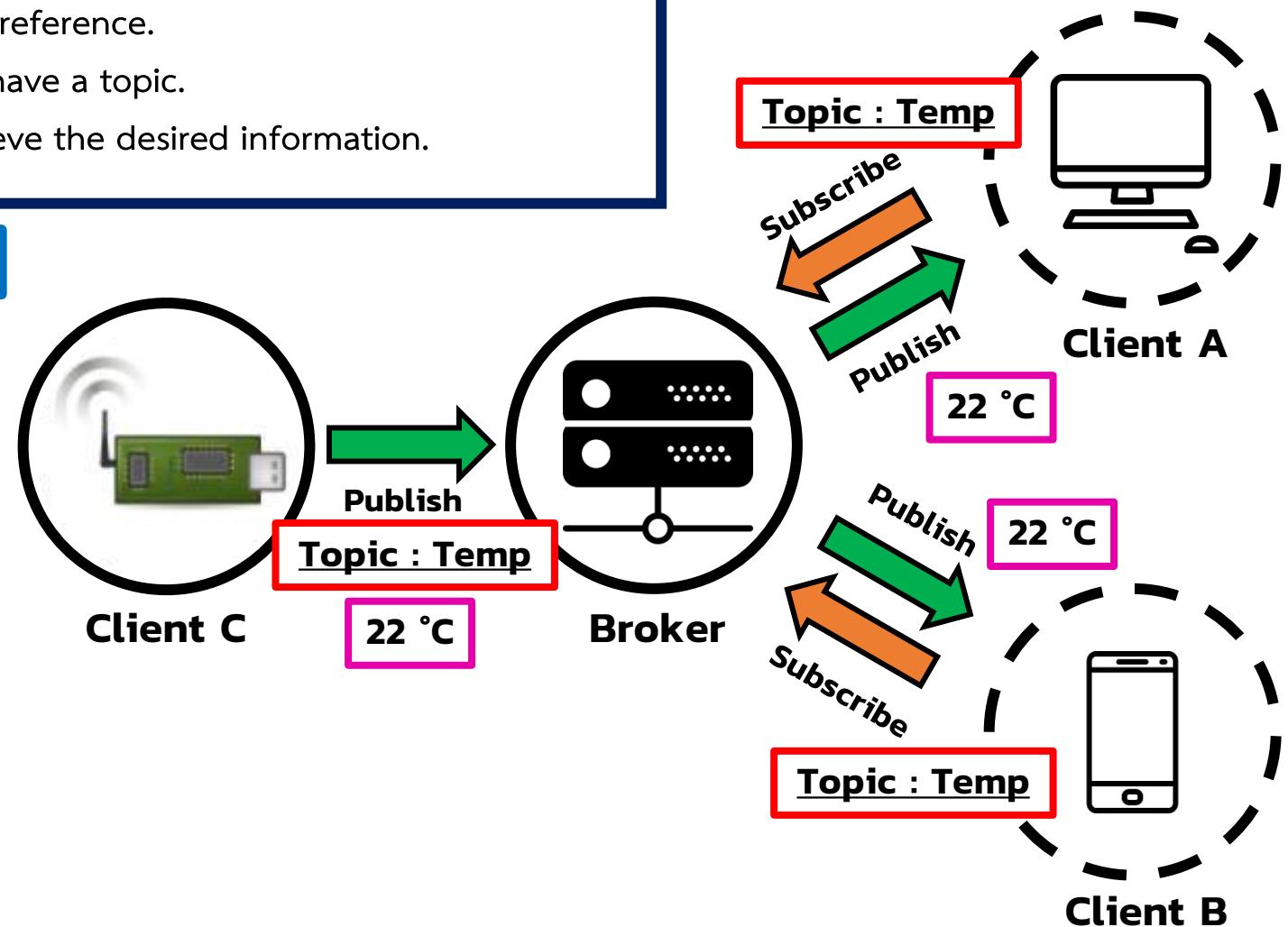
# 3 - Type of Communication in NETPIE2020

transmissions on the MQTT have a topic as their primary reference.

- The data to be published to the Broker must always have a topic.
- On the subscribe end, it must refer to a topic to retrieve the desired information.

## MQTT communication example

- Client C publishes information on a topic named **Temp**, while Client A subscribes to topic **Temp**.
- The data from Client C is a temperature with values equal to  $22^{\circ}\text{C}$ , so Client A receives the information.
- Later, Client B subscribes to topic **Temp** as well. The stored temperature is immediately sent to Client B after subscribing.



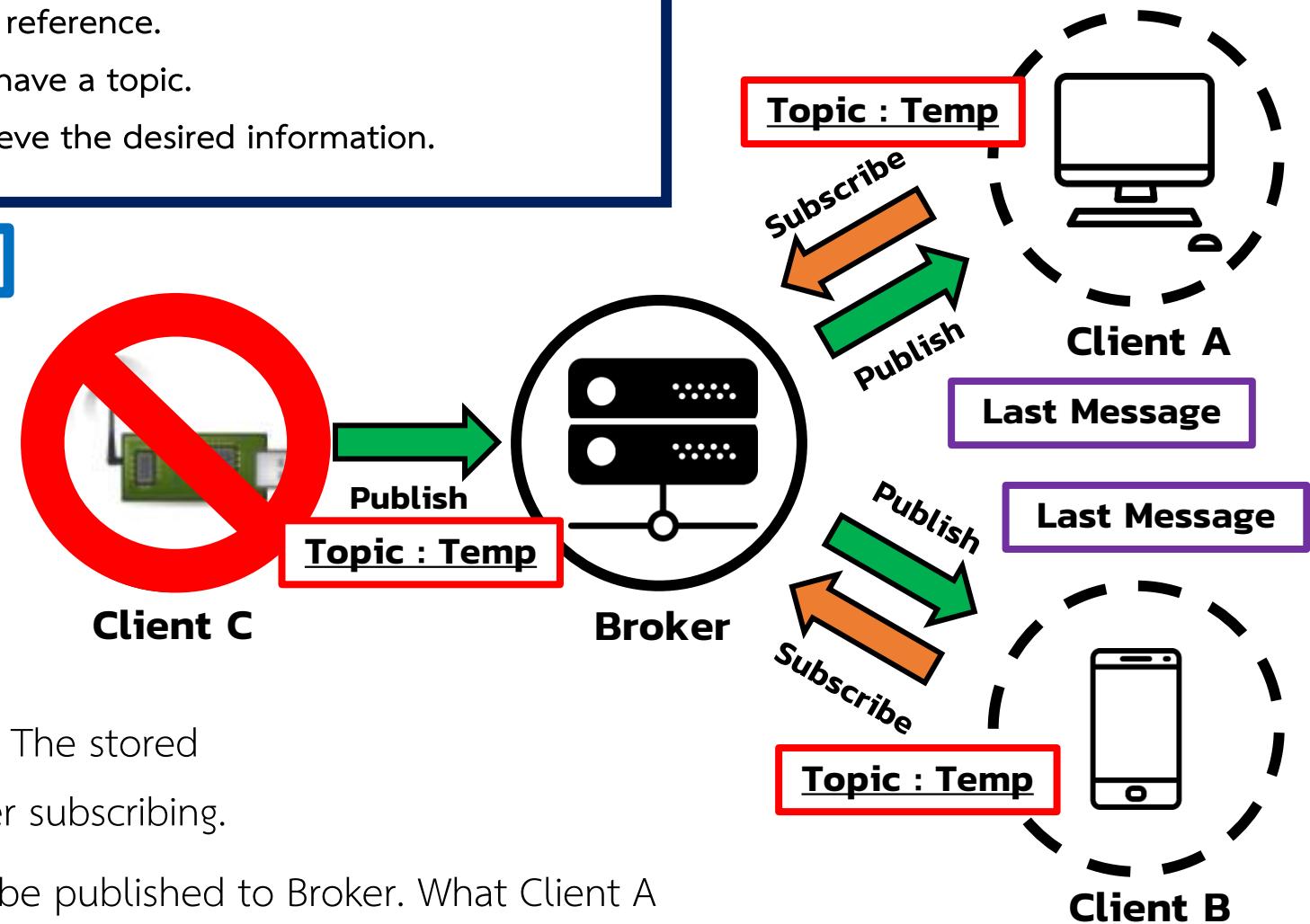
# 3 - Type of Communication in NETPIE2020

transmissions on the MQTT have a topic as their primary reference.

- The data to be published to the Broker must always have a topic.
- On the subscribe end, it must refer to a topic to retrieve the desired information.

## MQTT communication example

- Client C publishes information on a topic named **Temp**, while Client A subscribes to topic **Temp**.
- The data from Client C is a temperature with values equal to  $22^{\circ}\text{C}$ , so Client A receives the information.
- Later, Client B subscribes to topic **Temp** as well. The stored temperature is immediately sent to Client B after subscribing.
- But when Client C is disconnected, no data will be published to Broker. What Client A and B display is the last message sent by Client C

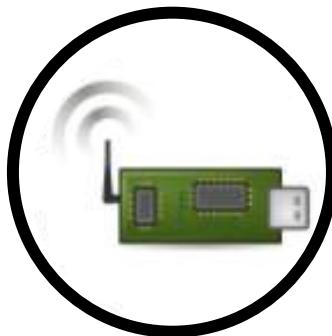


# 3 - Type of Communication in NETPIE2020

There are two types of messages sent to NETPIE2020 via MQTT.

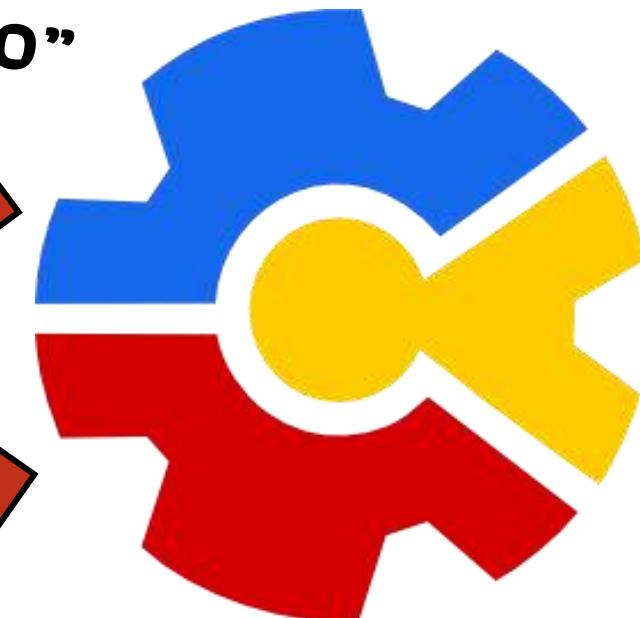
1

**Message**



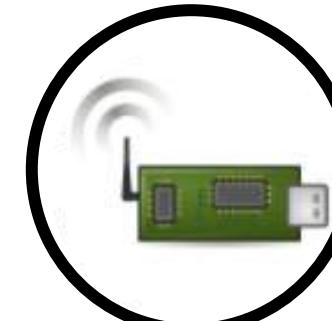
*“Hello NETPIE2020”*

**MQTT Protocol**



2

**Data**



**MQTT Protocol**  
*Temp = 25*

# 3 - Type of Communication in NETPIE2020

MQTT on NETPIE2020 has two types of transmission

1

**Message**

**Send a message via MQTT by referring to a topic.  
The format is @msg/topic**

2

**Shadow [Data]**

**Send message via MQTT to Device Shadow, to record latest  
message. The topic is @shadow/data/update  
with message in JSON format. For example, {data:{temp:24}}**

# 3 - Type of Communication in NETPIE2020

## Exercise 2 : Communication on NETPIE2020 with MQTT Box [Publish & Subscribe]

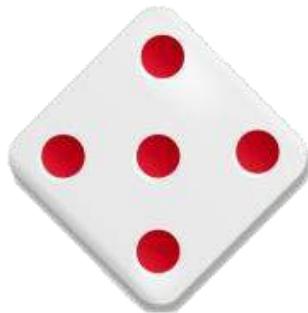


# 3 - Type of Communication in NETPIE2020

## Exercise 2 : Communication on NETPIE2020 with MQTT Box [Publish & Subscribe]

### Understand communication patterns on NETPIE2020

To send a message NETPIE2020 we need to reference a subject heading called **Topic**



**Topic : @msg/test**  
**"Hello NETPIE2020"**



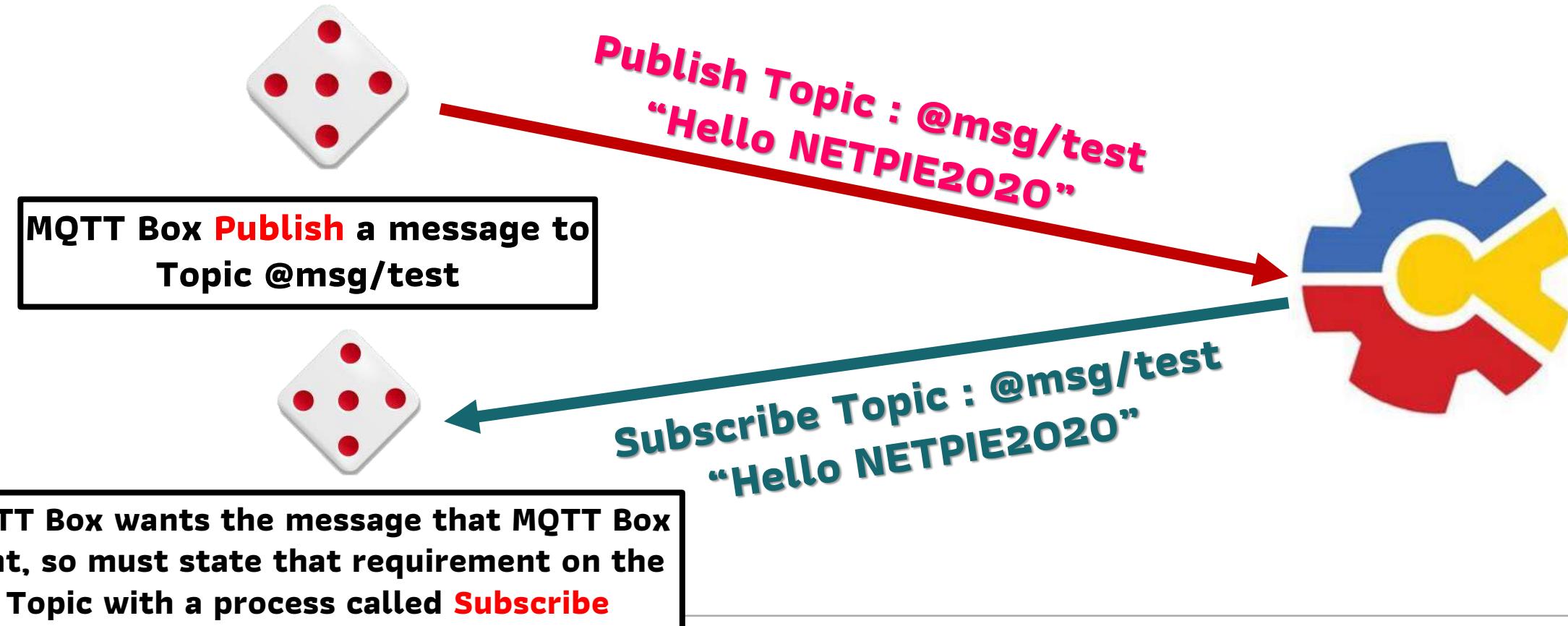
MQTTBox sends a message to NETPIE2020  
with the phrase "Hello NETPIE2020"  
**Publish**

NETPIE2020 acts like a chatroom  
**Broker**

# 3 - Type of Communication in NETPIE2020

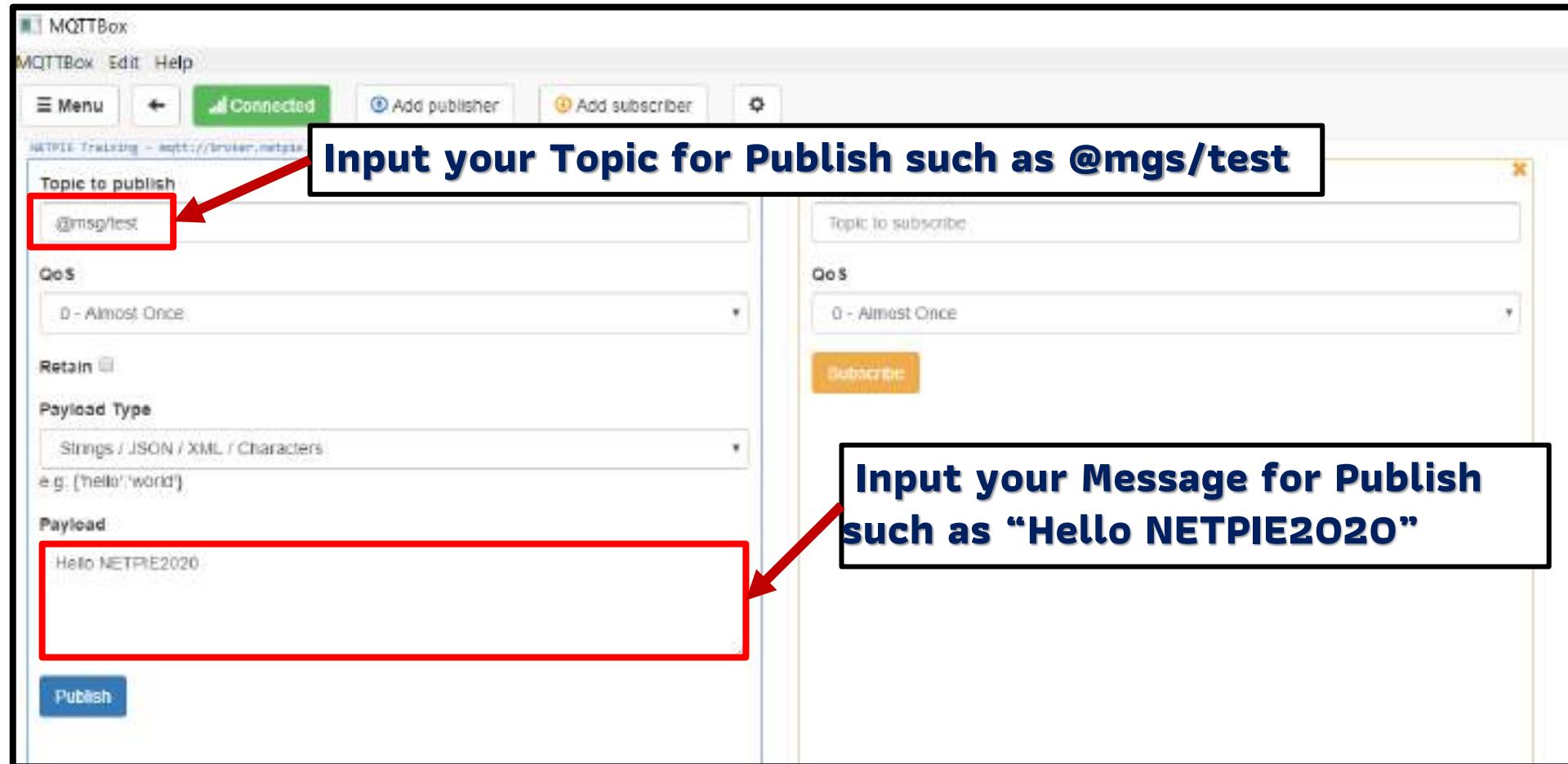
## Exercise 2 : Communication on NETPIE2020 with MQTT Box [Publish & Subscribe]

### Understand communication patterns on NETPIE2020



# 3 - Type of Communication in NETPIE2020

## Exercise 2 : Communication on NETPIE2020 with MQTT Box [Publish & Subscribe]



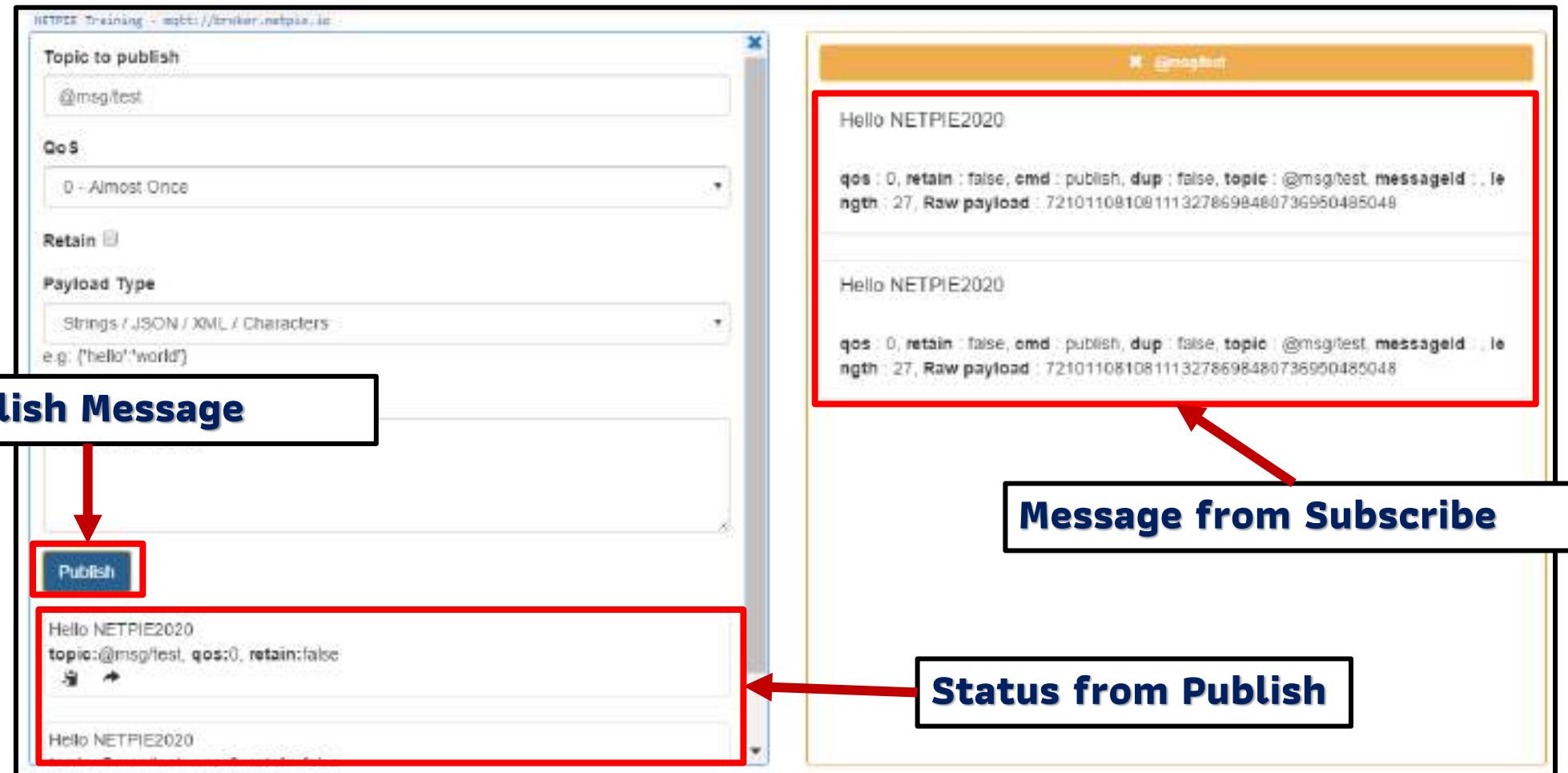
# 3 - Type of Communication in NETPIE2020

## Exercise 2 : Communication on NETPIE2020 with MQTT Box [Publish & Subscribe]



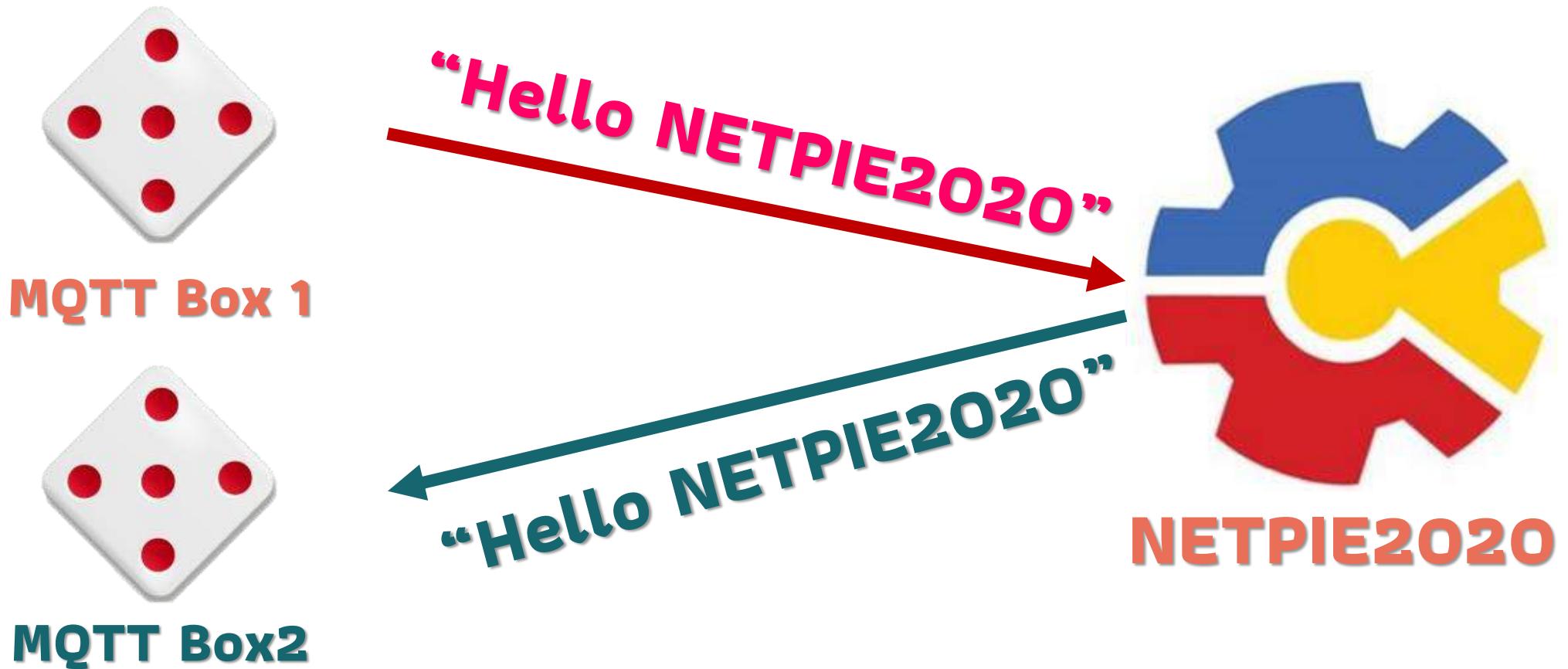
# 3 - Type of Communication in NETPIE2020

## Exercise 2 : Communication on NETPIE2020 with MQTT Box [Publish & Subscribe]



# 3 - Type of Communication in NETPIE2020

## Exercise 3 : Communication on NETPIE2020 with Device Group



# 3 - Type of Communication in NETPIE2020

## Exercise 3 : Communication on NETPIE2020 with Device Group

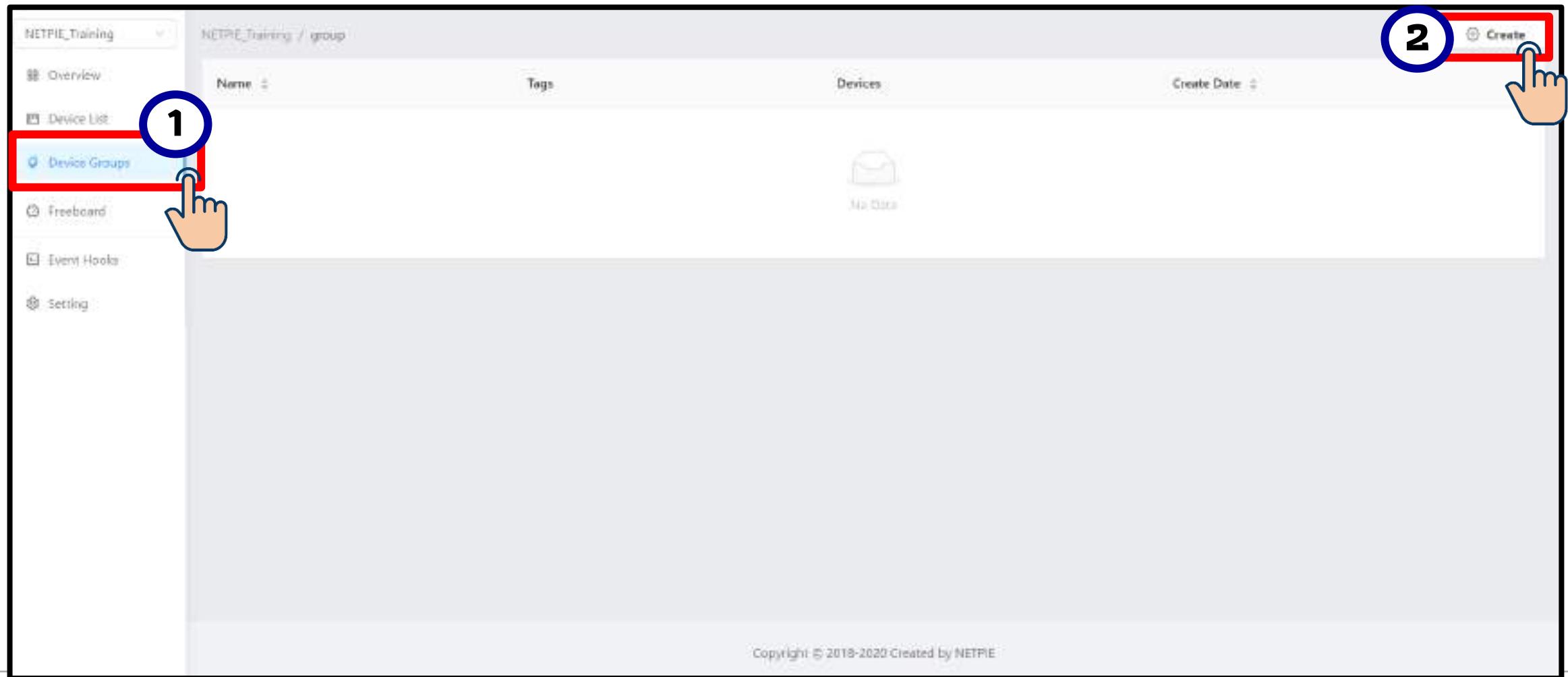
### Understand communication patterns on NETPIE2020



The two devices must be on the same group

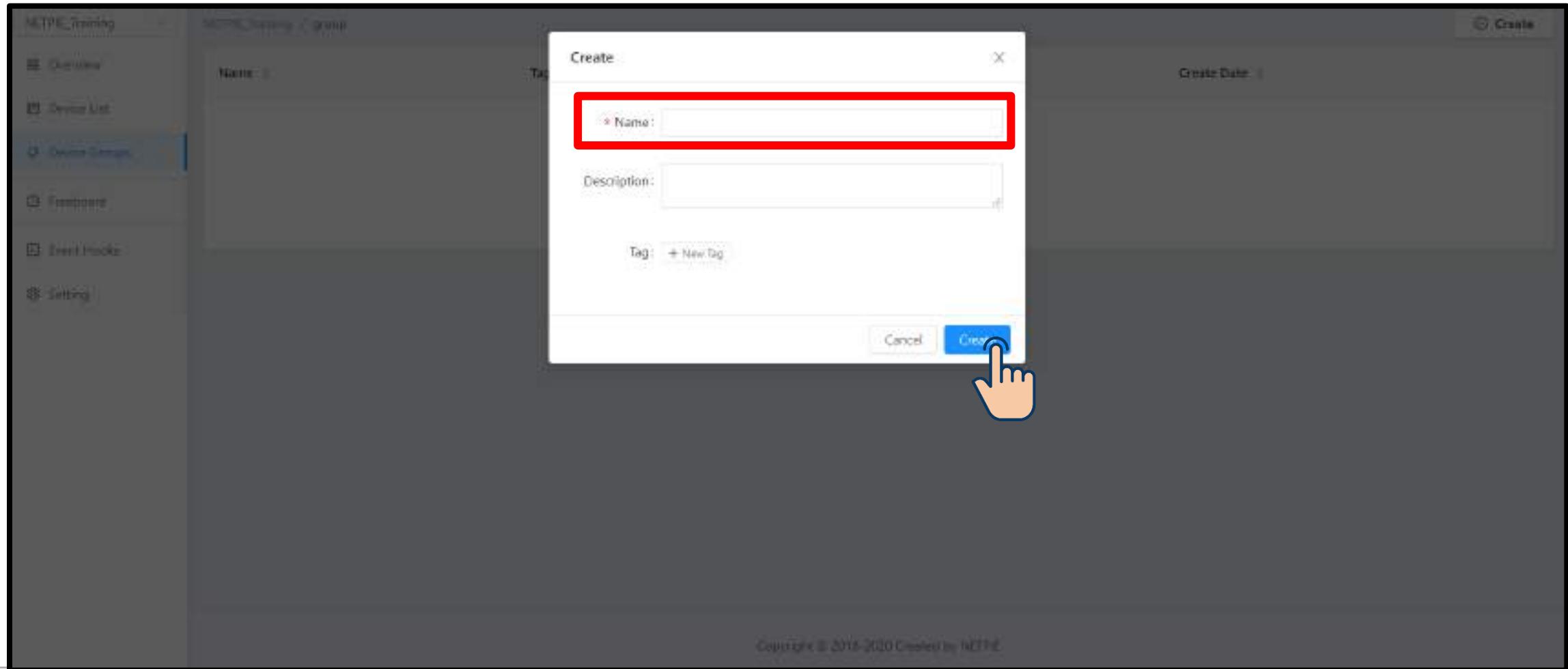
# 3 - Type of Communication in NETPIE2020

## Exercise 3 : Communication on NETPIE2020 with Device Group



# 3 - Type of Communication in NETPIE2020

## Exercise 3 : Communication on NETPIE2020 with Device Group



# 3 - Type of Communication in NETPIE2020

## Exercise 3 : Communication on NETPIE2020 with Device Group

The screenshot shows the NETPIE\_Training application's 'Device Groups' section. On the left, there's a sidebar with links: Overview, Device List, Device Groups (which is selected and highlighted in blue), Freeboard, Event Hooks, and Setting. The main area has a header 'NETPIE\_Training / group' with a 'Create' button. Below the header is a table with columns: Name, Tags, Devices, and Create Date. A single row is visible, labeled 'Group1'. To the right of the table are status indicators: '0 Online' and '0 Offline', followed by the creation date '2020-05-08 23:35'. At the bottom of the main area, there are navigation buttons for items (1-1 of 1 items) and a page size selector (10 / page). A red box highlights the 'Group1' row, and a red arrow points from this box down to a callout box containing the following text.

**Group created. But in this group there is no device yet, so you need to bring in the devices.**

# 3 - Type of Communication in NETPIE2020

## Exercise 3 : Communication on NETPIE2020 with Device Group

| Name    | Tags | Group | Create Date      |
|---------|------|-------|------------------|
| Device1 |      |       | 2020-03-08 14:34 |
| Device2 |      |       | 2020-05-08 23:24 |

**Create a new device for MQTTBox**

# 3 - Type of Communication in NETPIE2020

## Exercise 3 : Communication on NETPIE2020 with Device Group

The screenshot shows the NETPIE web interface with the title "NETPIE\_Training / device". On the left sidebar, "Device List" is selected. The main area displays a table with columns: Name, Tags, and Group. Two rows are visible: "Device1" (green circle) and "Device2" (grey circle). Each row has a checkbox in the first column. Two checkboxes are highlighted with red boxes and a hand cursor icon: the one for "Device1" and the one for "Device2".

# 3 - Type of Communication in NETPIE2020

## Exercise 3 : Communication on NETPIE2020 with Device Group

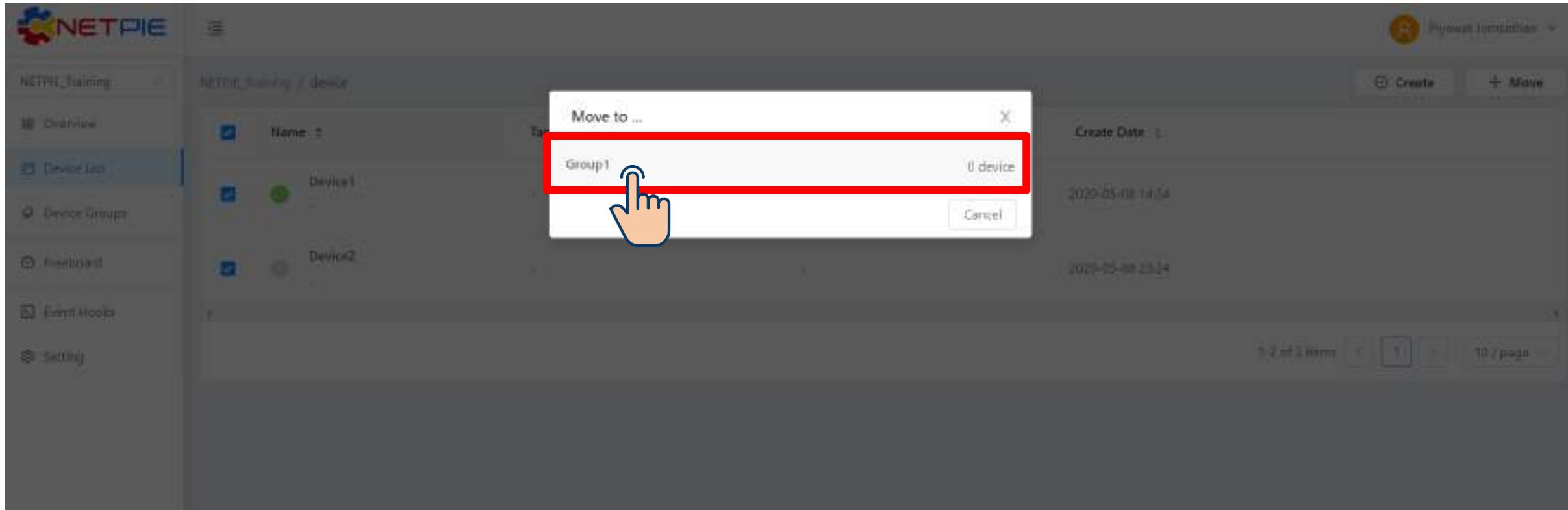
The screenshot shows the NETPIE2020 web interface for managing devices. On the left is a sidebar with navigation links: Overview, Device List (which is selected), Device Groups, Freeboard, Event Hooks, and Setting. The main area displays a table titled 'NETPIE\_Training / device' with two entries:

| Name    | Tags | Group | Create Date      |
|---------|------|-------|------------------|
| Device1 |      |       | 2020-05-08 14:34 |
| Device2 |      |       | 2020-05-08 23:24 |

In the top right corner of the main area, there are two buttons: 'Create' and '+ Move'. A red box highlights the '+ Move' button, and a hand cursor icon is positioned over it, indicating it is the target of the exercise.

# 3 - Type of Communication in NETPIE2020

## Exercise 3 : Communication on NETPIE2020 with Device Group



# 3 - Type of Communication in NETPIE2020

## Exercise 3 : Communication on NETPIE2020 with Device Group

The screenshot shows the NETPIE\_Training application's device management interface. On the left, a sidebar menu includes 'Overview', 'Device List' (which is selected and highlighted in blue), 'Device Groups', 'Freeboard', 'Event Hooks', and 'Setting'. The main area displays a table titled 'NETPIE\_Training / device' with columns for 'Name', 'Tags', 'Group', and 'Create Date'. Two devices are listed: 'Device1' and 'Device2', both belonging to 'Group1'. A red box highlights the 'Group' column for both devices, and a red arrow points from this box to a callout box at the bottom containing the text 'Show Group name that the devices belong'.

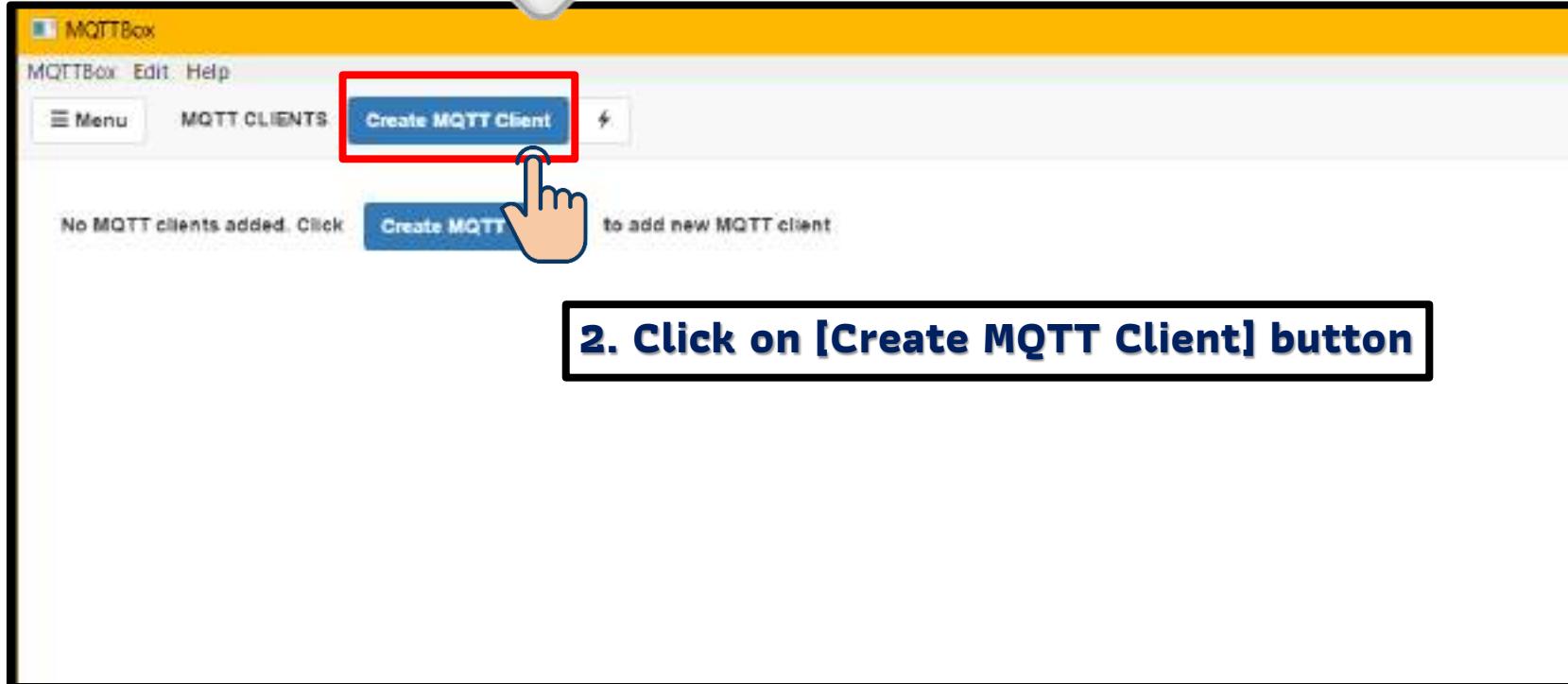
| Name    | Tags | Group  | Create Date      |
|---------|------|--------|------------------|
| Device1 |      | Group1 | 2020-05-08 14:34 |
| Device2 |      | Group1 | 2020-05-08 23:24 |

Show Group name that  
the devices belong

# 3 - Type of Communication in NETPIE2020

## Exercise 3 : Communication on NETPIE2020 with Device Group

### 1. Launch new MQTTBox



2. Click on [Create MQTT Client] button

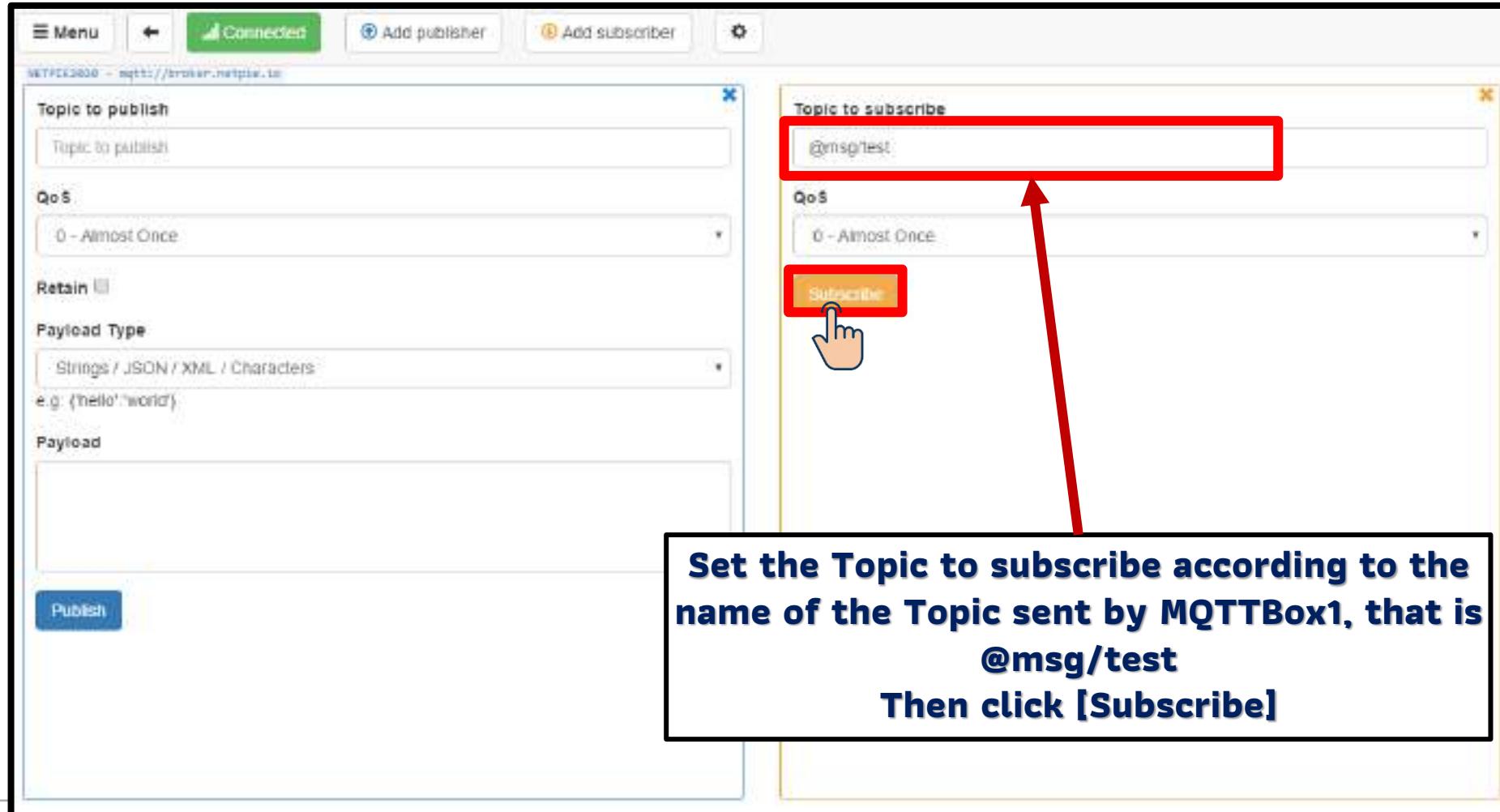
# 3 - Type of Communication in NETPIE2020

## Exercise 3 : Communication on NETPIE2020 with Device Group



# 3 - Type of Communication in NETPIE2020

## Exercise 3 : Communication on NETPIE2020 with Device Group



# 3 - Type of Communication in NETPIE2020

## Exercise 3 : Communication on NETPIE2020 with Device Group

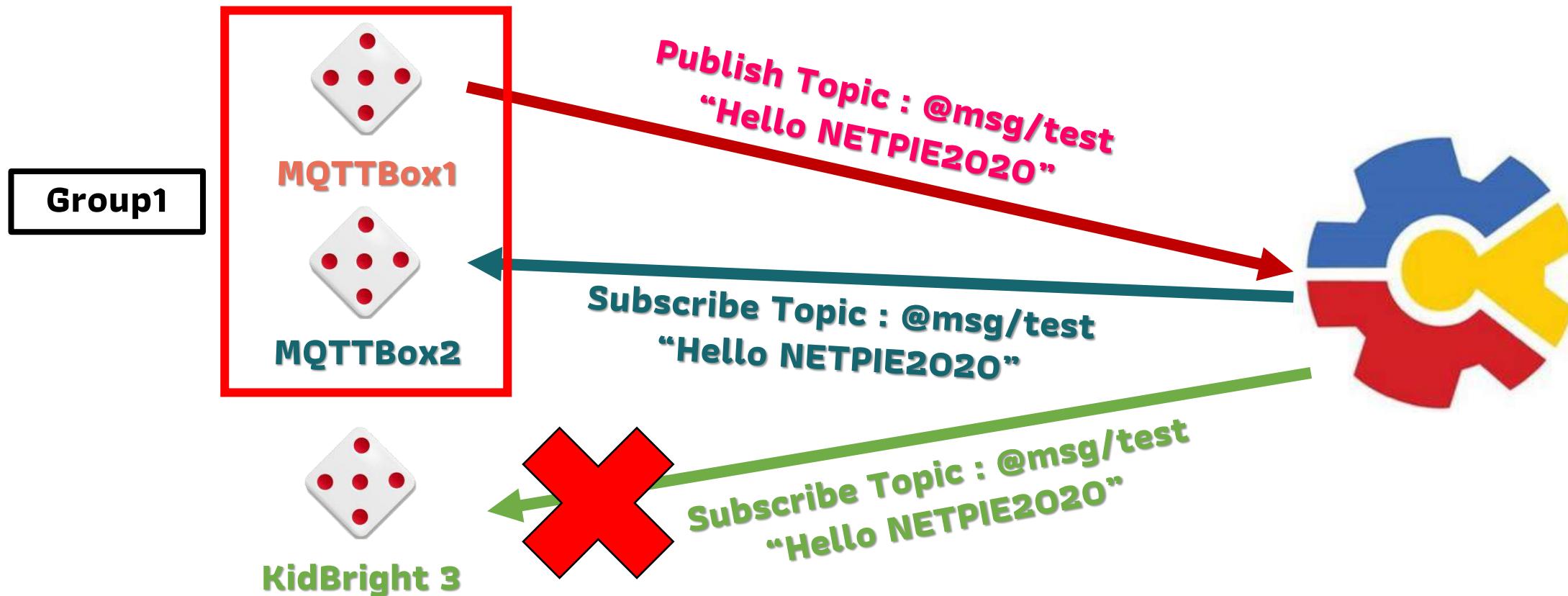
The screenshot shows the MQTTBox2 interface. On the left, there is a configuration panel for publishing a message. It includes fields for 'Topic to publish' (set to '@msg/test'), 'QoS' (set to '0 - Almost Once'), 'Retain' (unchecked), 'Payload Type' (set to 'Strings / JSON / XML / Characters'), and a 'Payload' area containing the string 'Hello NETPIE2020'. A blue 'Publish' button is at the bottom. On the right, a message list shows two entries under the heading '@msg/test': 'Hello NETPIE2020' with a timestamp and raw payload details, and another identical entry directly below it. A red arrow points from the text box below to the second message in the list.

The MQTTBox2 will receive message "Hello NETPIE2020" that MQTTBox1 has sent

# 3 - Type of Communication in NETPIE2020

## Exercise 3 : Communication on NETPIE2020 with Device Group

In case a device not belong to the group subscribes to this topic



# 3 - Type of Communication in NETPIE2020

## Exercise 3 : Communication on NETPIE2020 with Device Group

The screenshot shows the NETPIE\_2020 application interface. On the left is a sidebar with options: Overview, Device List (which is selected and highlighted in blue), Device Groups, Freeboard, Event Hooks, and Setting. The main area is titled 'NETPIE\_Training / device' and displays a table of devices. The columns are Name, Type, Group, and Create Date. There are three entries: Device3 (Group1, 2020-05-09 08:32), Device2 (Group1, 2020-05-08 23:24), and Device1 (Group1, 2020-05-08 14:34). A red box highlights the first row (Device3). A red arrow points from a callout box at the bottom to the 'Device3' row. The callout box contains the text 'Create a new device for MQTTBox3'.

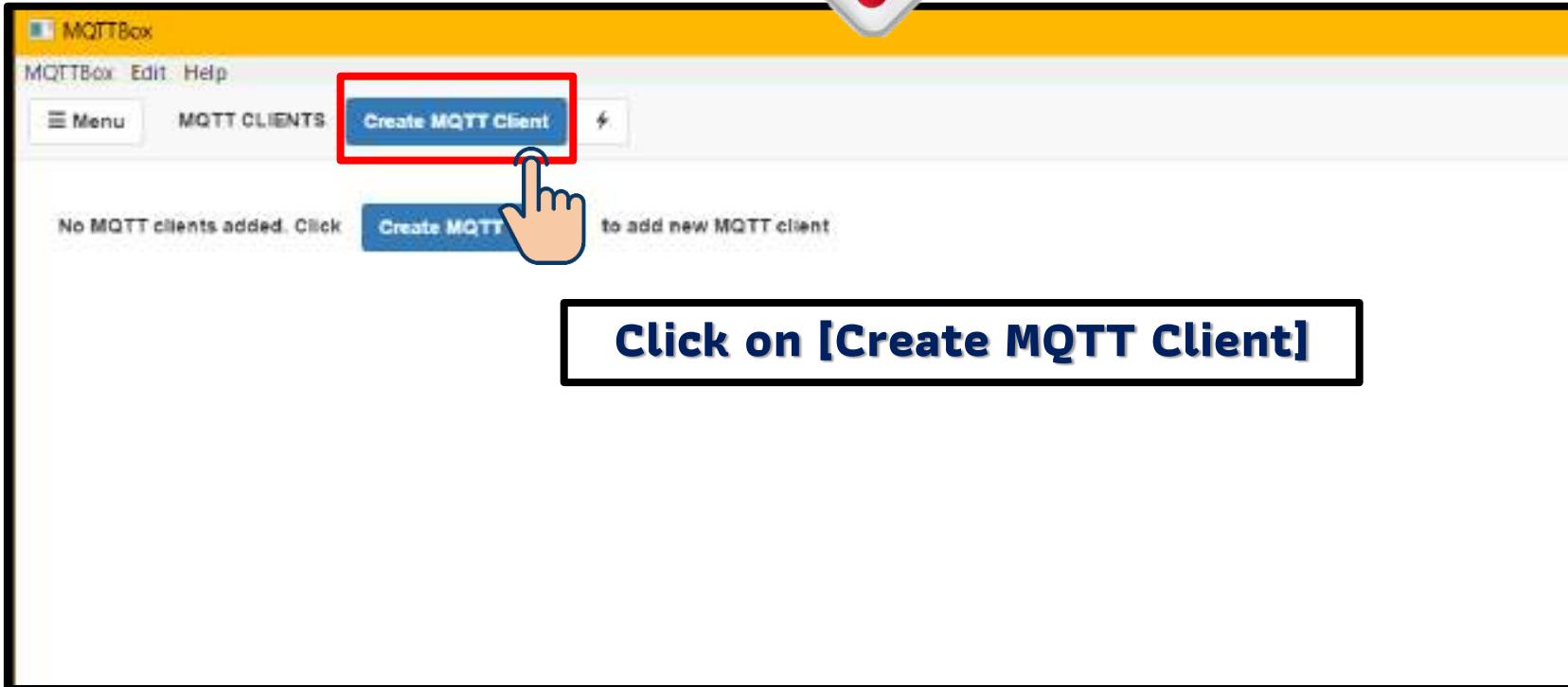
| Name    | Type | Group  | Create Date      |
|---------|------|--------|------------------|
| Device3 |      | Group1 | 2020-05-09 08:32 |
| Device2 |      | Group1 | 2020-05-08 23:24 |
| Device1 |      | Group1 | 2020-05-08 14:34 |

Create a new device for MQTTBox3

# 3 - Type of Communication in NETPIE2020

## Exercise 3 : Communication on NETPIE2020 with Device Group

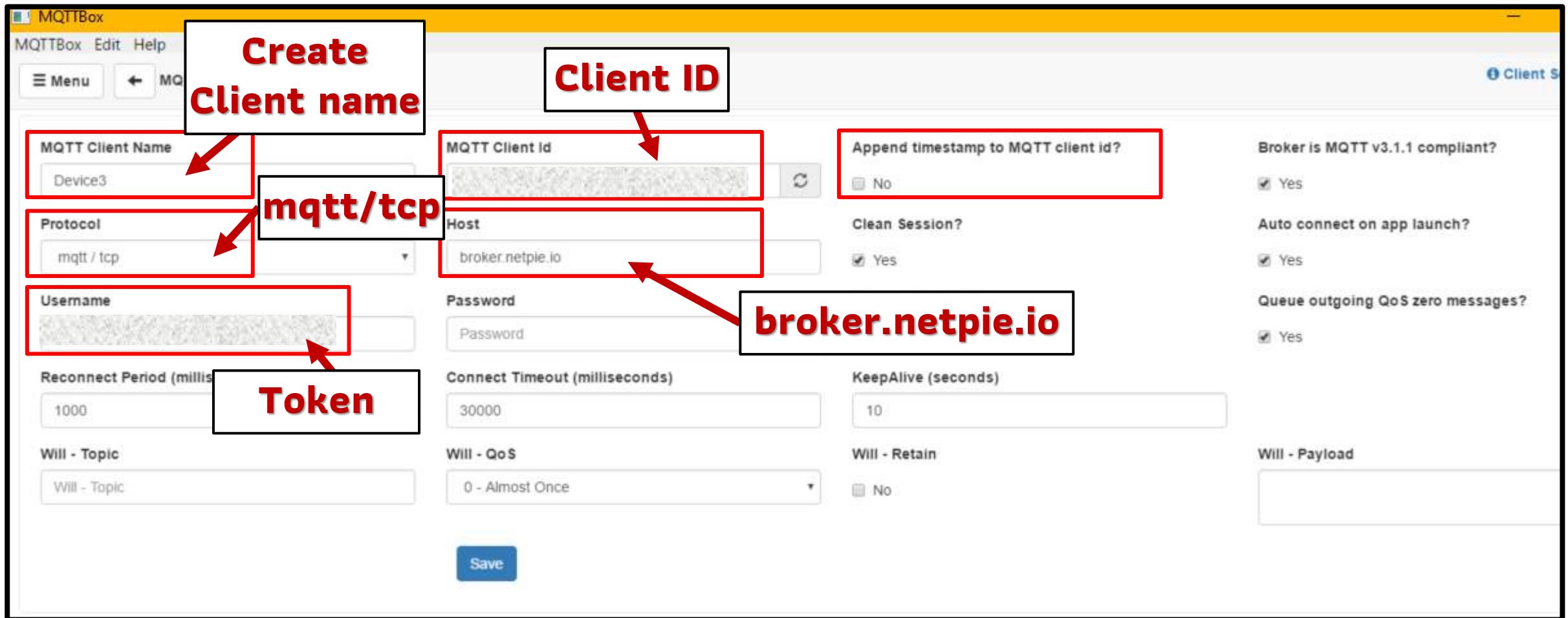
Open another MQTTBox window



Click on [Create MQTT Client]

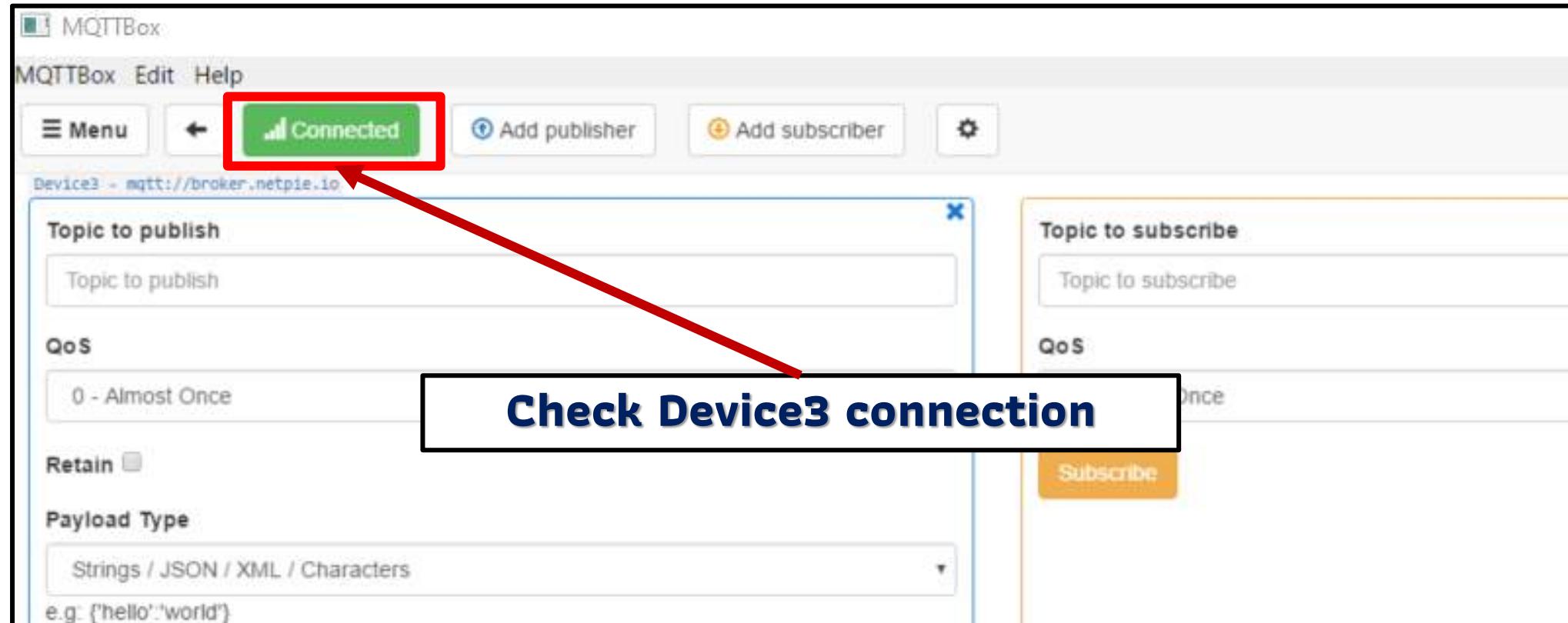
# 3 - Type of Communication in NETPIE2020

## Exercise 3 : Communication on NETPIE2020 with Device Group



# 3 - Type of Communication in NETPIE2020

## Exercise 3 : Communication on NETPIE2020 with Device Group



# 3 - Type of Communication in NETPIE2020

## Exercise 3 : Communication on NETPIE2020 with Device Group

The image shows two MQTTBox interfaces, MQTTBox2 and MQTTBox3, connected via a Device Group.

**MQTTBox2 (Top Left):** This interface is connected to the broker at `mqtt://broker.netpie.io:1883`. It has a topic to publish set to `@msg/test`, QoS set to 0 - Almost Once, and a retain checkbox checked. The payload type is set to Strings / JSON / XML / Characters, with the payload value being `e.g. ("hello","world")`. A red box highlights the topic `@msg/test` in the "Topic to subscribe" field of the configuration window, and a red arrow points from this box to the "Topic to subscribe" field in MQTTBox3.

**MQTTBox3 (Bottom Right):** This interface is connected to the broker at `mqtt://broker.netpie.io:1883`. It has a topic to publish set to "Topic to publish". The topic to subscribe is set to `@msg/test`, QoS is set to 0 - Almost Once, and the retain checkbox is checked. A red box highlights the topic `@msg/test` in the "Topic to subscribe" field, and a red arrow points from this box to the "Topic to subscribe" field in MQTTBox2.

**Text Overlay:** A callout box contains the following instructions:

**Set the Topic to subscribe according to the name of the Topic sent by MQTTBox1, that is  
 @msg/test  
 Then click [Subscribe]**

# 3 - Type of Communication in NETPIE2020

## Exercise 3 : Communication on NETPIE2020 with Device Group

The screenshot shows the MQTTBox2 interface. In the top left, there's a configuration panel with fields for Topic to publish (@msgtest), QoS (0 - Almost Once), Retain, Payload Type (Strings / JSON / XML / Characters), and a Payload input field containing "Hello Smart Factory IoT Challenge 2020". A large red box labeled "MQTTBox2" covers the top portion of the interface. Below it, three message logs are displayed:

- Message 1: Hello NETPIE2020  
qos: 0, retain: false, cmd: publish, dup: false, topic: @msgtest, messageId: length: 27, Raw payload: 721 011081081113278698480736950485048
- Message 2: Hello NETPIE2020  
qos: 0, retain: false, cmd: publish, dup: false, topic: @msgtest, messageId: length: 27, Raw payload: 721 011081081113278698480736950485048
- Message 3: Hello NETPIE2020  
qos: 0, retain: false, cmd: publish, dup: false, topic: @msgtest, messageId: length: 27, Raw payload: 721 011081081113278698480736950485048

**MQTTBox2 received the message sent by  
MQTTBox1  
“Hello NETPIE2020”**

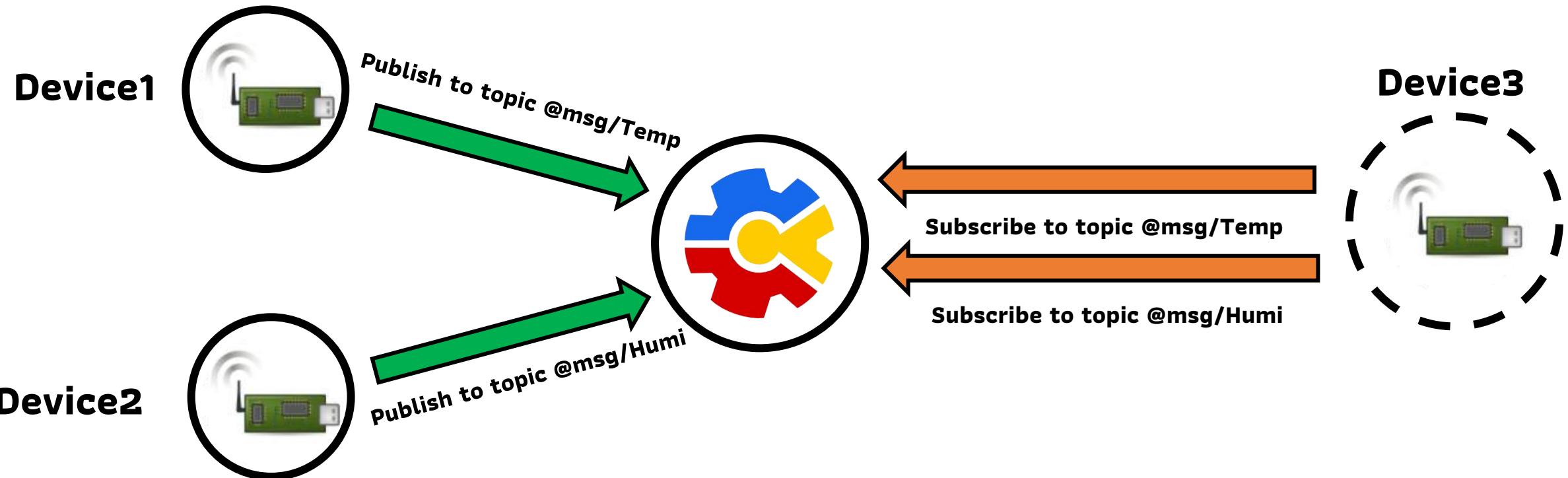
The screenshot shows the MQTTBox3 interface. It has a similar layout to MQTTBox2, with fields for Topic to publish (@msgtest), QoS (0 - Almost Once), Retain, Payload Type (Strings / JSON / XML / Characters), and a Payload input field. A large red box labeled "MQTTBox3" covers the top portion. Below it, there are no message logs displayed.

**MQTTBox3 did not receive message  
because it does not belong to the group**

# 3 - Type of Communication in NETPIE2020

## Wildcard Topic

If you want Device 3 to receive messages from Device1 and Device2,  
you need to subscribe to 2 topics



4

# Data Management in NETPIE2020

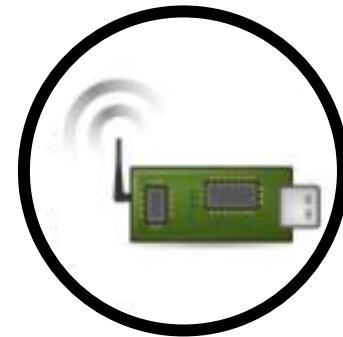


# 4 - Data Management in NETPIE2020

There are two types of messages sent to NETPIE2020 via MQTT.

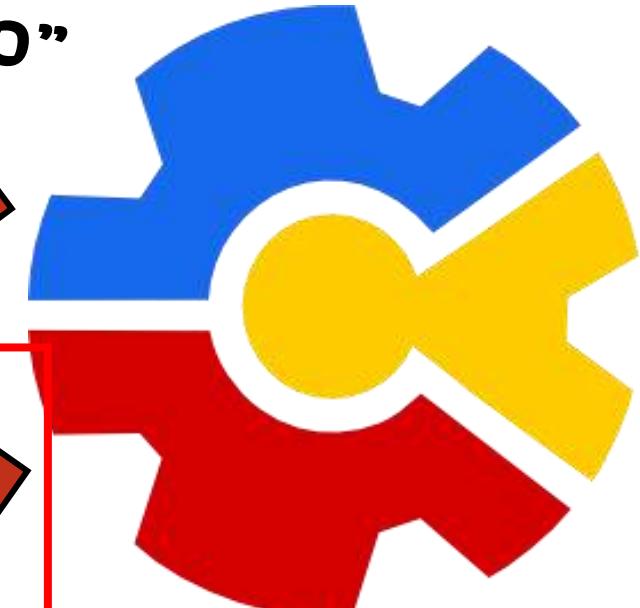
1

**Message**



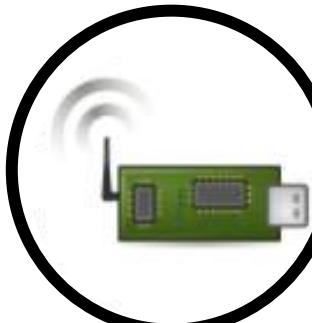
*“Hello NETPIE2020”*

**MQTT Protocol**



2

**Data**



**MQTT Protocol**  
*Temp = 25*

# 4 - Data Management in NETPIE2020

**NETPIE2020 manages device information in 5 main sections.**

**1 Device Shadow : Latest device database**

**2 Device Schema : Device data structure**

**3 Device Feed : Device Data Storage**

**4 Device Trigger : Conditional device data**

**5 Event Hooks : Device transmission formats**

**We will talk about this on 11 Sep'21**



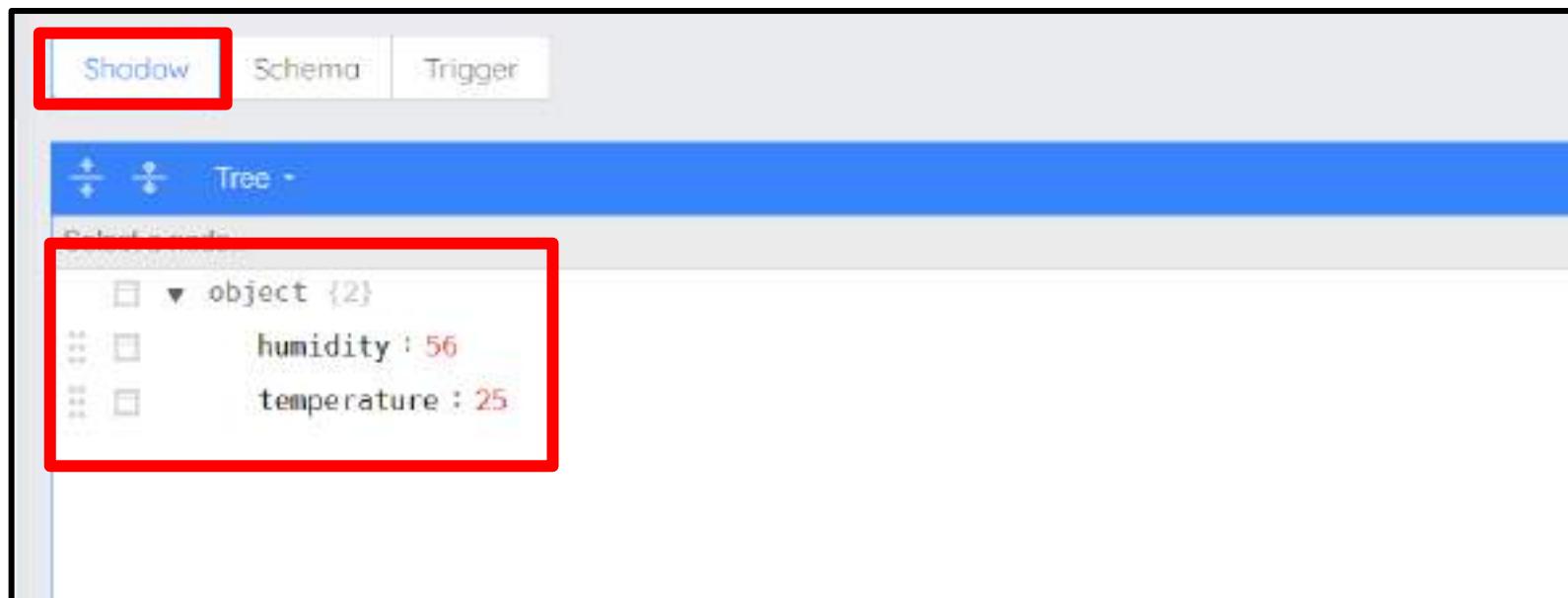
**NETPIE2020**

# 4 - Data Management in NETPIE2020

## Device Shadow

**Device Shadow is the virtual database of the device that is allocated for every device for two types of data storage**

- 1. Device Shadow Data : sensor data, for example**
- 2. Device Shadow State : device online/offline status, for example**



# 4 - Data Management in NETPIE2020

## Device Shadow

**The MQTT topic that is involved with managing Device Shadow**

### Shadow Topic

Used to manage own device shadow for publishing to edit Shadow information, and subscribing to receive Shadow information.

| Publish Topic              | Description                                                     | Subscribe Topic             |
|----------------------------|-----------------------------------------------------------------|-----------------------------|
| <b>@shadow/data/update</b> | To update Shadow Data value by sending a payload in JSON format | <b>@shadow/data/updated</b> |

# 4 - Data Management in NETPIE2020

## Device Shadow

Example of sending data in JSON format



MQTTBox1

@shadow/data/update  
Publish : { "data": { "Temp" : 24, "light" : 80 }}

JSON format



NETPIE2020

"Temp" : 24  
"light" : 58



Device Shadow

But in order to store data in Device Shadow, you need to create a Device Schema first.

# 4 - Data Management in NETPIE2020

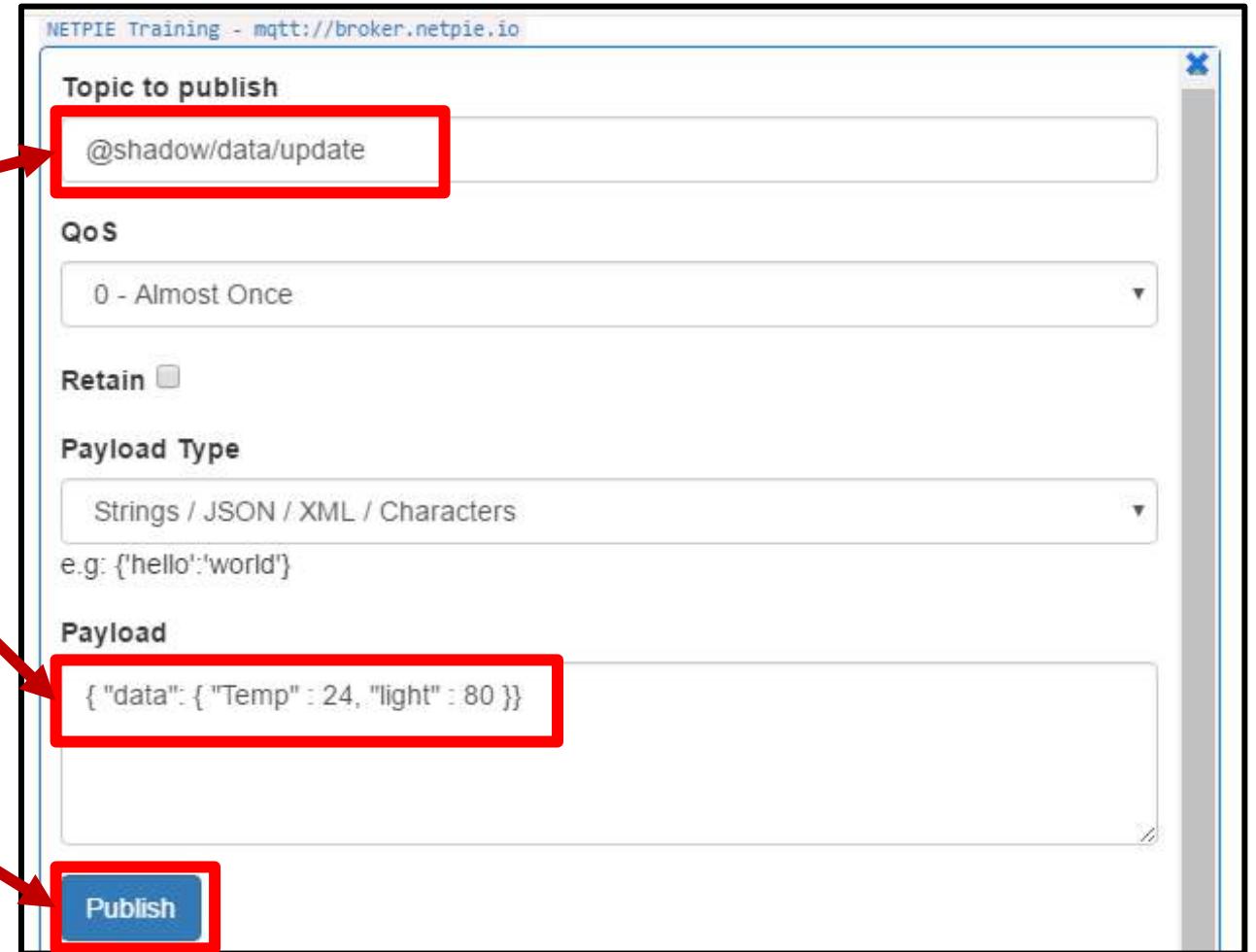
## Exercise 4 : Sent data to NETPIE

Open MQTTBox1 window

Topic to send data

Set payload data such as  
{ "data": { "Temp" : 24, "light" : 80 }}

Click Publish for sent data



# 4 - Data Management in NETPIE2020

## Exercise 4 : Sent data to NETPIE

NETPIE\_Training / device / Device1

Description

Key

- Client ID : d97f67f4-734a-4d21-9
- Token : RLpomZchpumm1ugt
- Secret : !l3zKPgpmd#mg\*17N

Status : Online

Enable

**Data from MQTTBox1 Publish**  
[But this is last data only]

Shadow Schema Trigger Feed

Tree

Select a node...

object /

Temp : 24  
light : 80

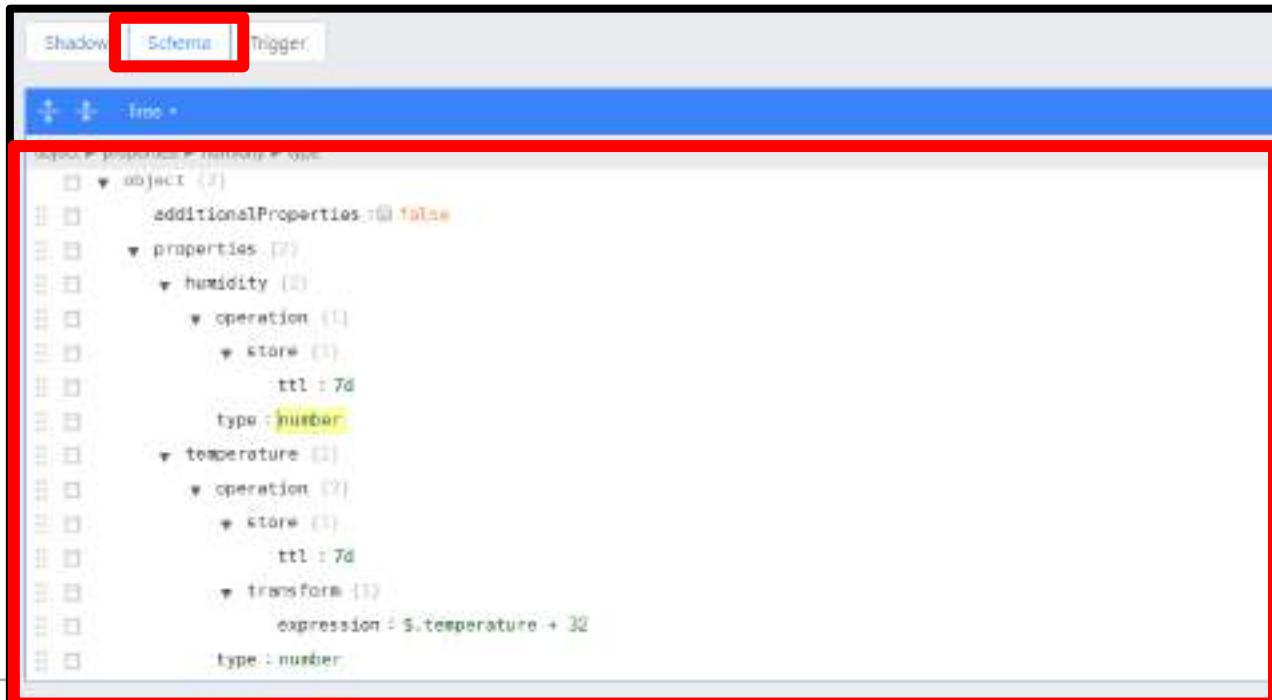
**But in order to store data in Device Shadow,  
you need to create a Device Schema first.**

# 4 - Data Management in NETPIE2020

## Device Schema

Device Schema is a data structure defined to manage Device Shadow. For devices that need data management, a Device Schema should be created. This Device Schema allows the server to

- Check data types before storing
- Converting data before storage, such as changing data units.
- Data collection in Timeseries Database (Feed)



# 4 - Data Management in NETPIE2020

## Device Schema

### Declaring the Device Schema in JSON format.

```
{  
  "additionalProperties": false,  
  "properties": {  
    "light": {  
      "operation": {  
        "store": {  
          "ttl": "7d"  
        }  
      },  
      "type": "number"  
    },  
    "temperature": {  
      "operation": {  
        "store": {  
          "ttl": "7d"  
        }  
      },  
      "transform": {  
        "expression": "[[$.temperature]*1.8] + 32"  
      },  
      "type": "number"  
    }  
  }  
}
```

#### additionalProperties

This is the permission to save data to Shadow or Timeseries Database  
in case the data is not declared in the properties section. additionalProperties has 2 status values

true : authorize data write to Shadow or Timeseries Database

false : prohibit data write for data not defined in Properties

In the example, two items in properties are humidity and temperature

If the data received are temp, humid, light

additionalProperties = true : will record temperature, humidity and light

additionalProperties = false : will record only temperature, light

# 4 - Data Management in NETPIE2020

## Device Schema

### Declaring the Device Schema in JSON format

```
{  
    "additionalProperties": false,  
    "properties": {  
        "light": {  
            "operation": {  
                "store": {  
                    "ttl": "7d"  
                }  
            },  
            "type": "number"  
        },  
        "temperature": {  
            "operation": {  
                "store": {  
                    "ttl": "7d"  
                }  
            },  
            "transform": {  
                "expression": "[$.temperature]*1.8 + 32"  
            },  
            "type": "number"  
        }  
    }  
}
```

#### Properties

First, define a field name [for example, 'light' and 'Temperature'] and define properties for each field, which are divided into two parts:

**Operation** For setting up data handling in that field, including  
**store** for keeping data inTimeseries Database  
**ttl** duration of data inTimeseries Database. Data that exceeds expiration date will be automatically deleted. To store system data, this value needs to be set in units of ms [milliseconds], s [seconds], m [minutes], h [hours], d [days], y [years]

**Transform** data transformation before keeping  
**expression** formula for transformation

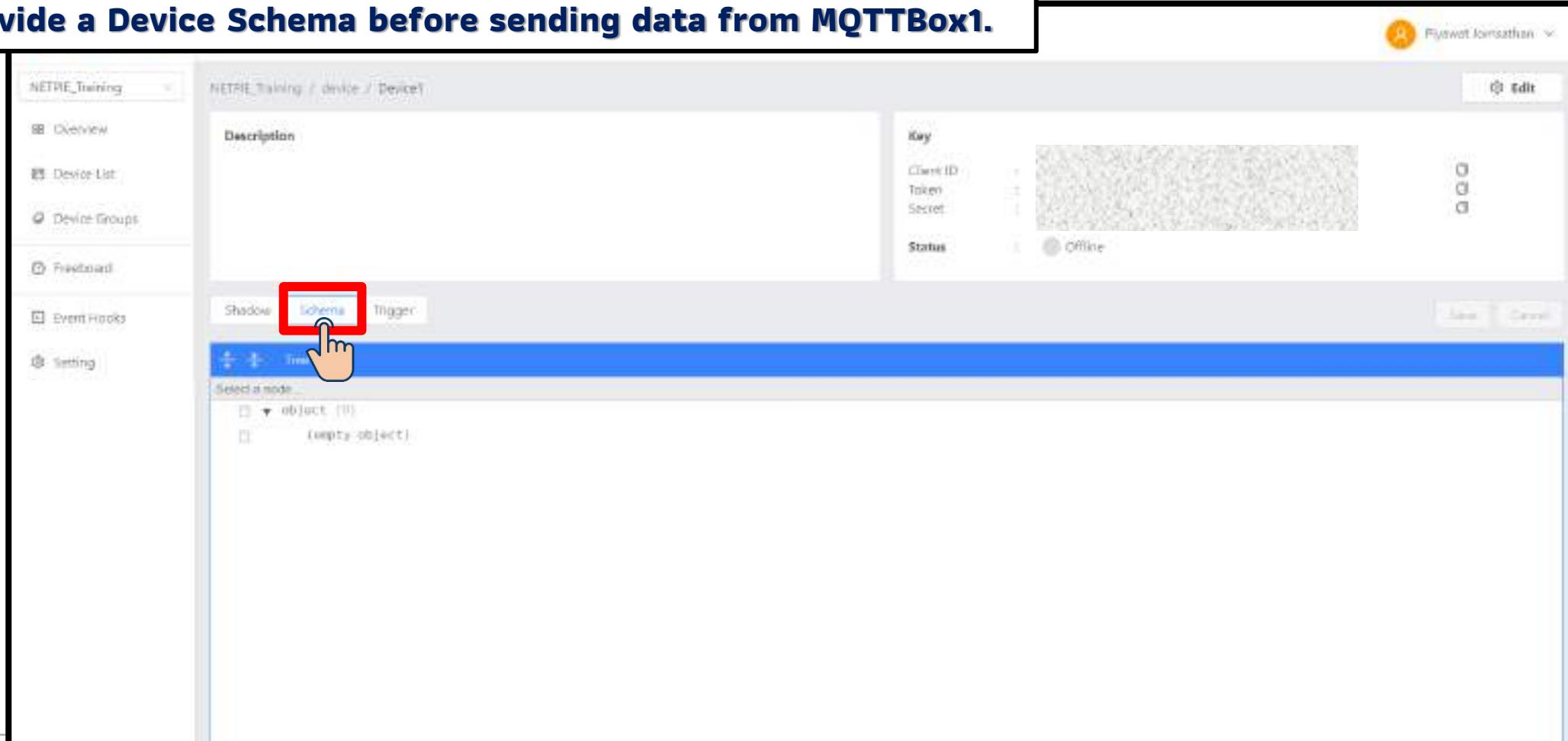
For example, a formula to convert Celsius to Fahrenheit= [\[\\$.temperature\\*1.8\] + 32](#)

**Type** data type in a field, such as number, string, array, object

# 4 - Data Management in NETPIE2020

## Exercise 5 : Set Schema and Send Data to Device Feed

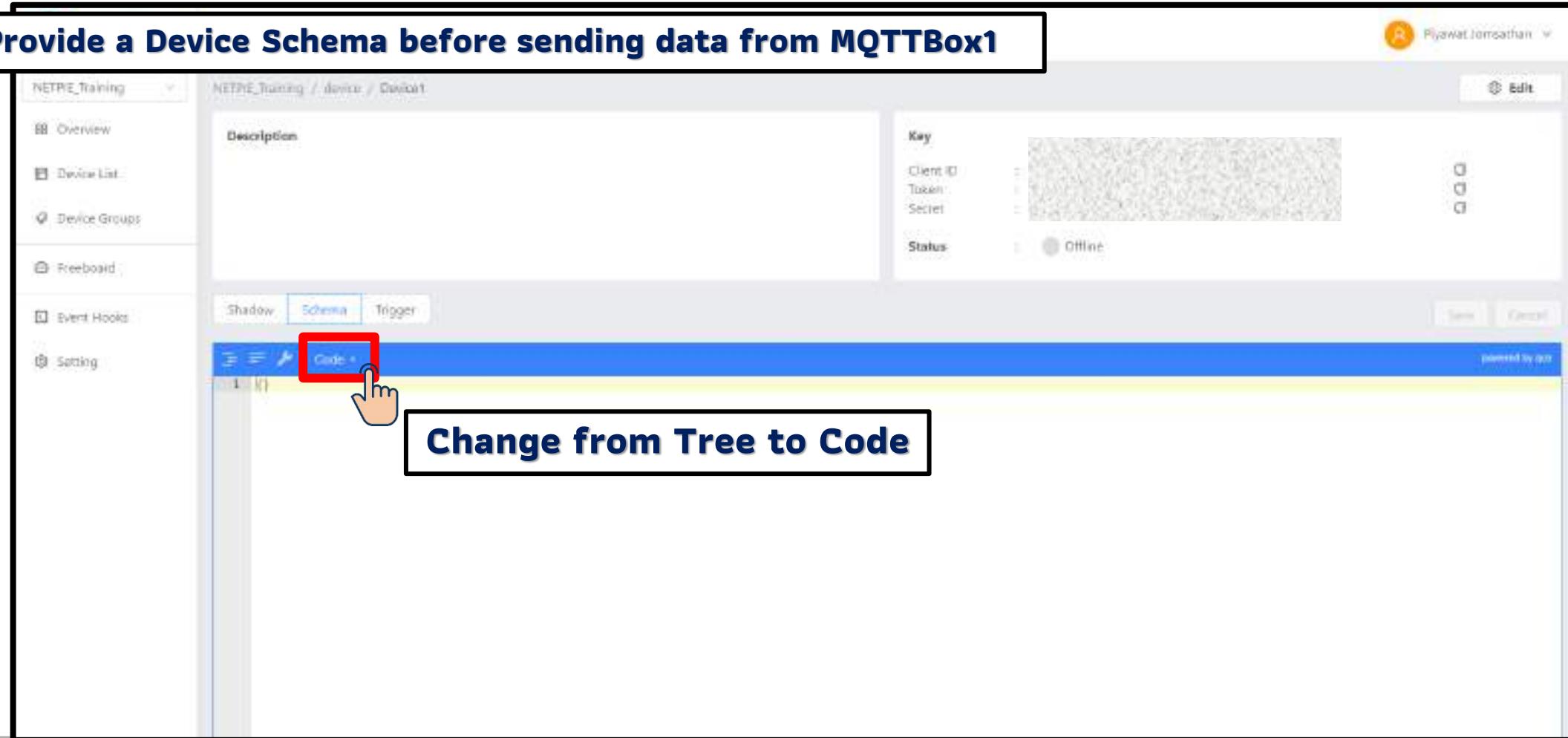
**Provide a Device Schema before sending data from MQTTBox1.**



# 4 - Data Management in NETPIE2020

## Exercise 5 : Set Schema and Send Data to Device Feed

**Provide a Device Schema before sending data from MQTTBox1**



# 4 - Data Management in NETPIE2020

## Exercise 5 : Set Schema and Send Data to Device Feed

### Device Schema in JSON format

```
{  
  "additionalProperties": false,  
  "properties": {  
    "light": {  
      "operation": {  
        "store": {  
          "ttl": "7d"  
        }  
      },  
      "type": "number"  
    },  
    "temperature": {  
      "operation": {  
        "store": {  
          "ttl": "7d"  
        }  
      },  
      "transform": {  
        "expression": "[[$.temperature]*1.8] + 32"  
      }  
    },  
    "type": "number"  
  }  
}
```

**Copy to Device Schema**

# 4 - Data Management in NETPIE2020

## Exercise 5 : Set Schema and Send Data to Device Feed



The screenshot shows the NETPIE Device Schema editor interface. At the top, there are tabs: Shadow, Schema (which is selected), Trigger, and Feed. Below the tabs is a toolbar with icons for Save, Cancel, and other functions. A large text area displays a JSON schema:

```
1+ {
2+   "additionalProperties": false,
3+   "properties": {
4+     "light": {
5+       "operation": {
6+         "store": {
7+           "ttl": "7d"
8+         }
9+       },
10+      "type": "number"
11+    },
12+    "temperature": {
13+      "operation": {
14+        "store": {
15+          "ttl": "7d"
16+        },
17+        "transform": {
18+          "expression": "(($.temperature)*1.8) + 32"
19+        }
20+      },
21+      "type": "number"
22+    }
23+  }
24+ }
```

A callout box with a black border and white background contains the text: "Click [Save] to save Device Schema". A hand cursor icon is pointing at the "Save" button in the top right corner of the editor window.

# 4 - Data Management in NETPIE2020

## Exercise 5 : Set Schema and Send Data to Device Feed

### Device Schema in JSON format

Set **additionalProperties = false**

In order not to save values other than listed in properties to Shadow.

The first variable is **light** with properties

- Store for 7 days
- Data type is number

```
{  
  "additionalProperties": false,  
  "properties": {  
    "light": {  
      "operation": {  
        "store": {  
          "ttl": "7d"  
        }  
      },  
      "type": "number"  
    },  
    "temperature": {  
      "operation": {  
        "store": {  
          "ttl": "7d"  
        }  
      },  
      "transform": {  
        "expression": "[[$.temperature]*1.8] + 32"  
      },  
      "type": "number"  
    }  
  }  
}
```

# 4 - Data Management in NETPIE2020

## Exercise 5 : Set Schema and Send Data to Device Feed

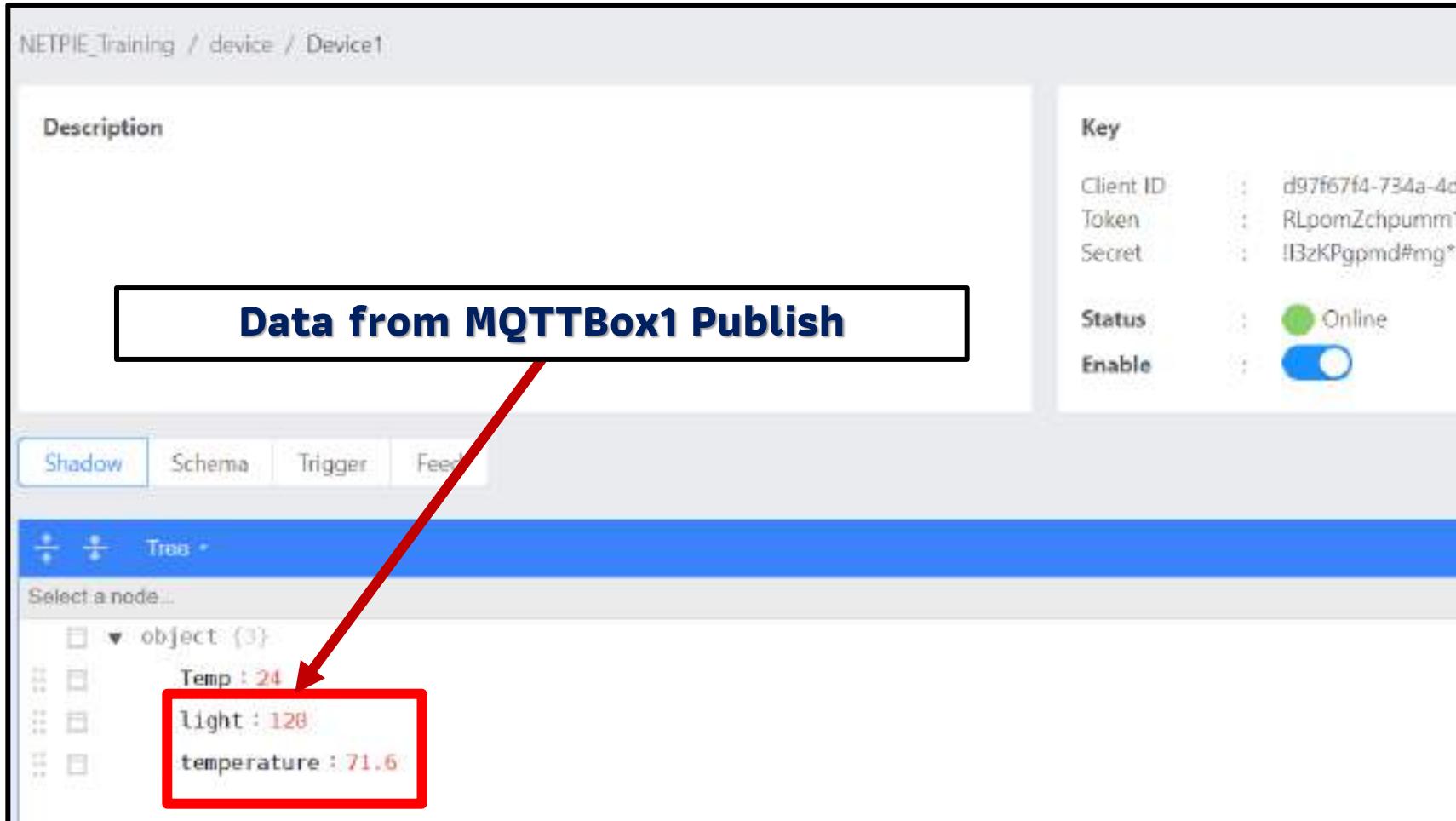
### Device Schema in JSON format

```
{  
  "additionalProperties": false,  
  "properties": {  
    "light": {  
      "operation": {  
        "store": {  
          "ttl": "7d"  
        }  
      },  
      "type": "number"  
    },  
    "temperature": {  
      "operation": {  
        "store": {  
          "ttl": "7d"  
        },  
        "transform": {  
          "expression": "[[$.temperature]*1.8] + 32"  
        },  
        "type": "number"  
      }  
    }  
  }  
}
```

Variable to store is **temperature** with properties  
- Store for 7 days  
- Convert from Celsius to Fahrenheit  
- Variable type is number

# 4 - Data Management in NETPIE2020

## Exercise 5 : Set Schema and Send Data to Device Feed



# 4 - Data Management in NETPIE2020

## Exercise 5 : Set Schema and Send Data to Device Feed



5

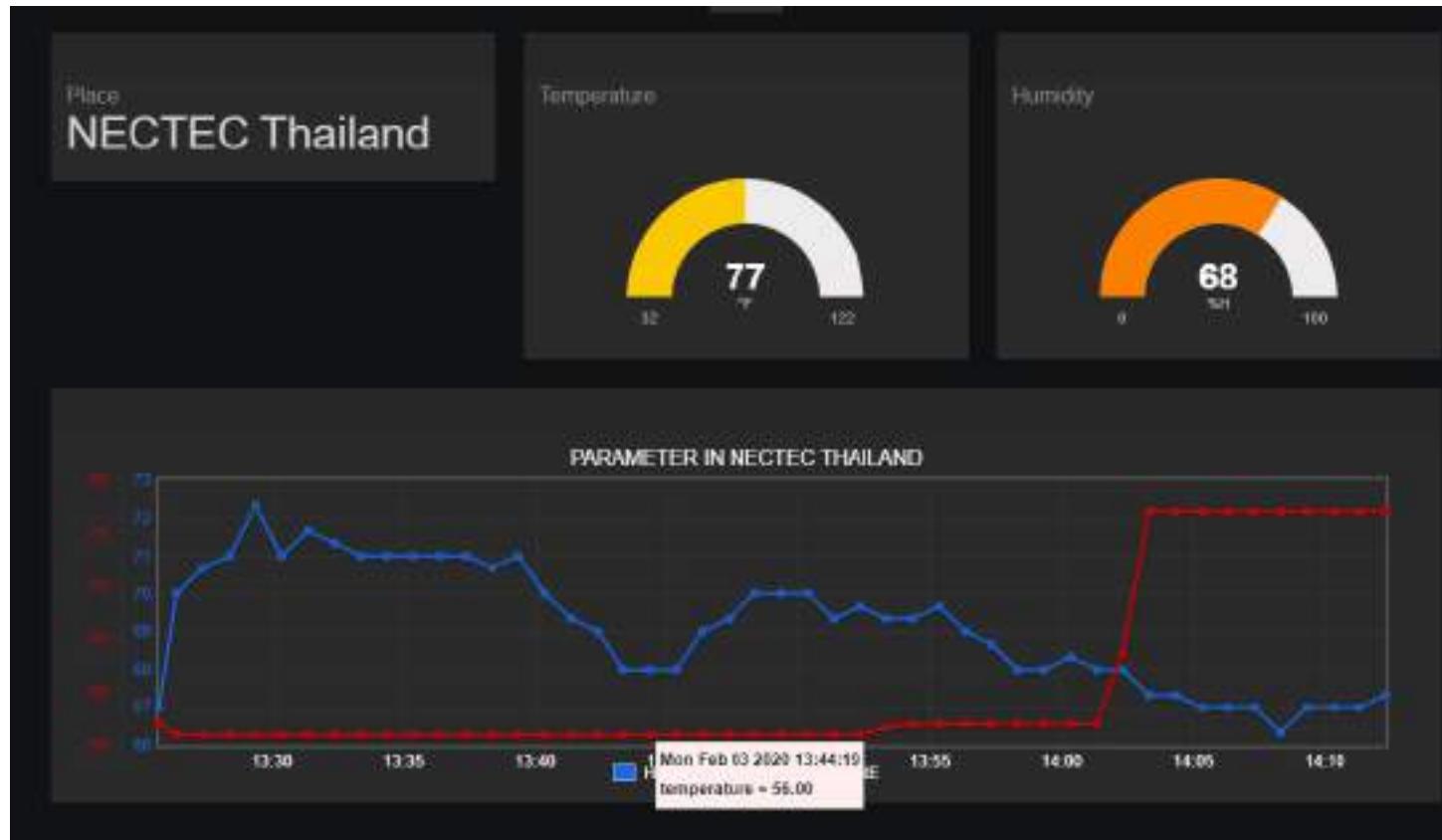
# Freeboard in NETPIE2020



# 5 - Freeboard in NETPIE2020

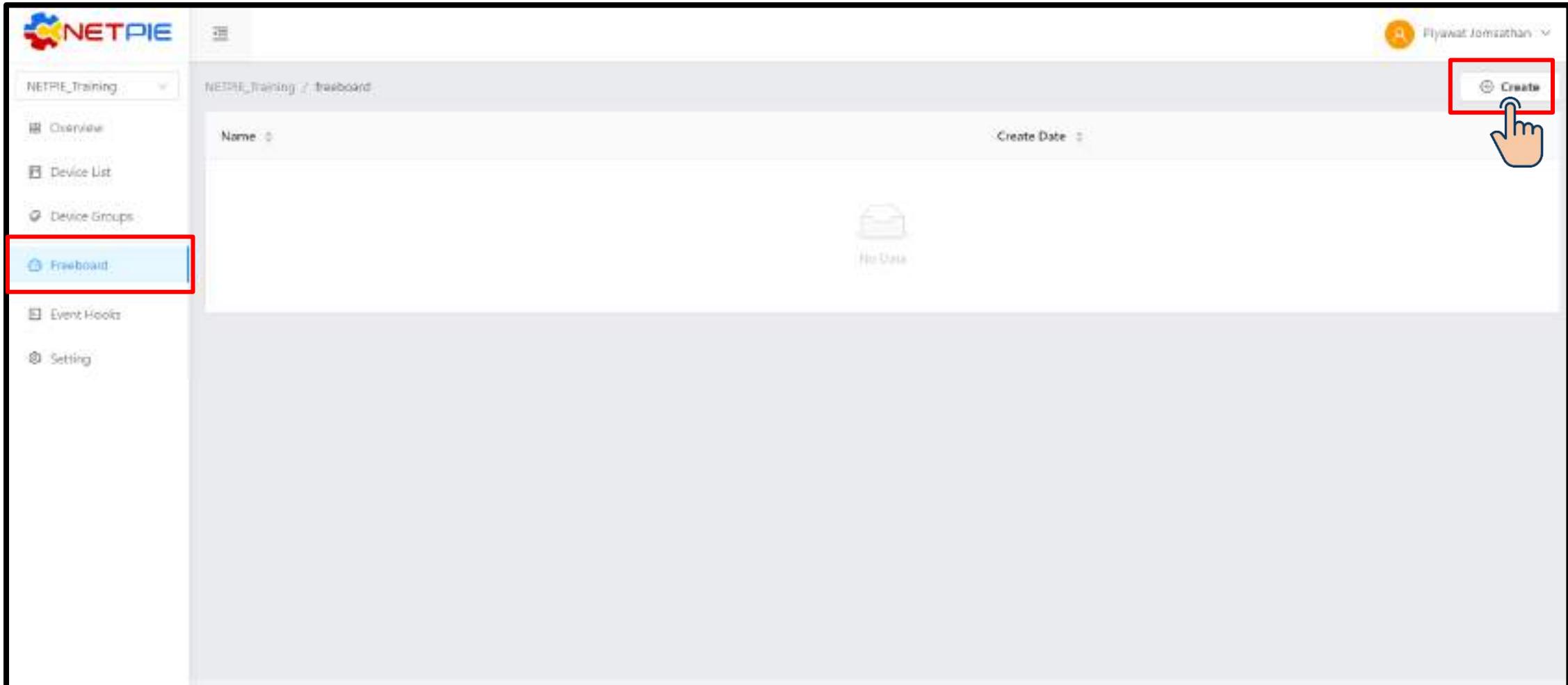
## What is Freeboard?

NETPIE2020 Freeboard is a software panel for control and display data retrieved from Device Shadow. A developer creates and customizes Widget Plugins to suit user requirements, such as gauges, sliders, control buttons, and put Javascript commands for various actions.



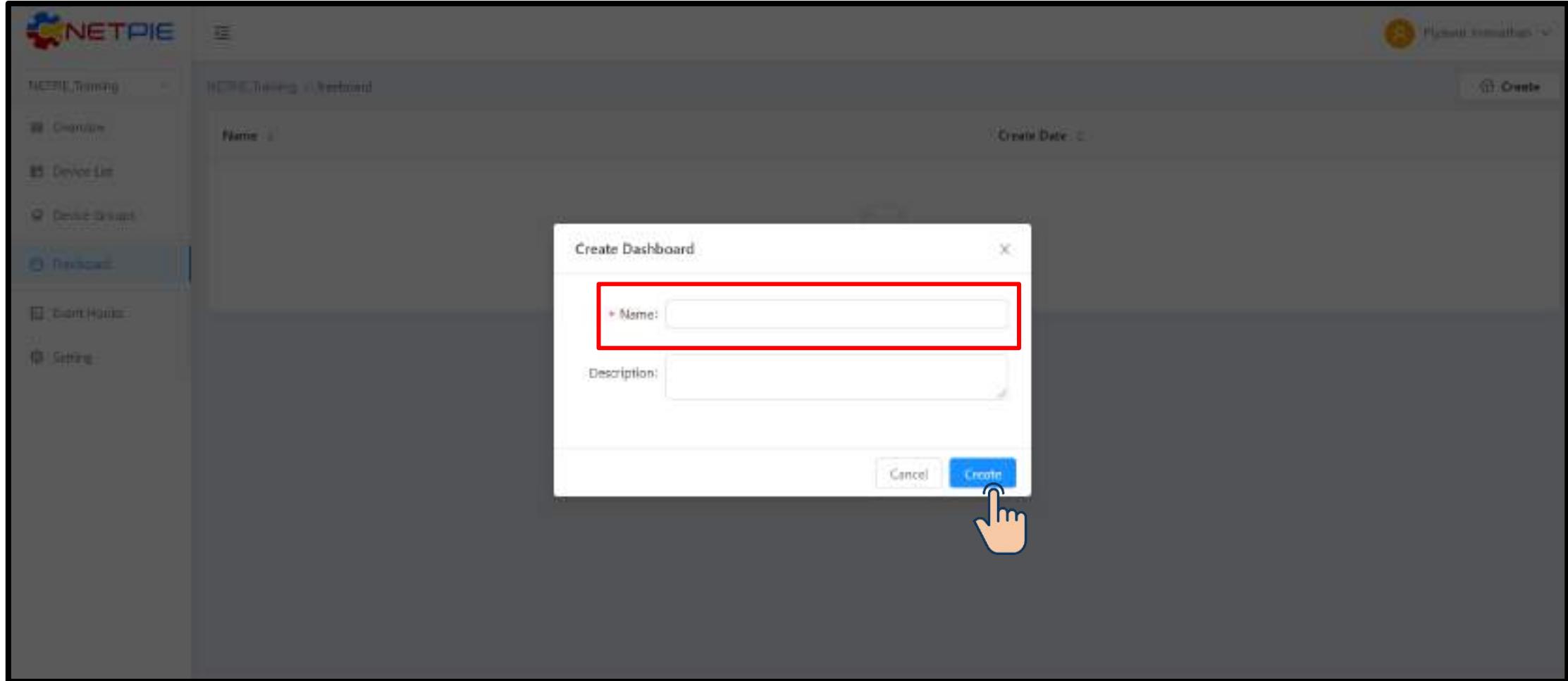
# 5 - Freeboard in NETPIE2020

## Create a Freeboard



# 5 - Freeboard in NETPIE2020

## Create a Freeboard



# 5 - Freeboard in NETPIE2020

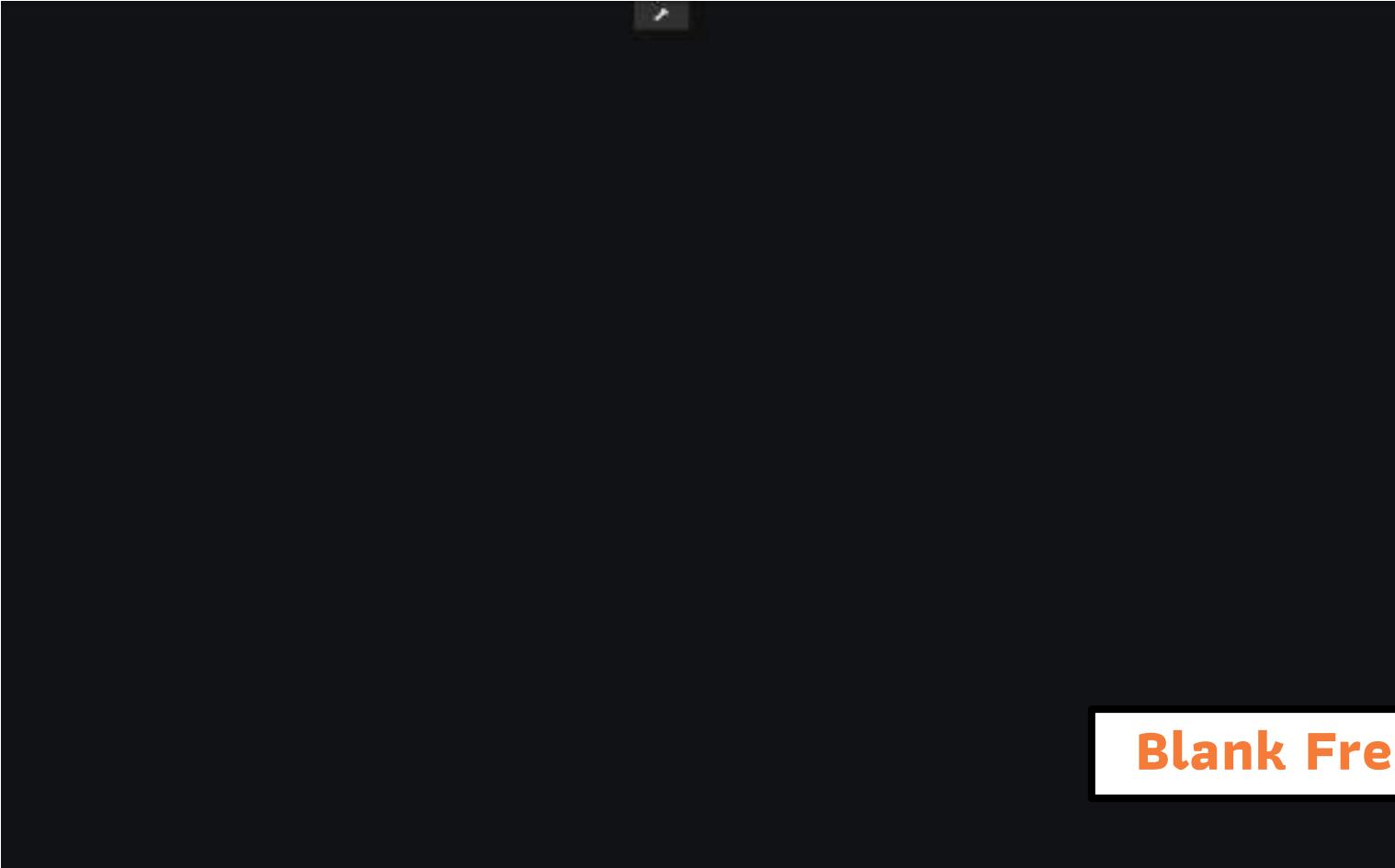
## Create a Freeboard

The screenshot shows the NETPIE2020 web interface. On the left, a sidebar menu includes 'Overview', 'Device List', 'Device Groups', 'Freeboard' (which is selected and highlighted in blue), 'Event Hooks', and 'Setting'. The main content area is titled 'NETPIE\_Training / freeboard' and displays a table with one item: 'Freeboard1' created on '2020-05-09 12:45'. A red box highlights the first row of the table, and a hand cursor icon is positioned over the 'Freeboard1' entry. The top right corner shows the user 'Piyawat Jomsathan' and a 'Create' button.

# 5 - Freeboard in NETPIE2020

---

Create a Freeboard



Blank Freeboard window

# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



MQTTBox1

**@shadow/data/update**

**Publish : { "data": { "temperature" : 24, "light" : 80, "place" : "NECTEC" }}**



**NETPIE2020**



**"temperature" : 75.2  
"light" : 80  
"place": "NECTEC"**



# 5 - Freeboard in NETPIE2020

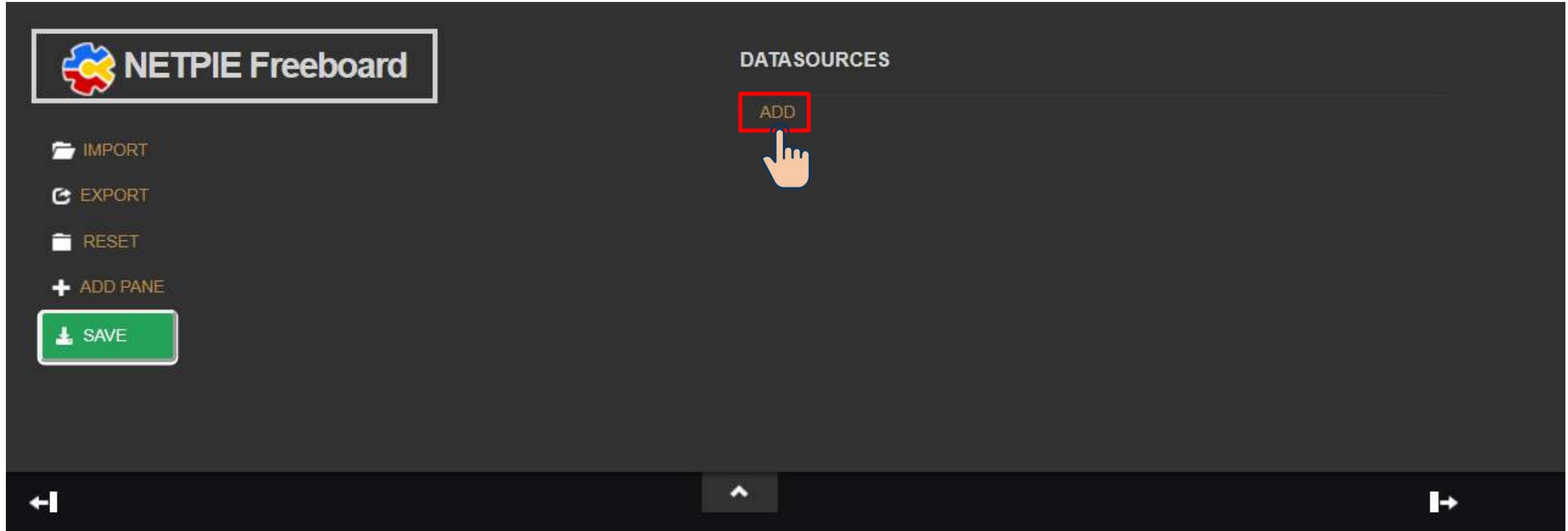
---

**Exercise 6 : Construct a Freeboard on NETPIE2020**



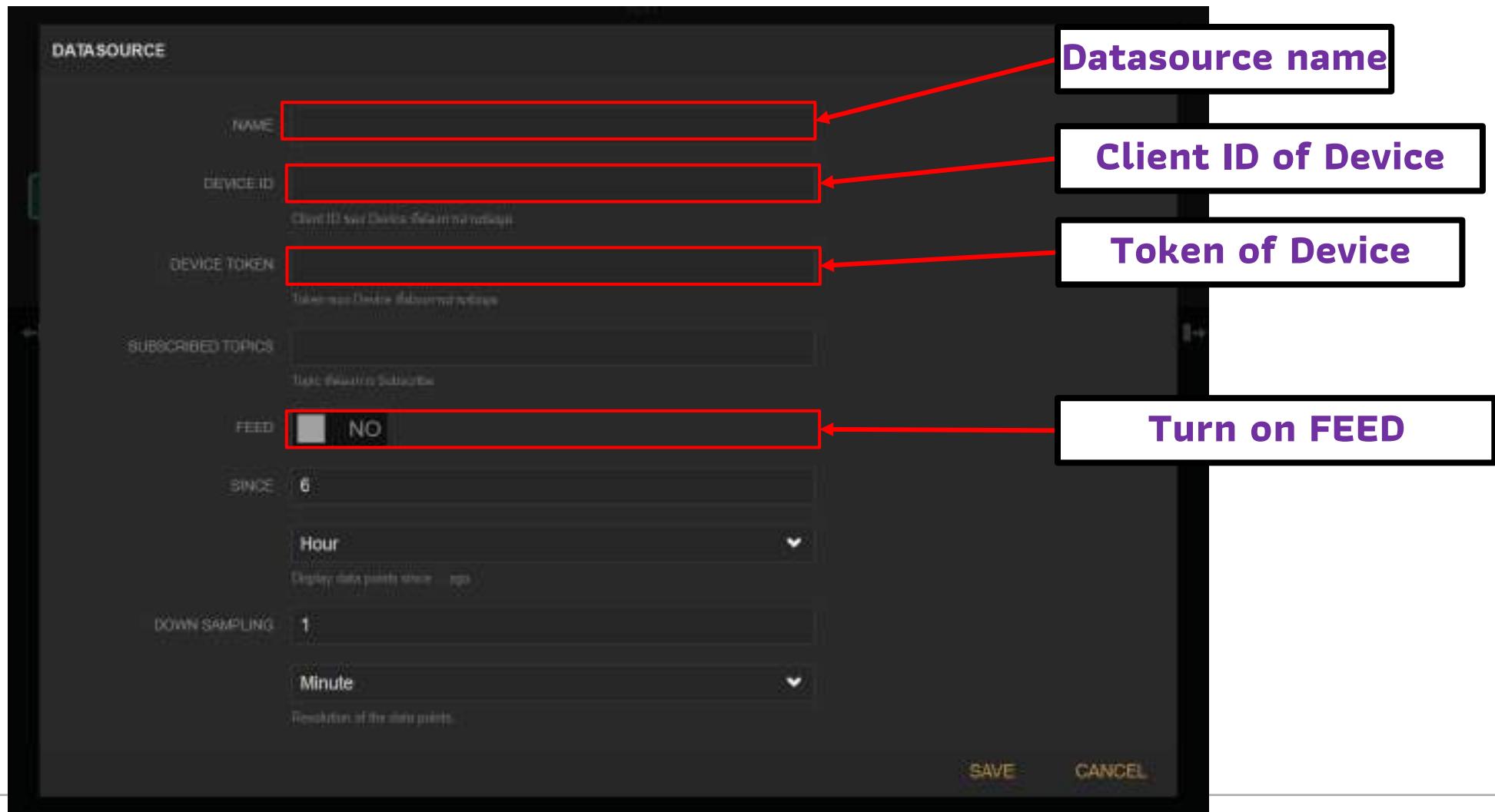
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



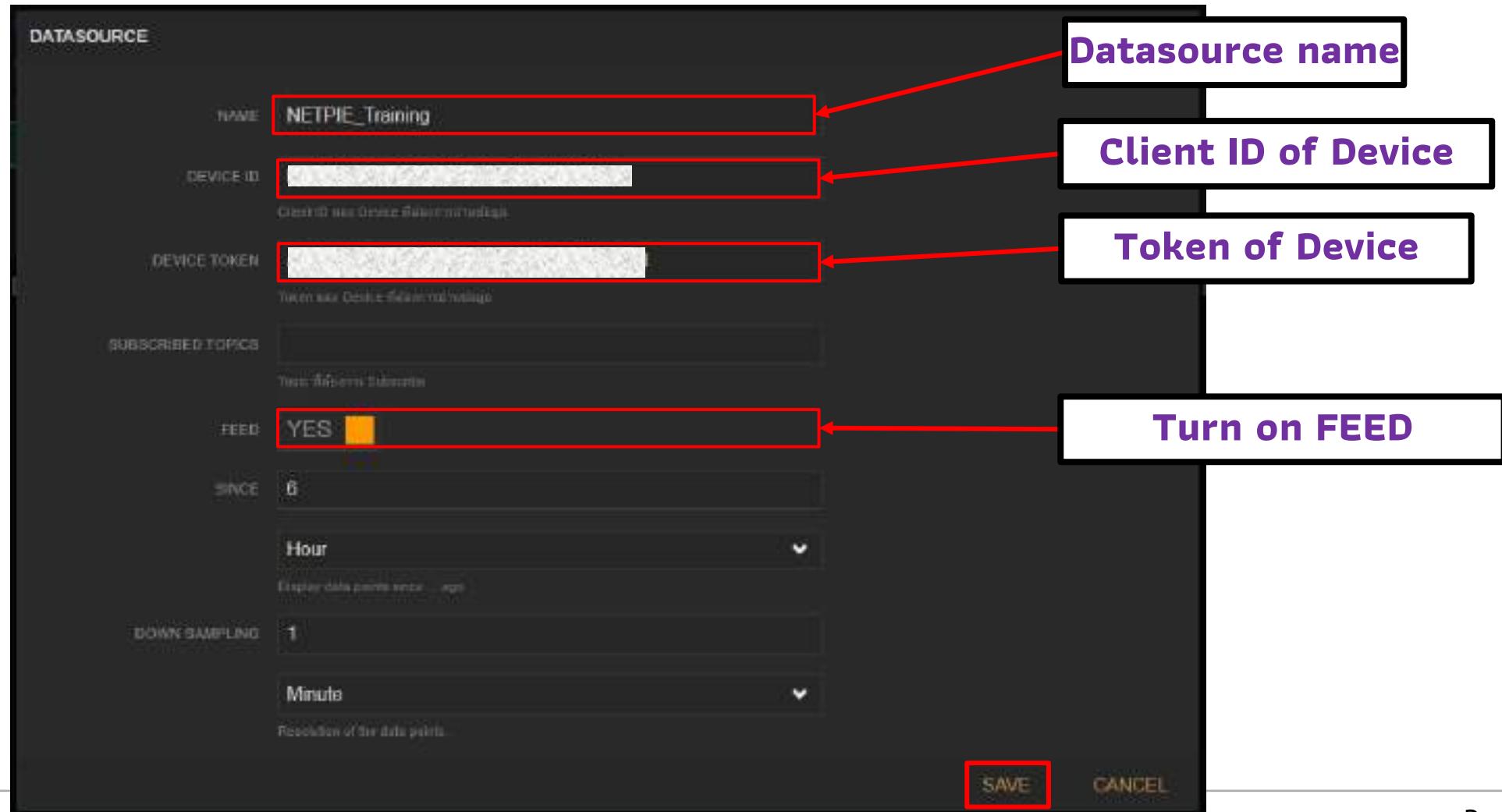
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020

The screenshot shows the NETPIE Freeboard application interface. On the left, there is a sidebar with icons for IMPORT, EXPORT, RESET, ADD PANE, and a large green SAVE button. The main area is titled "DATASOURCES" and contains a table with one row. The row has columns for "Name" (NETPIE\_Training) and "Last Updated" (8:41:38 PM). There are edit and delete icons next to the row. A red arrow points from a purple status message at the bottom right to the "NETPIE\_Training" entry in the table. The status message reads "Status shows successful connection". The bottom of the screen features navigation arrows and a scroll bar.

| Name            | Last Updated |
|-----------------|--------------|
| NETPIE_Training | 8:41:38 PM   |

Status shows successful connection

# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020

Shadow Schema Trigger Feed

Tree ▾

object ► temperature

object {4}

Temp : 24

light : 80

temperature : 75.2

place : NECTEC

The screenshot shows the NETPIE2020 interface with the 'Shadow' tab selected. A JSON object named 'temperature' is displayed under the 'object' section. The object contains four items: 'Temp' (value 24), 'light' (value 80), 'temperature' (value 75.2), and 'place' (value 'NECTEC'). Each of these properties has a red arrow pointing to its corresponding analog gauge display on the right side of the slide.

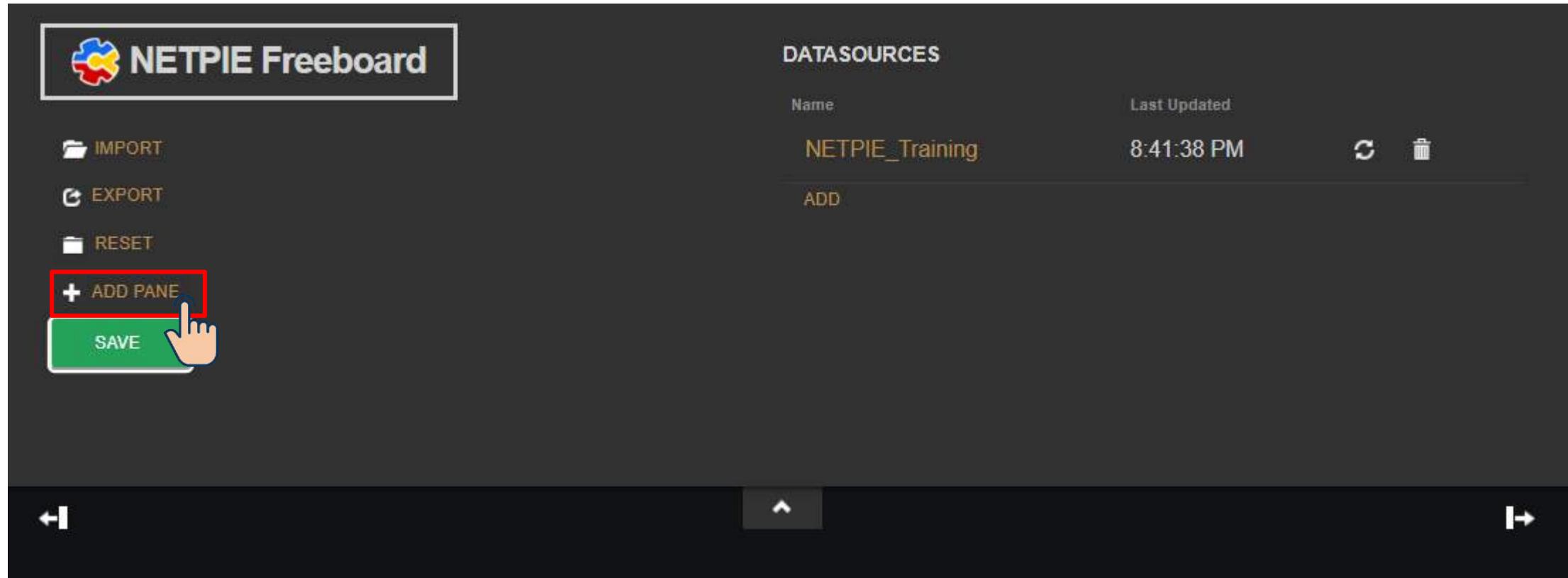


Place  
NECTEC

This figure shows a simple text display with the word 'Place' above 'NECTEC' in large white letters on a black background.

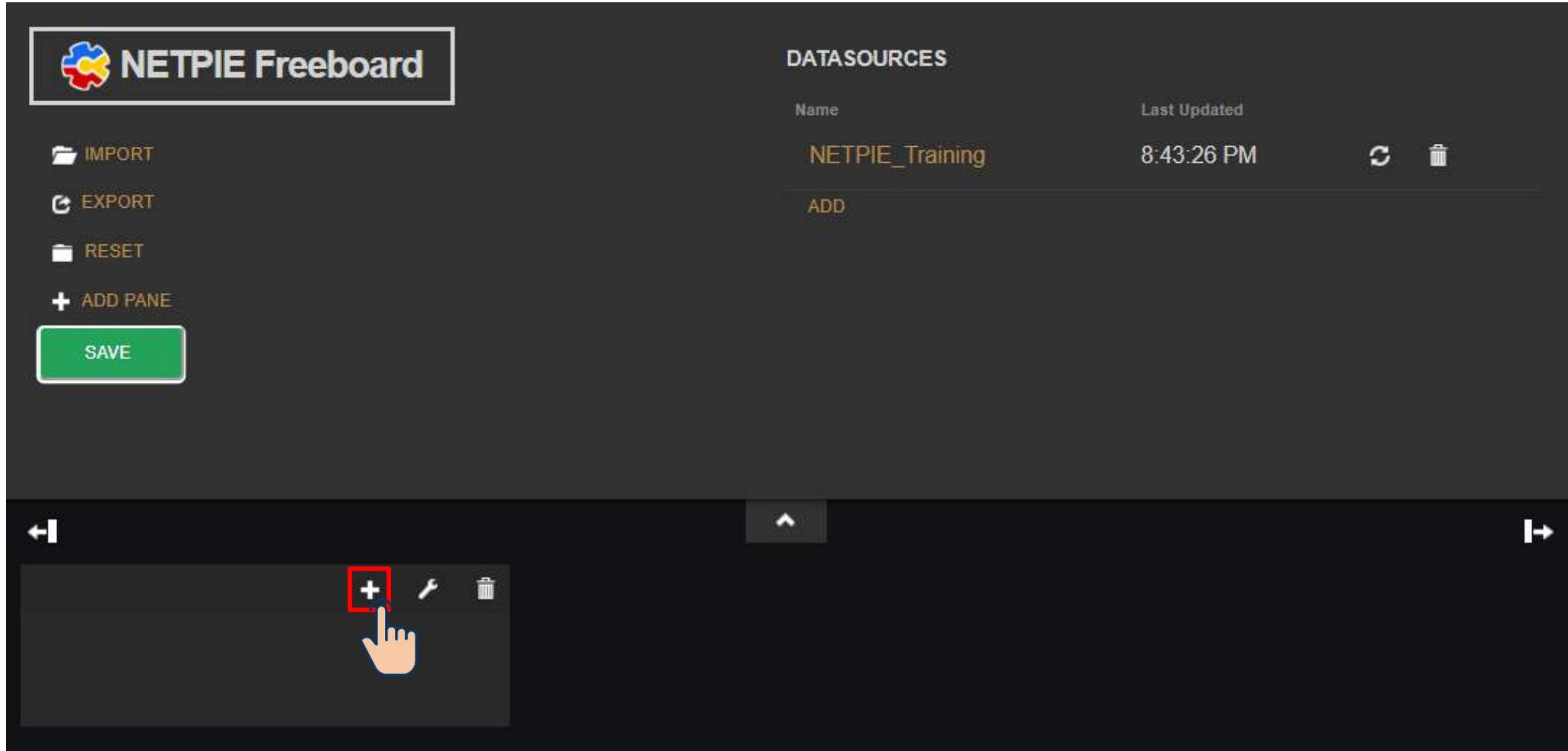
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



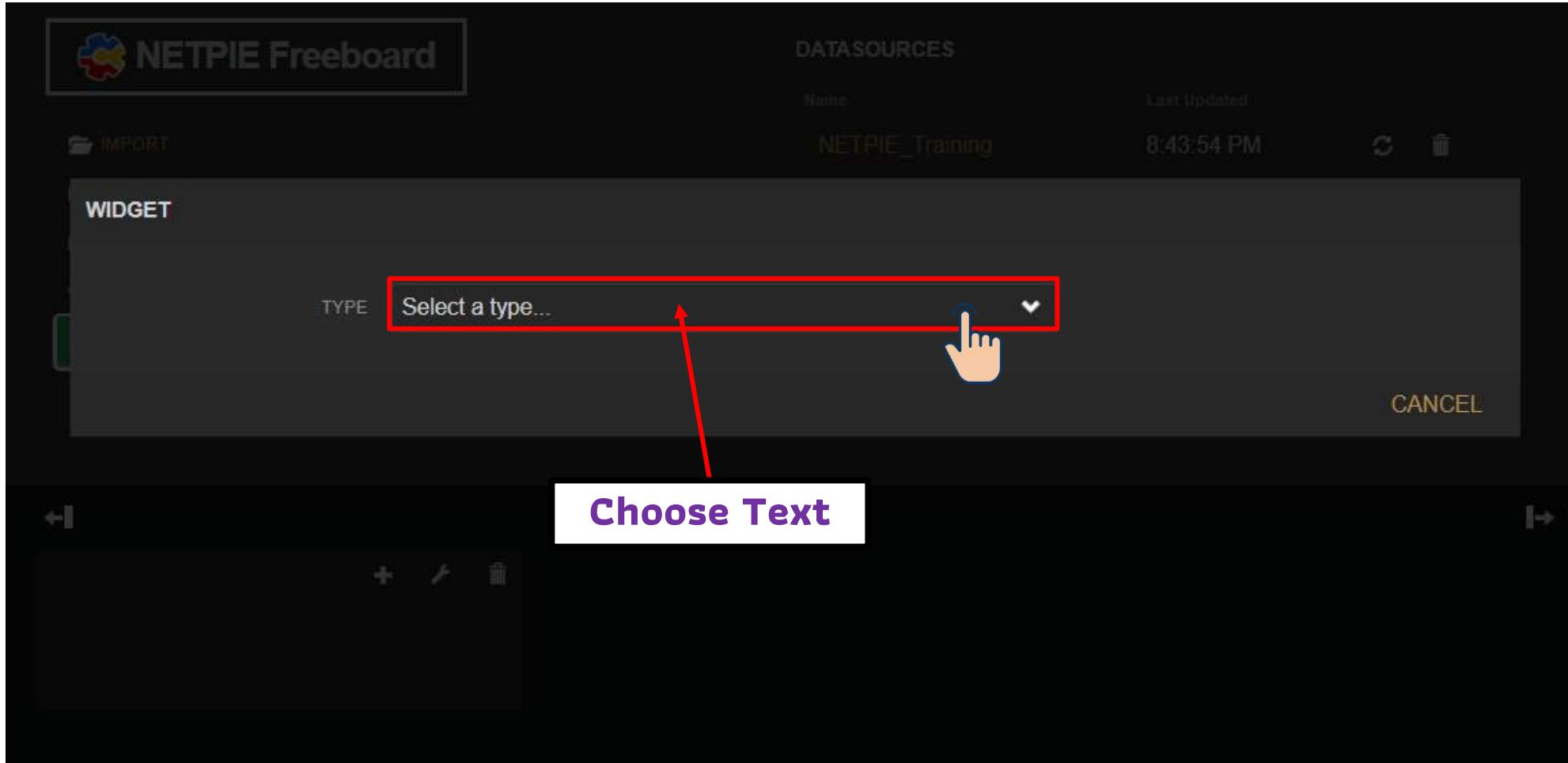
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



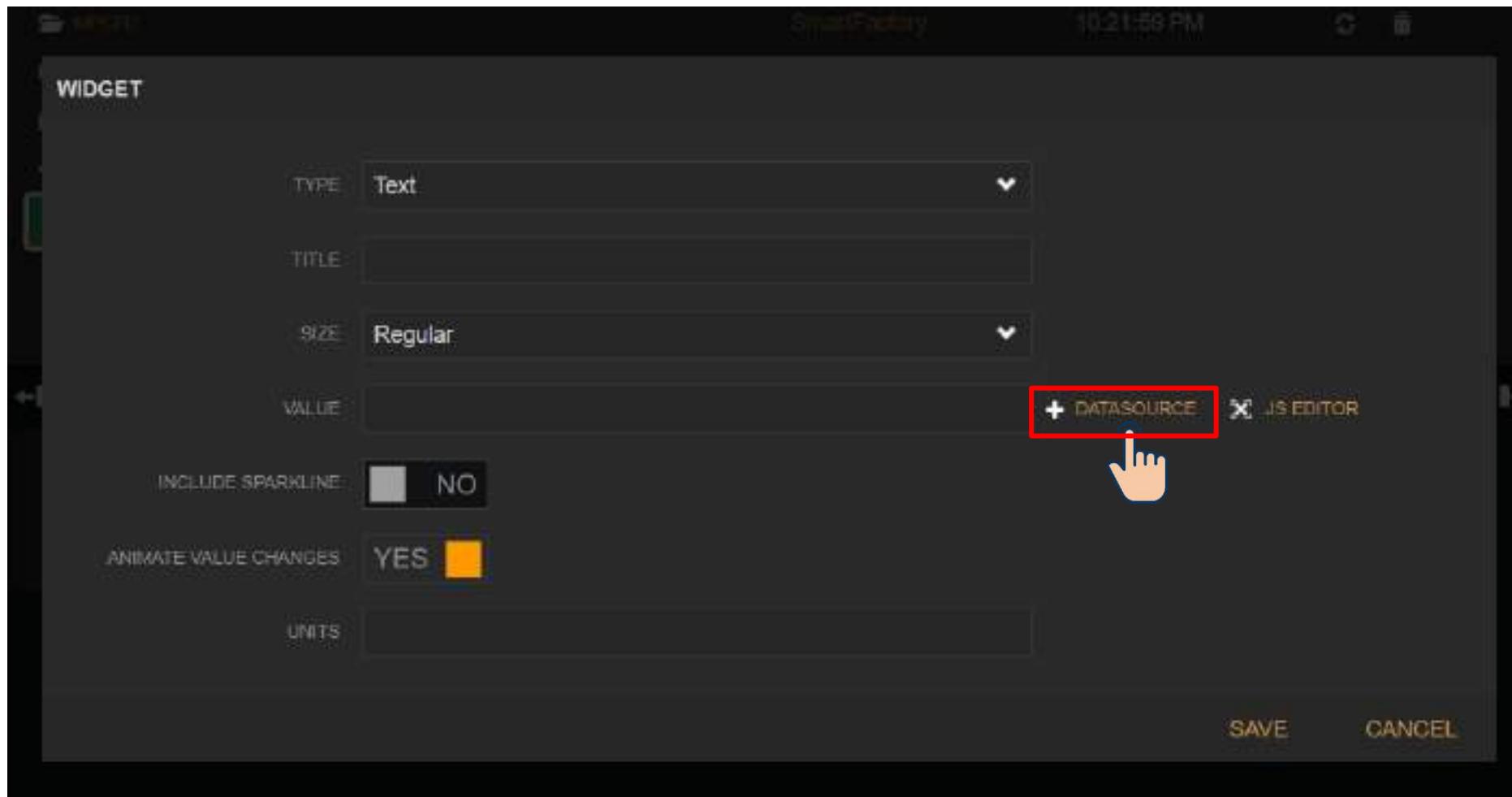
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



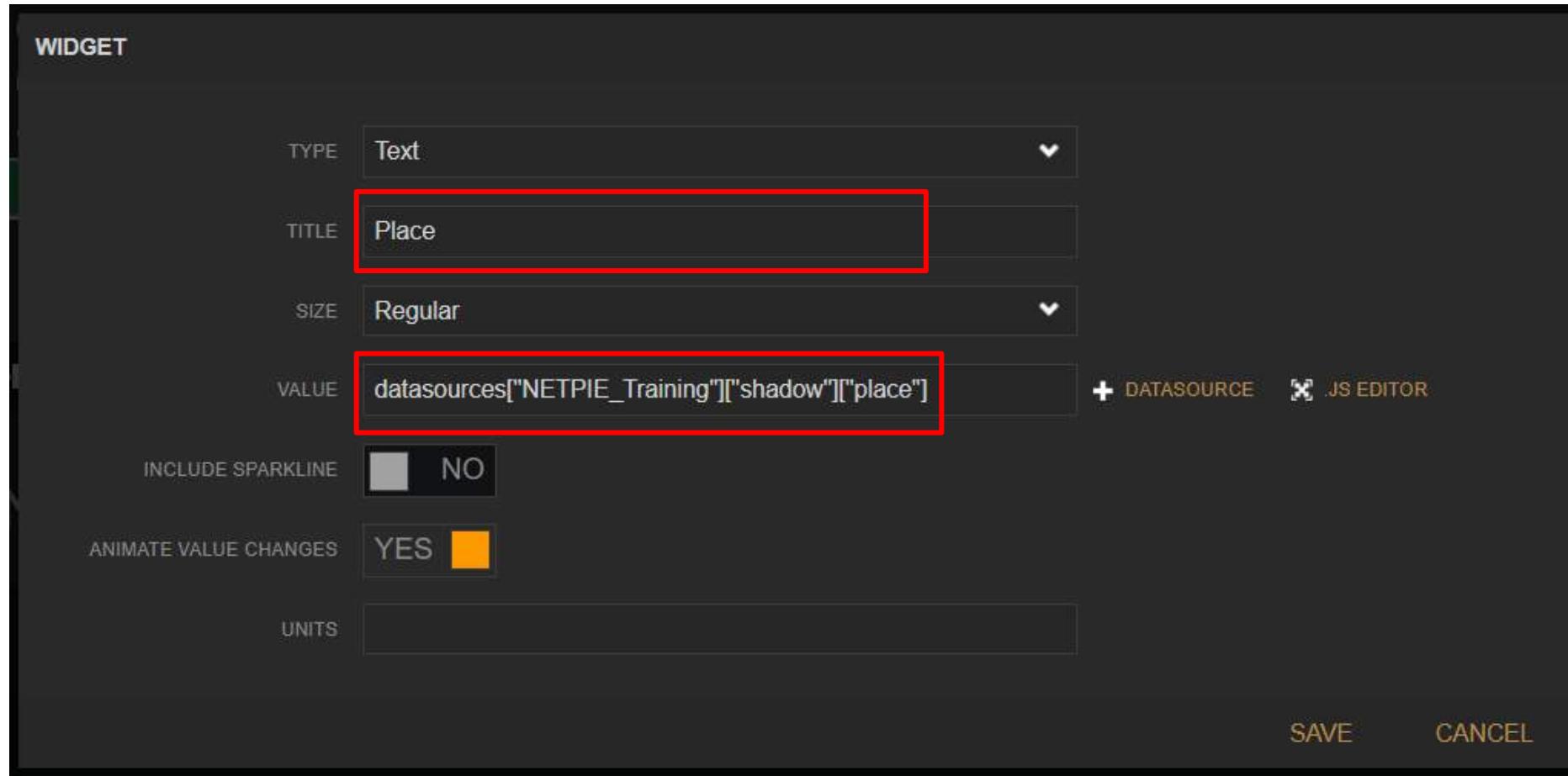
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



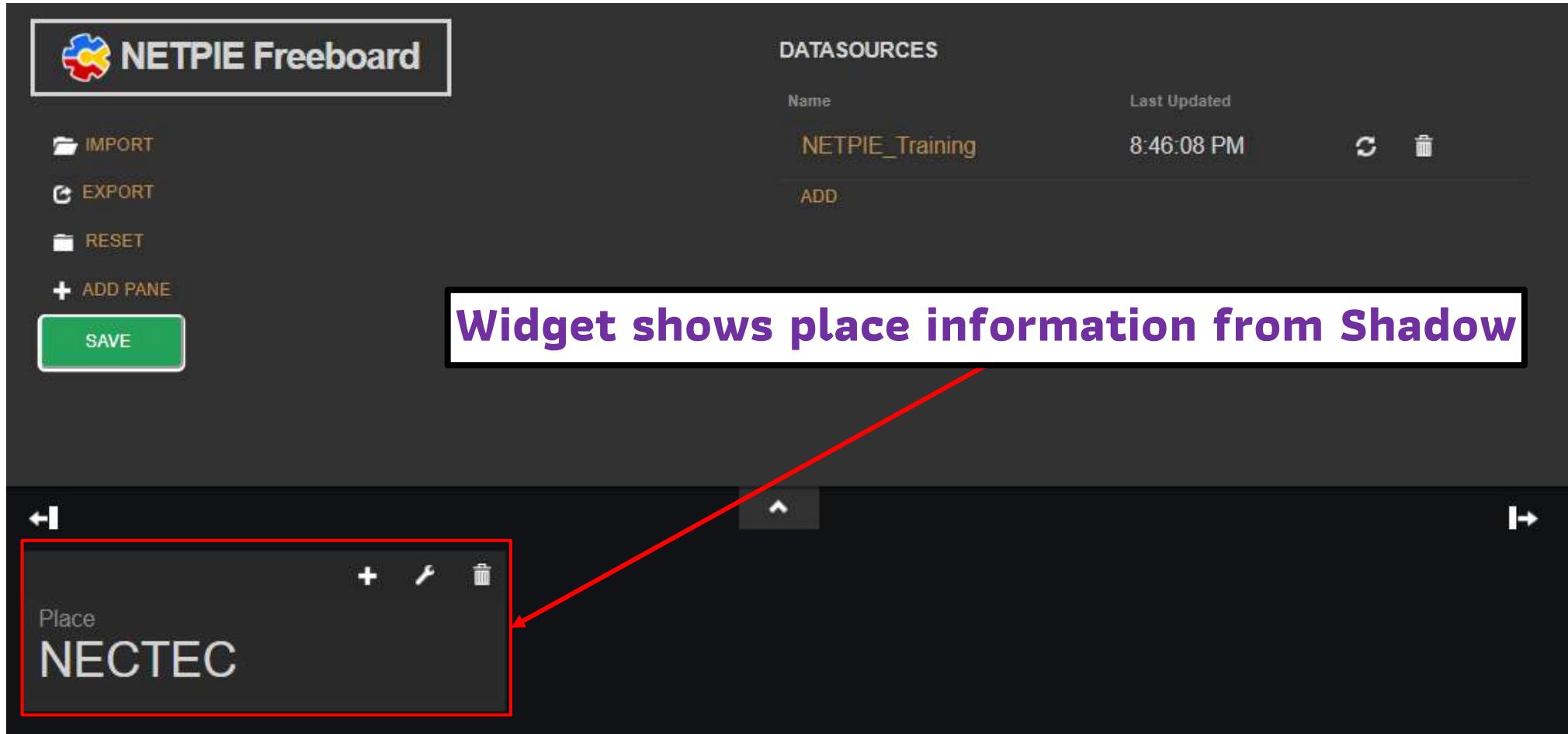
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



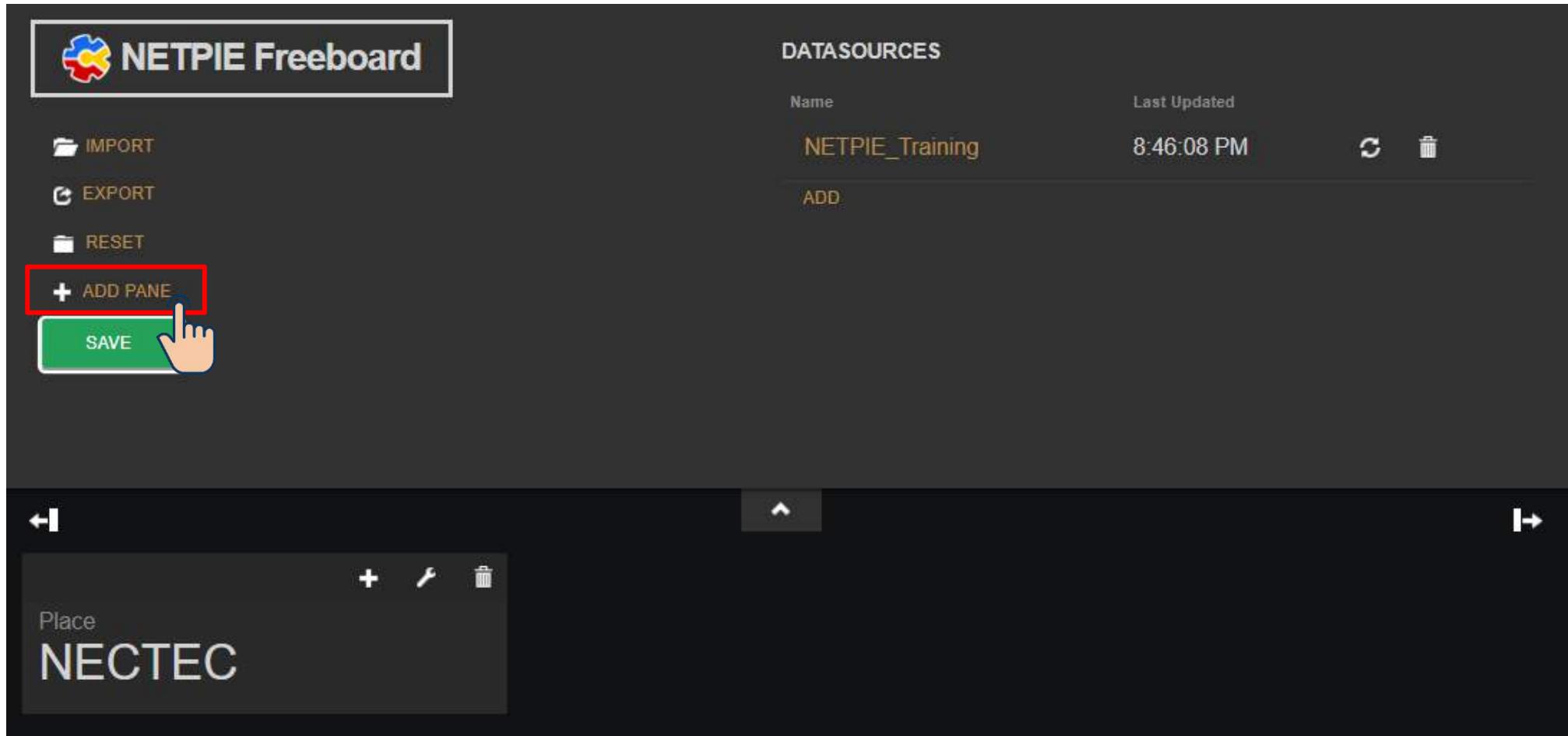
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



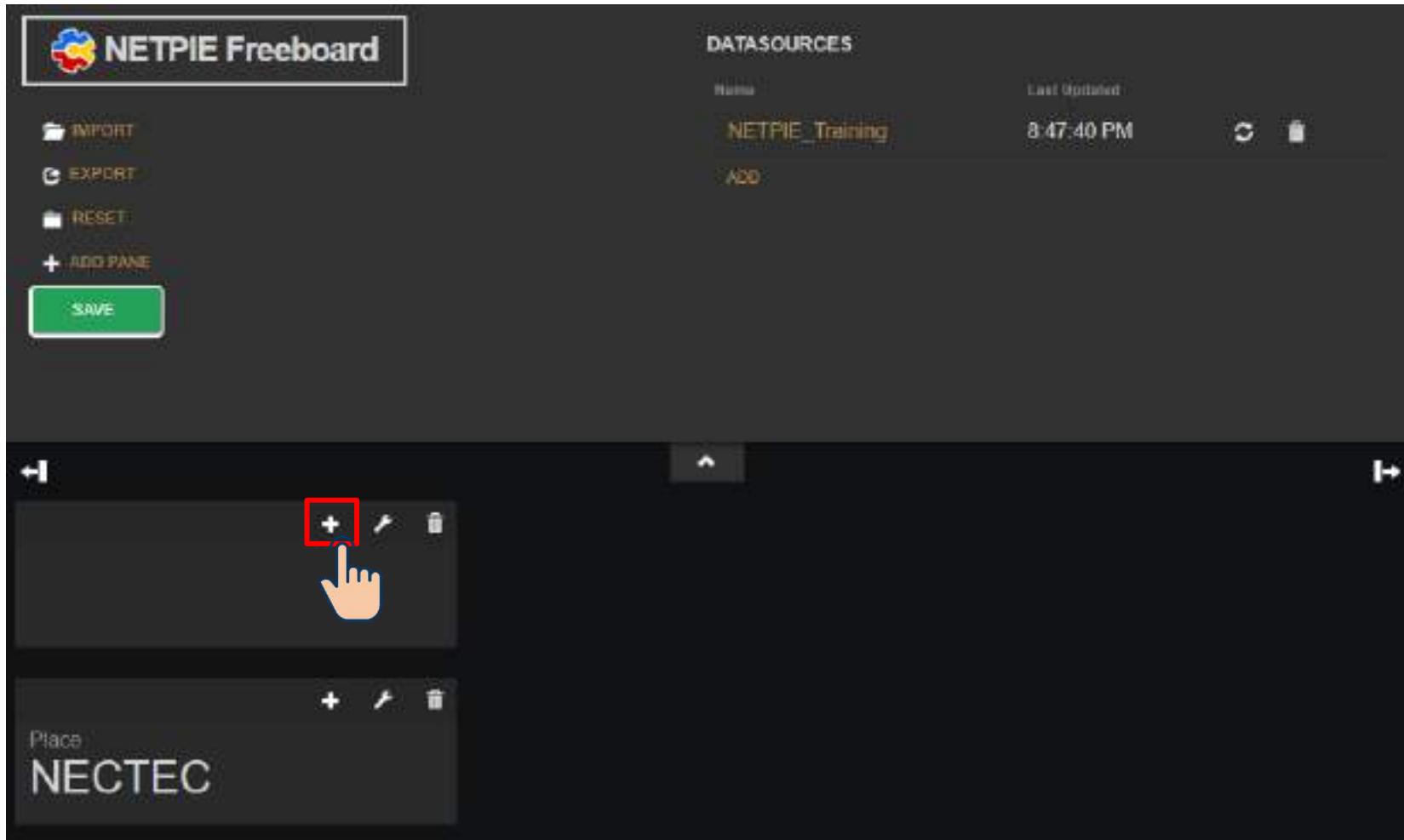
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



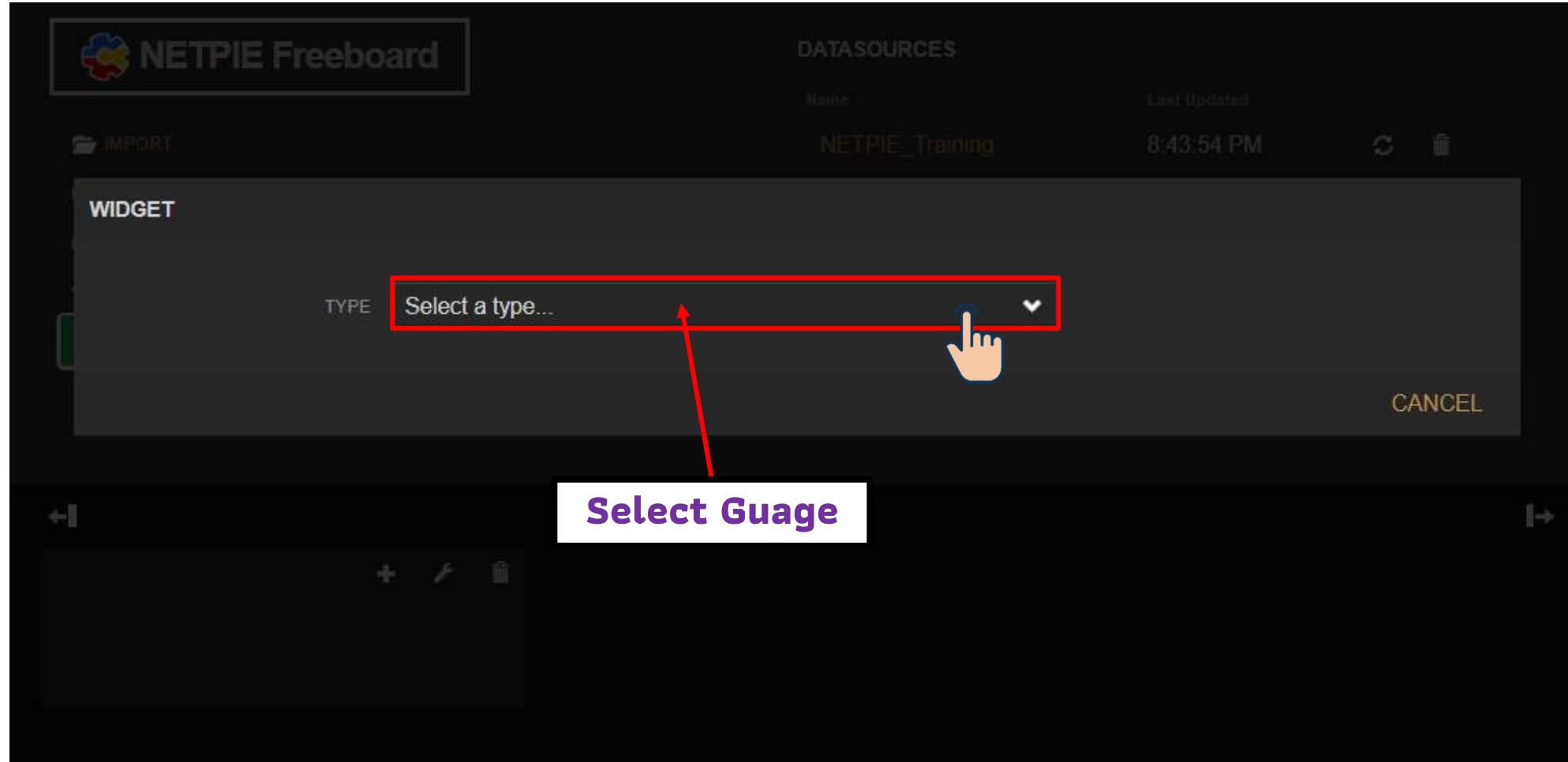
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



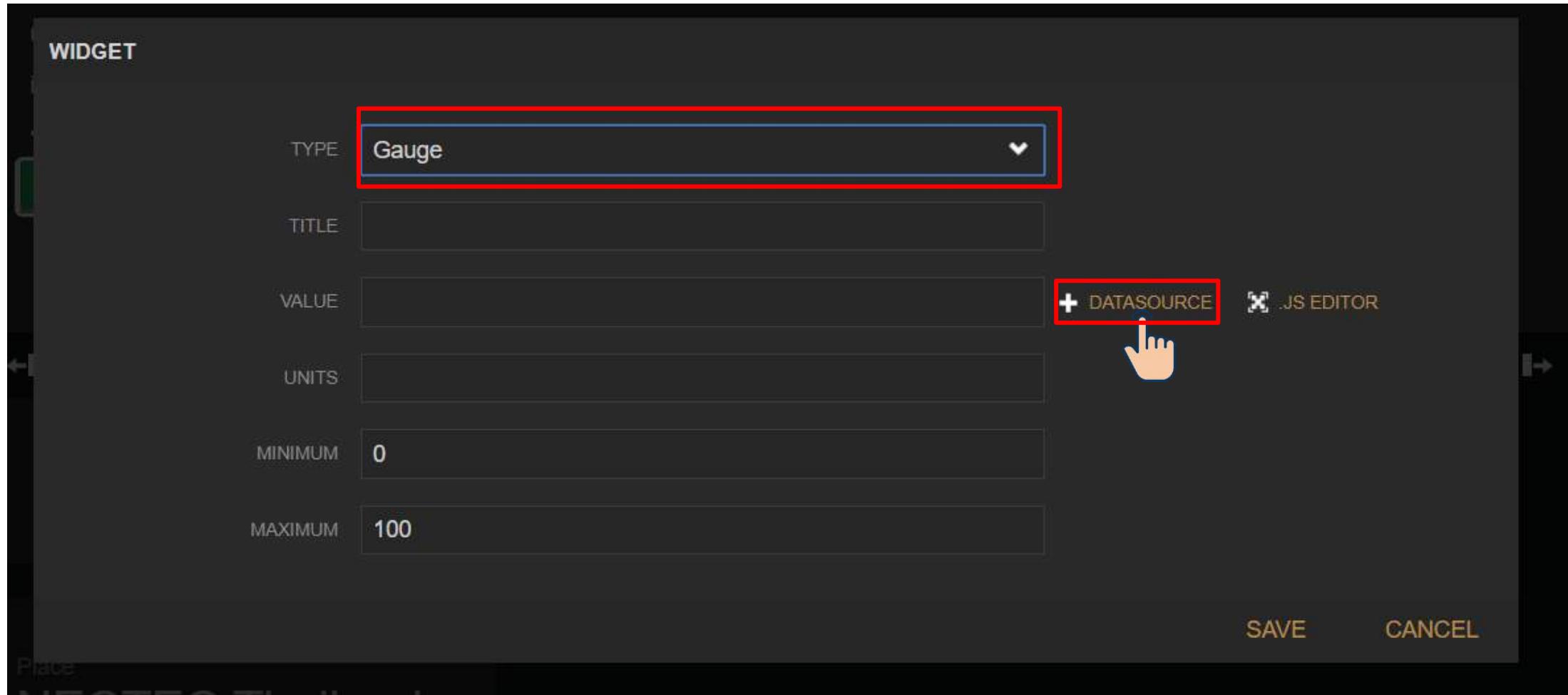
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



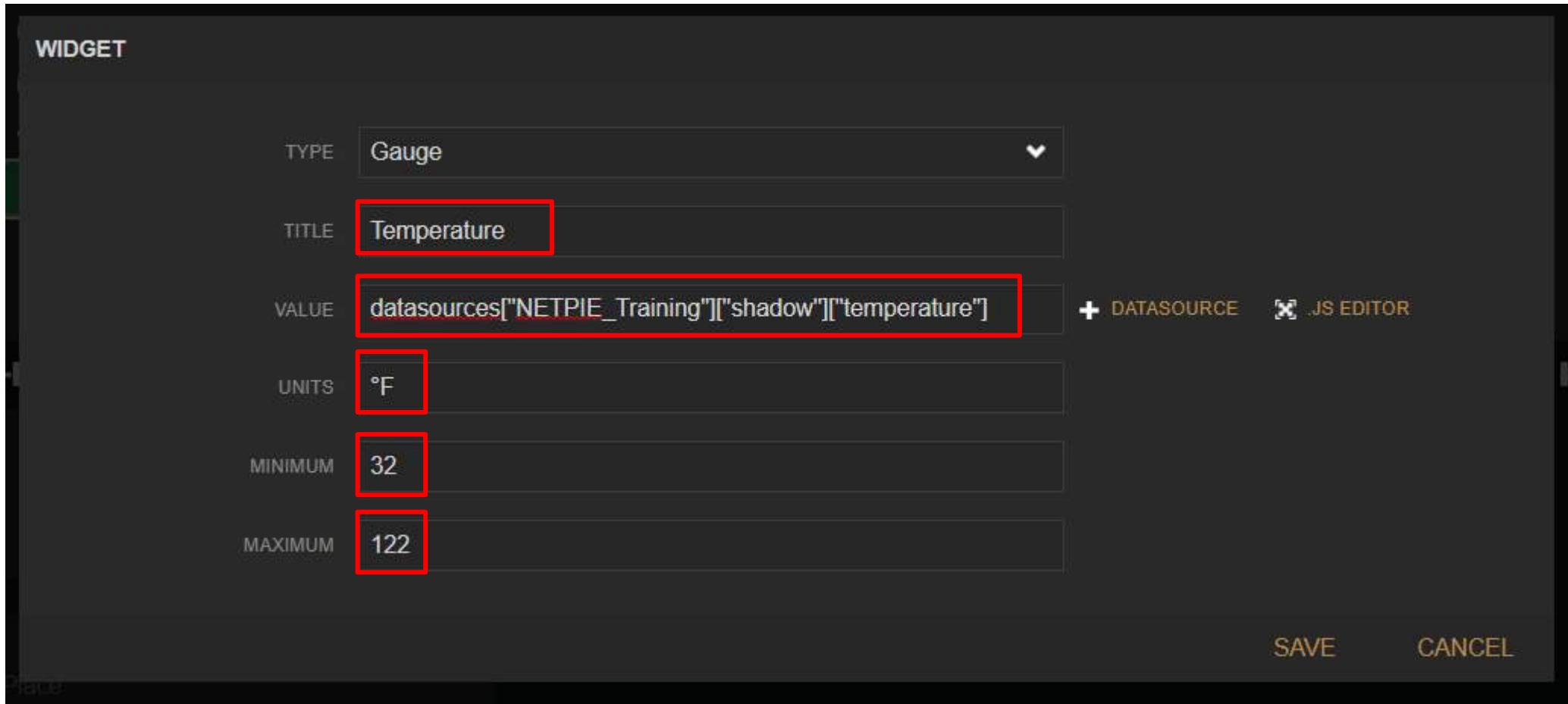
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



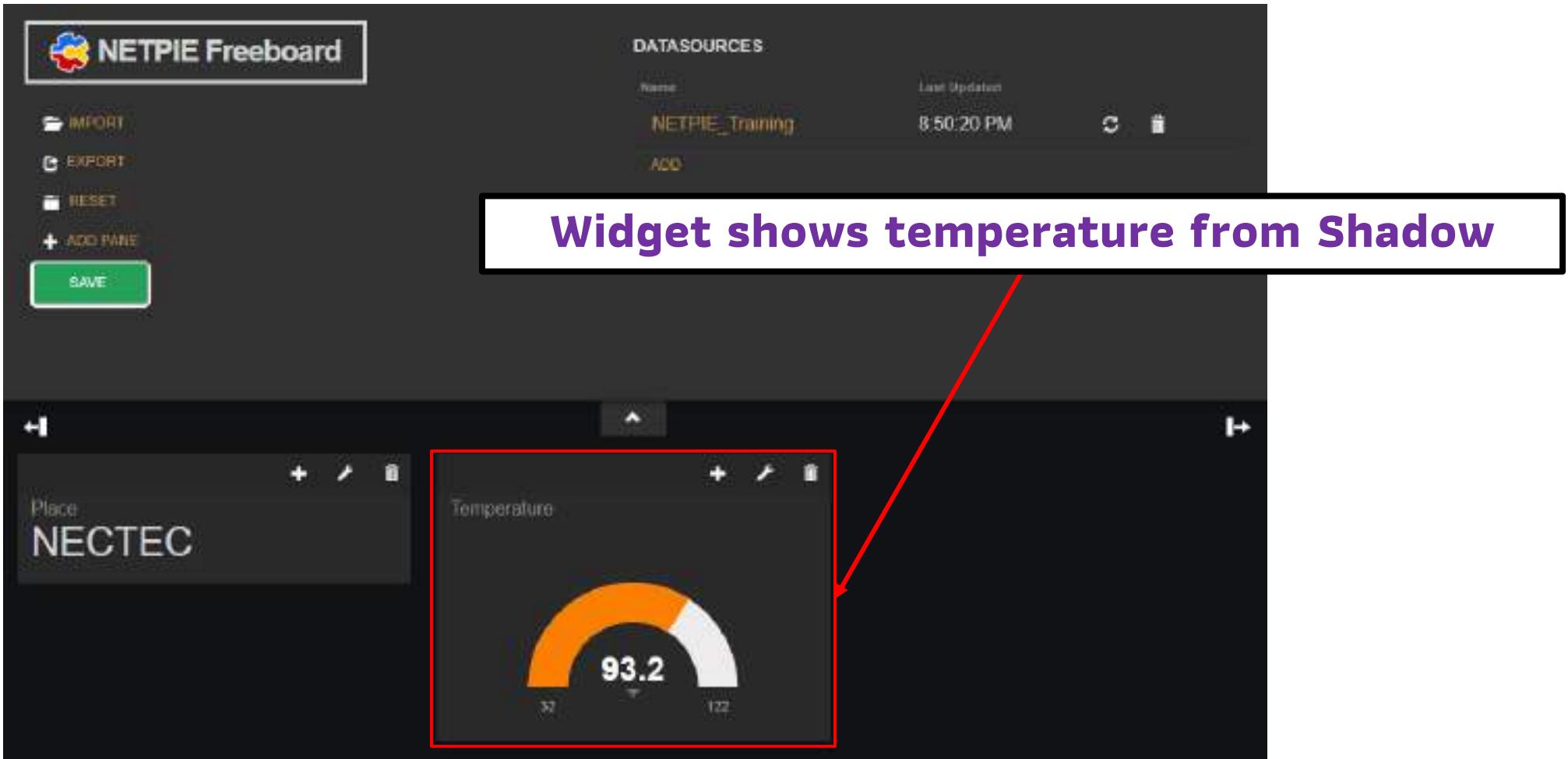
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



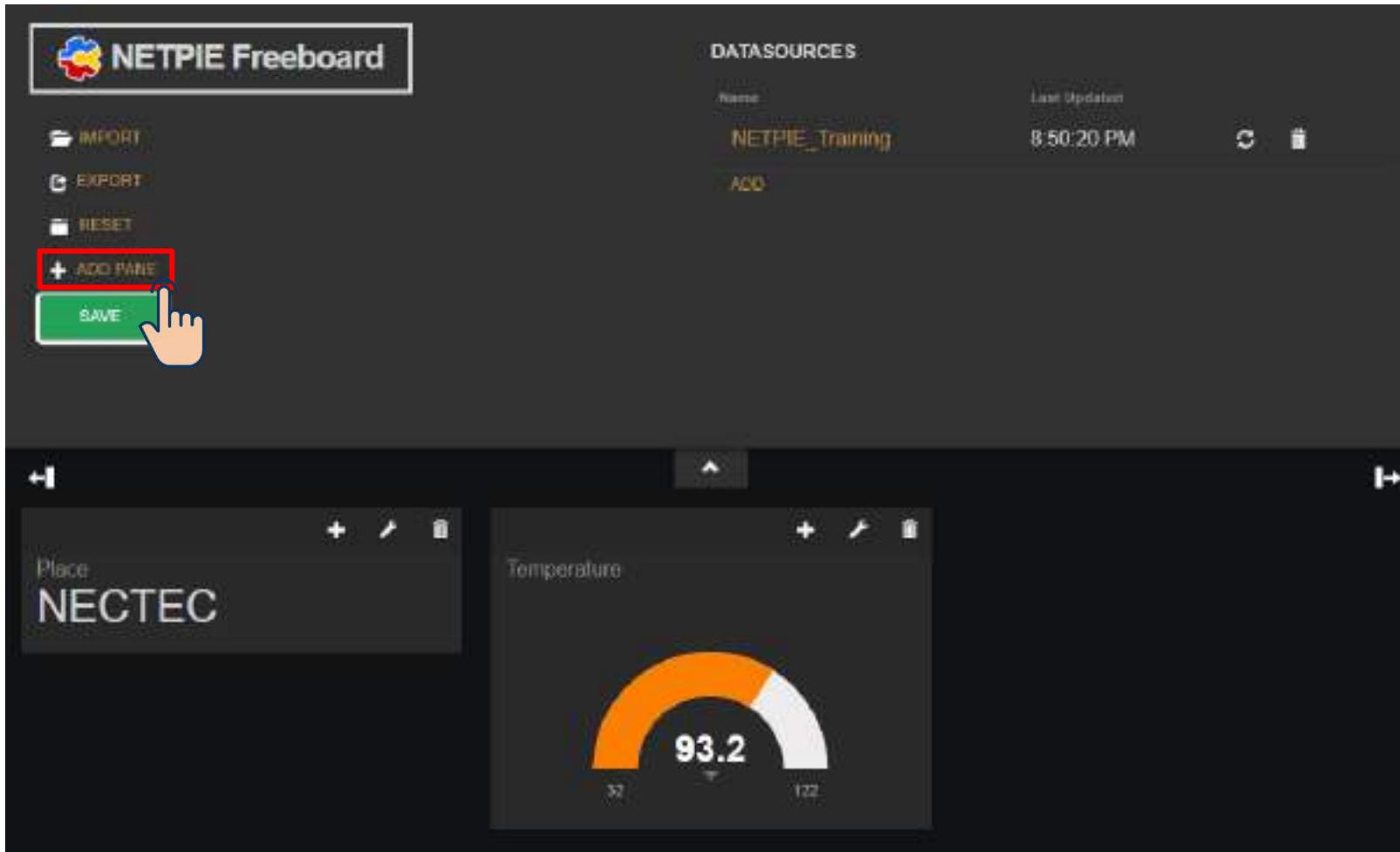
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



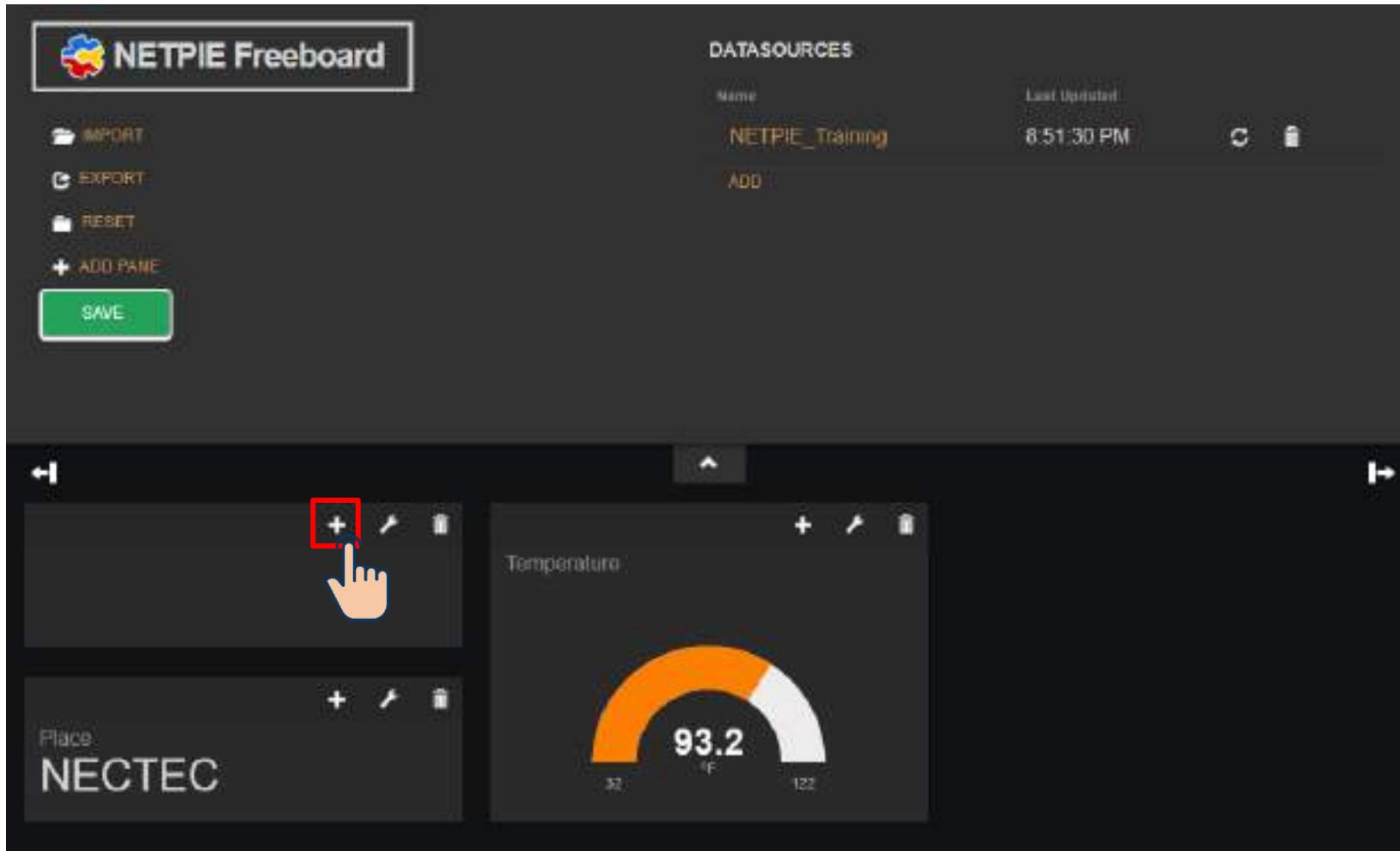
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



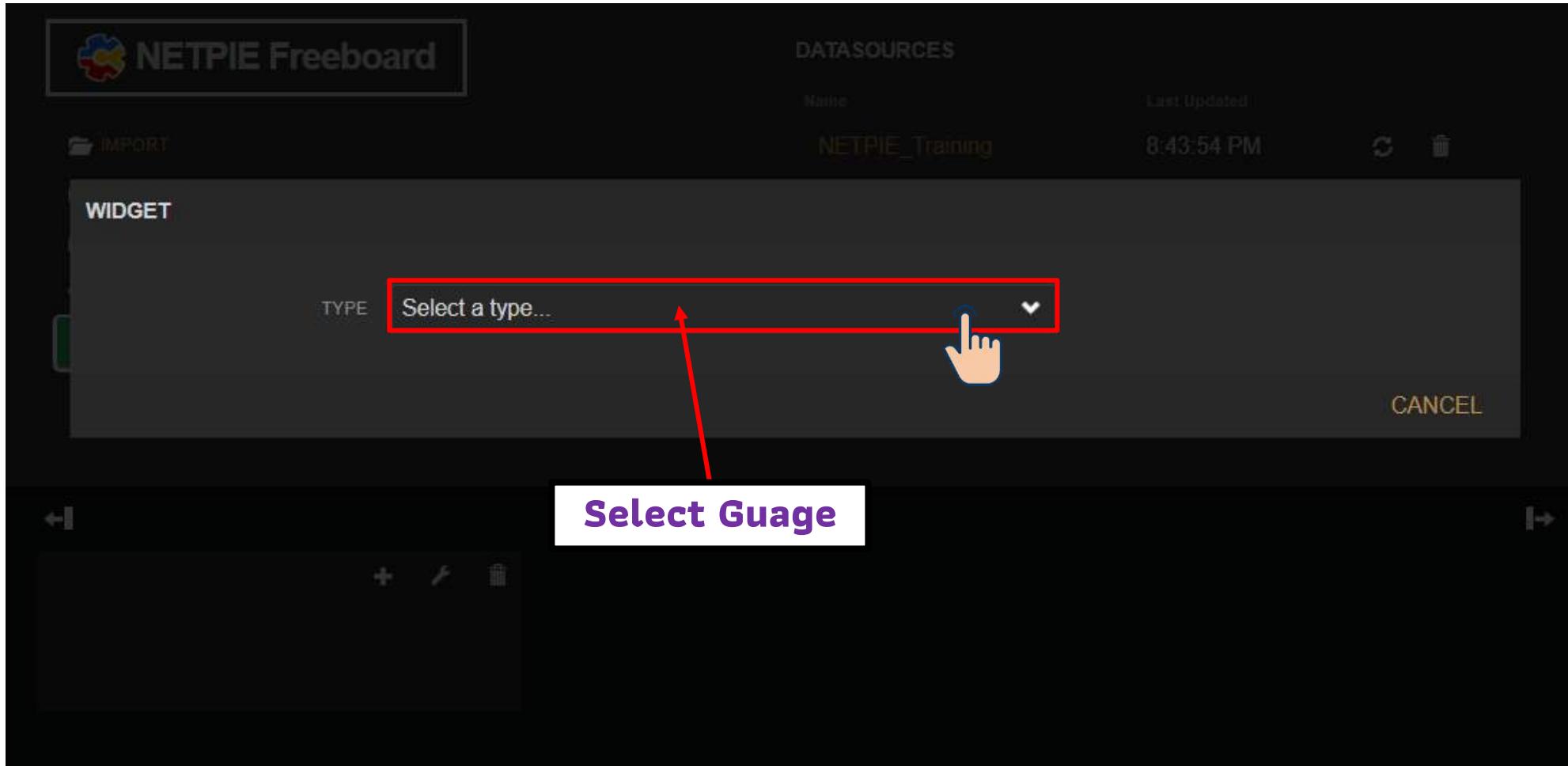
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



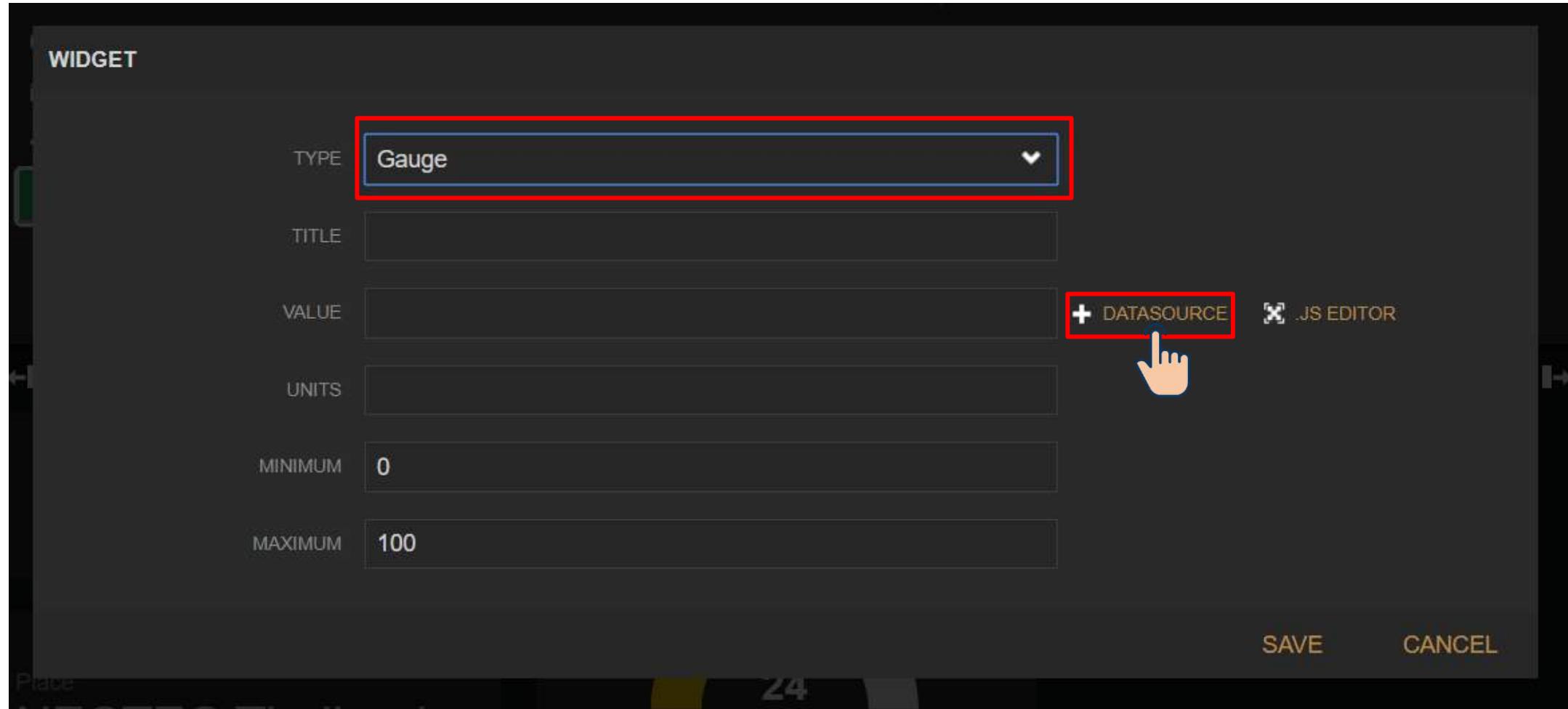
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



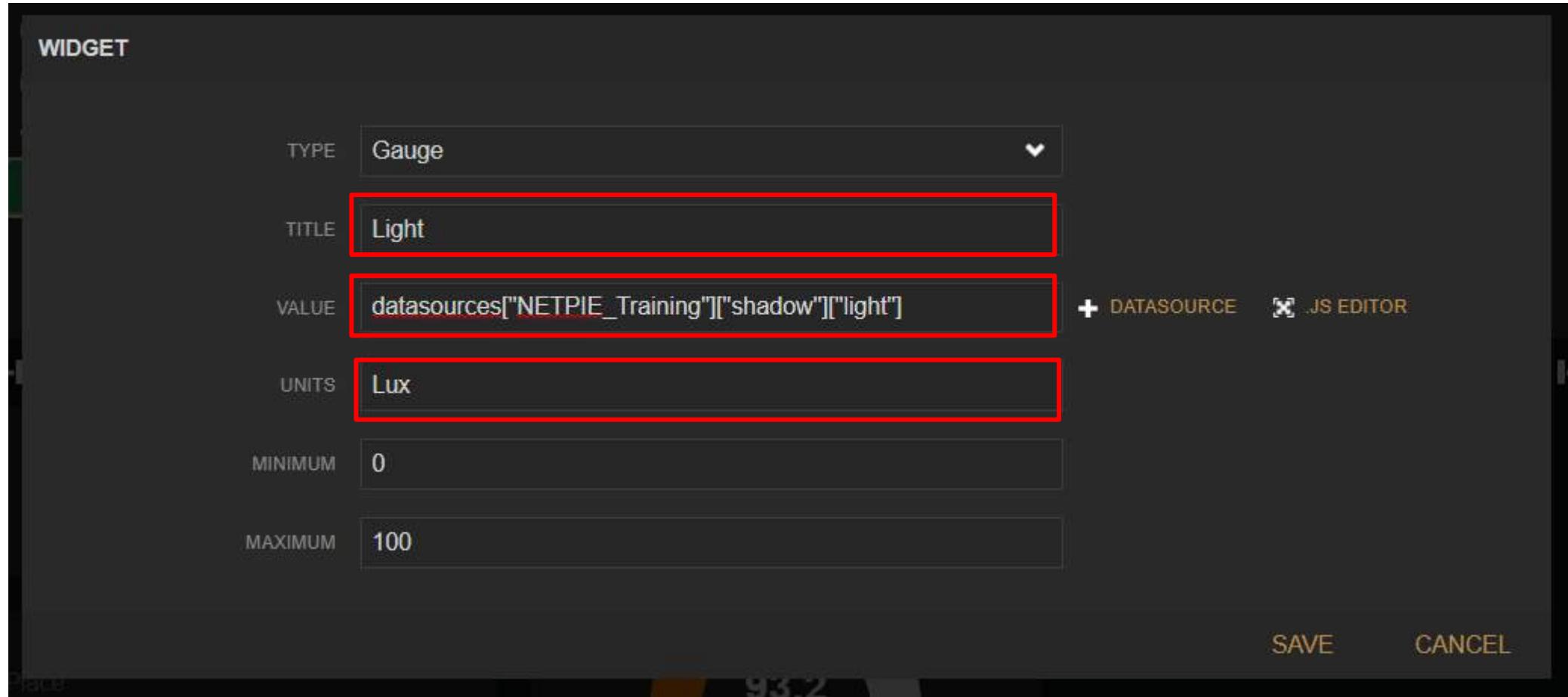
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



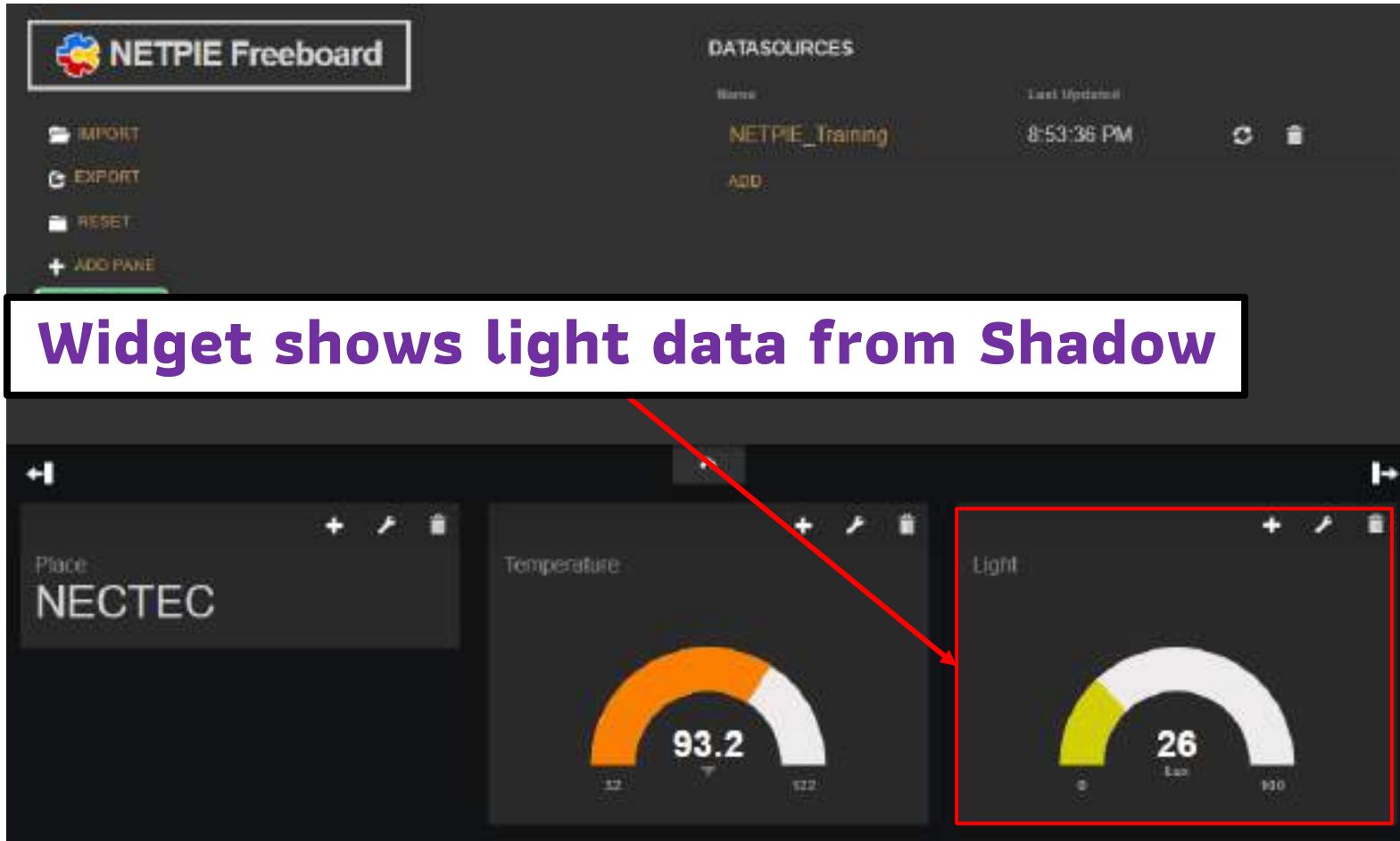
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



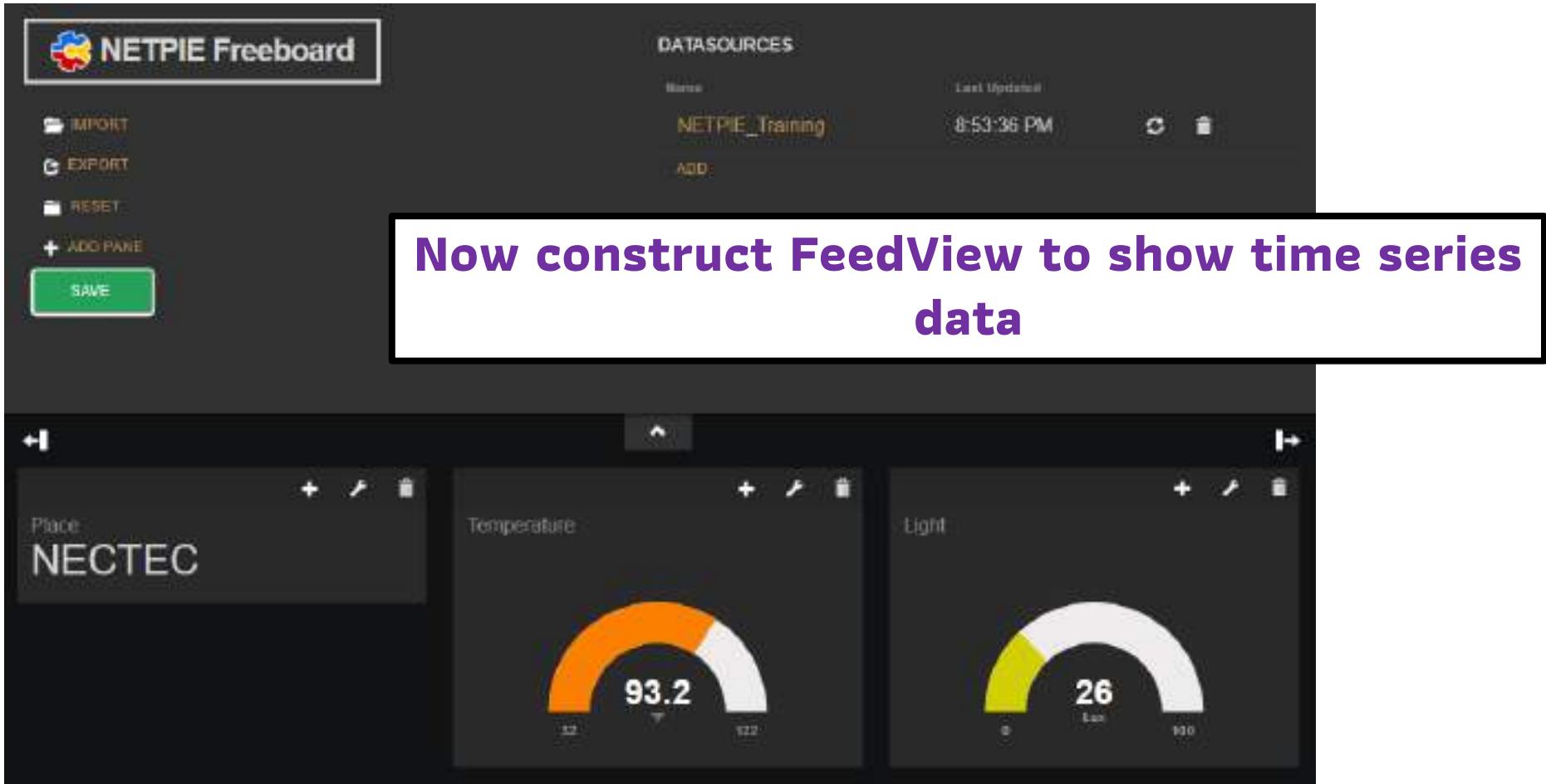
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



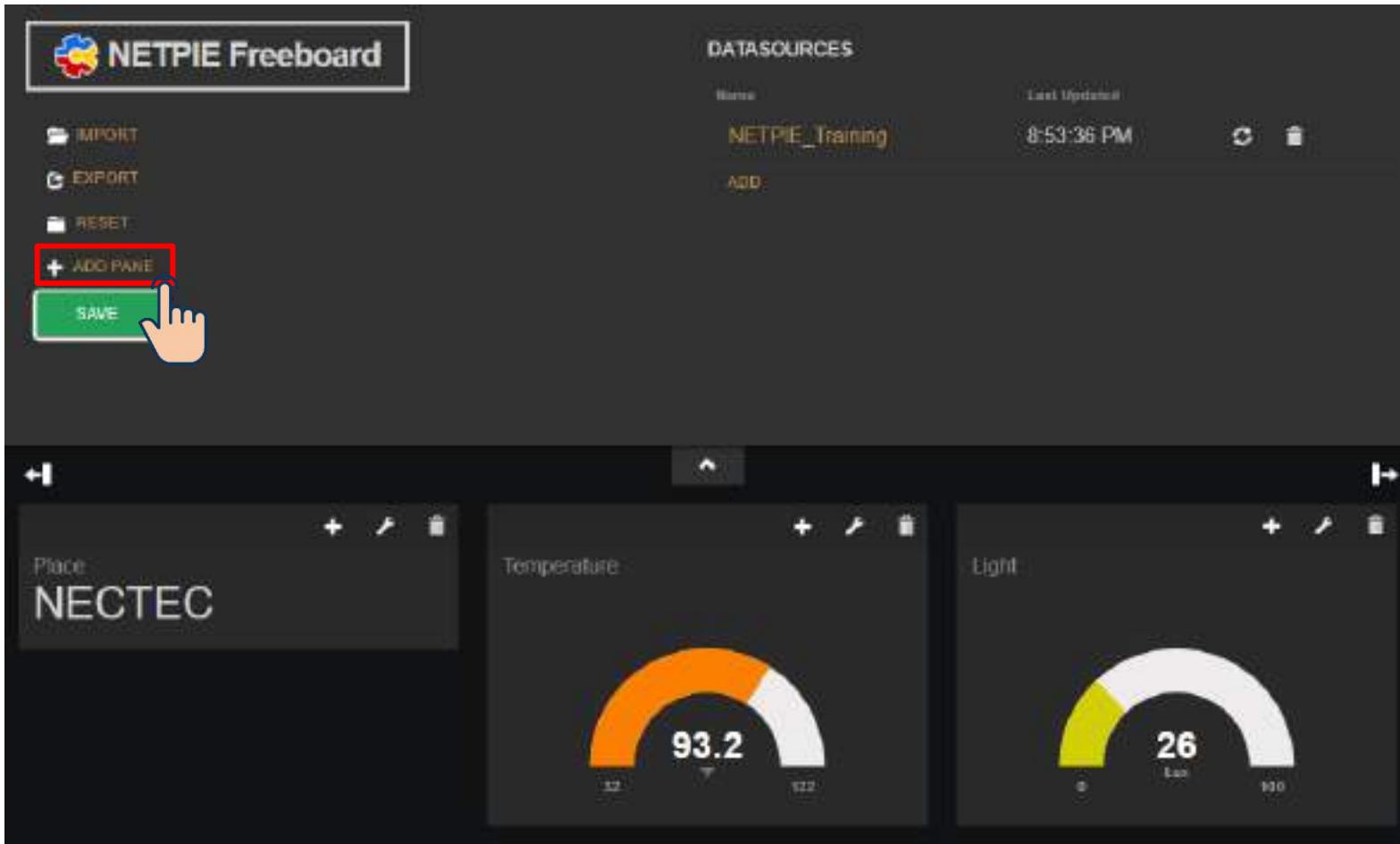
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



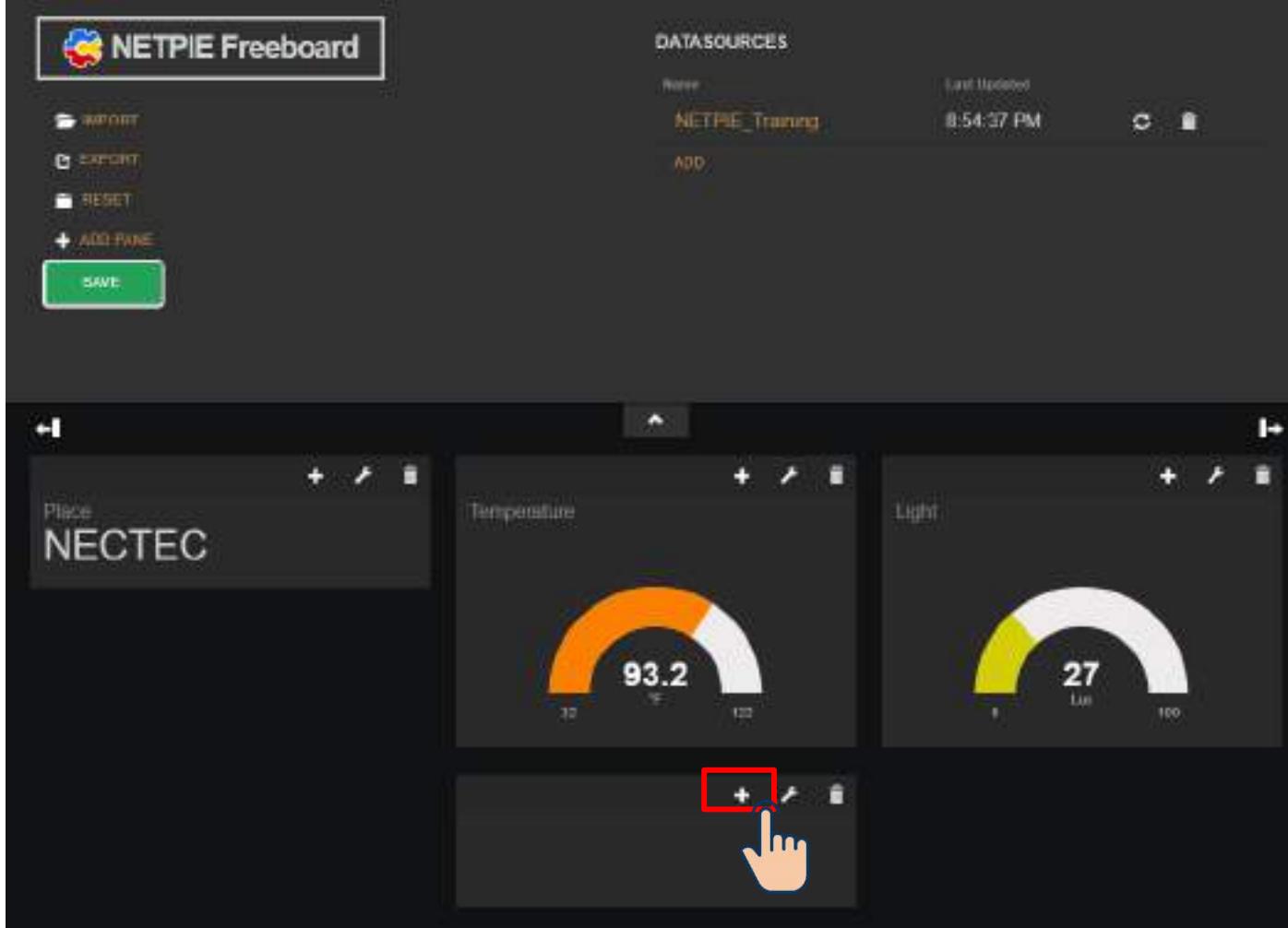
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



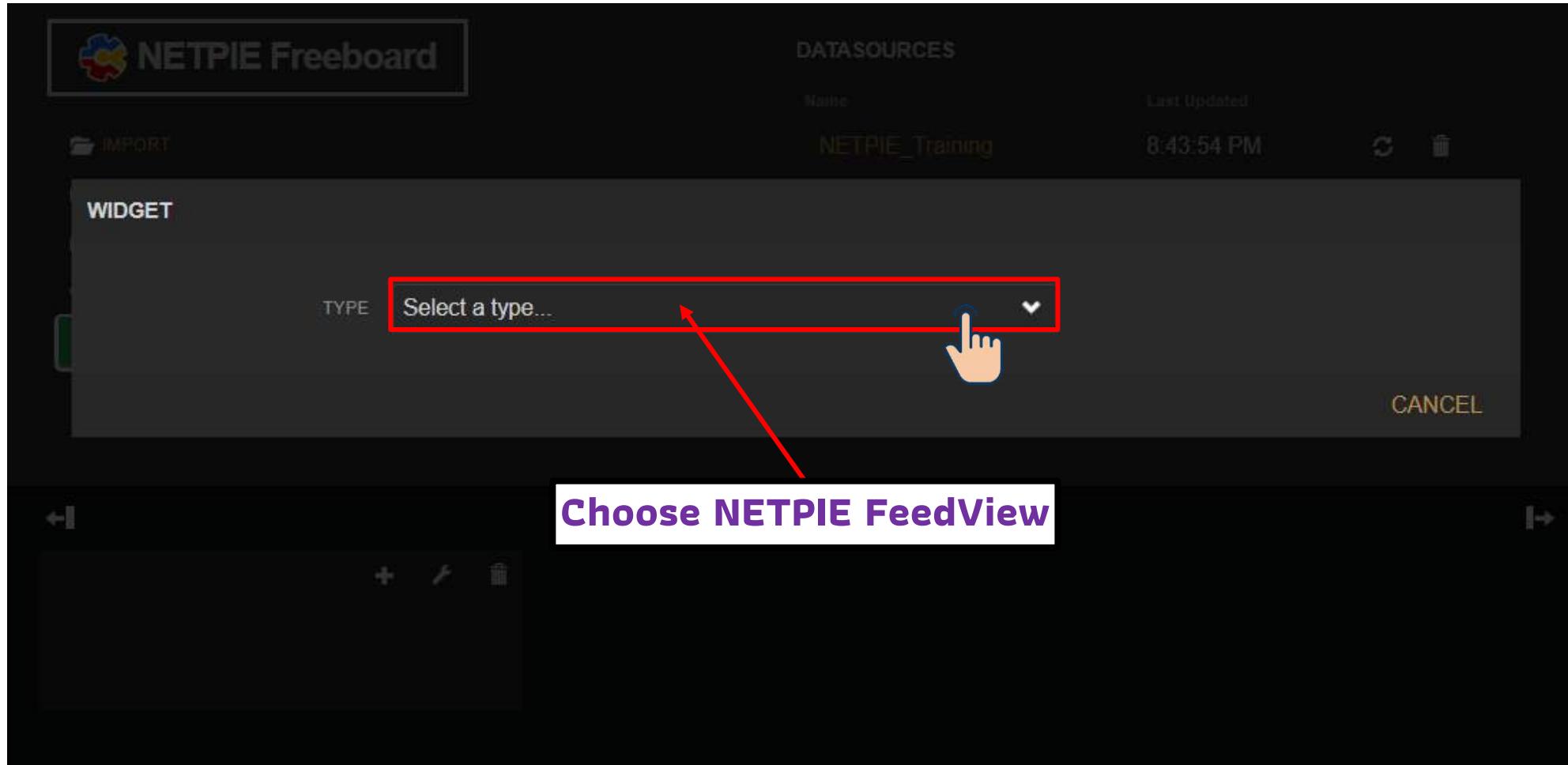
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



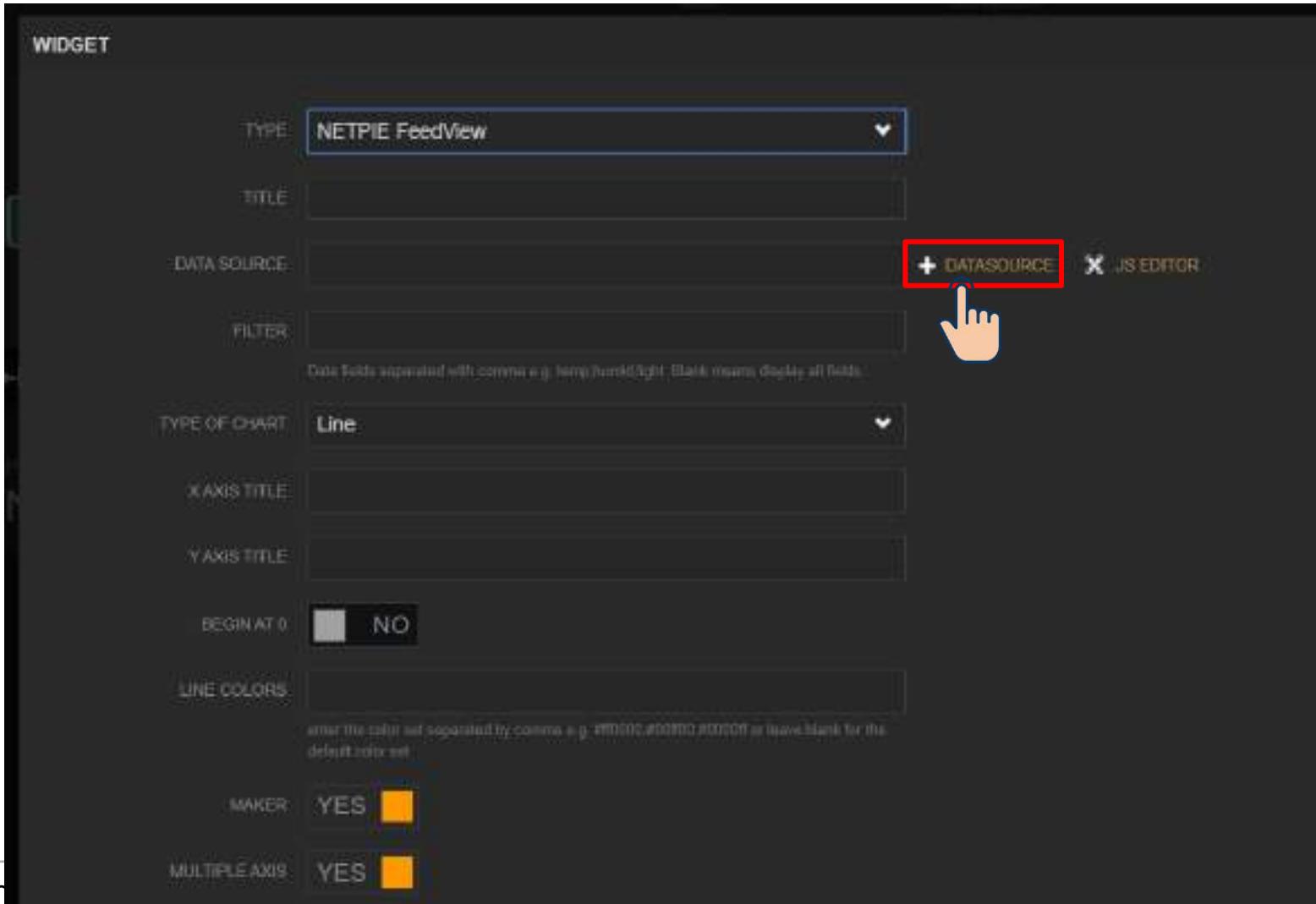
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



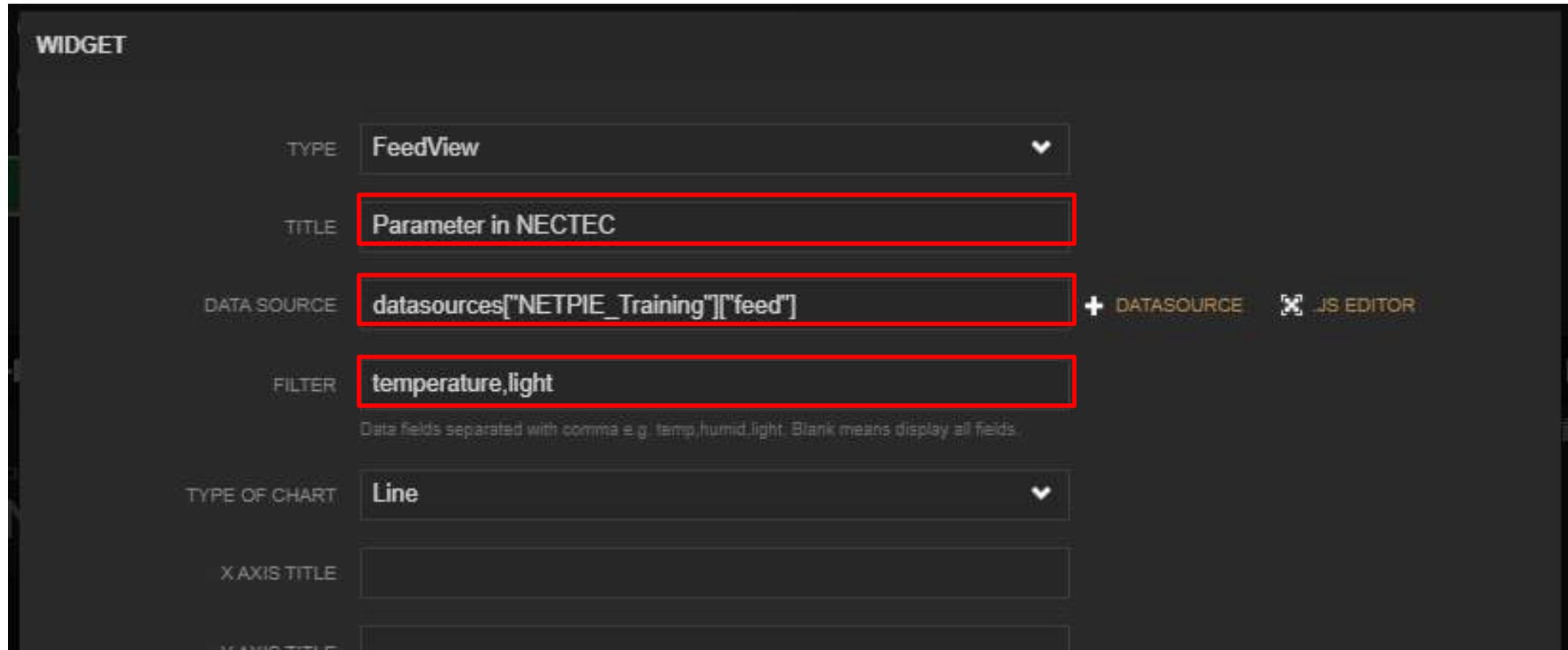
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



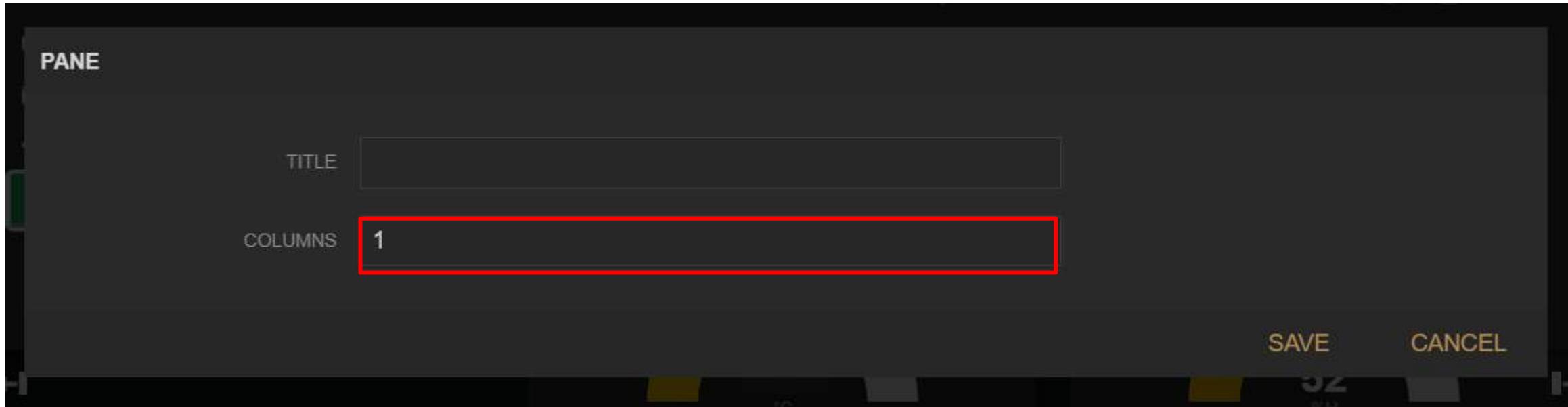
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



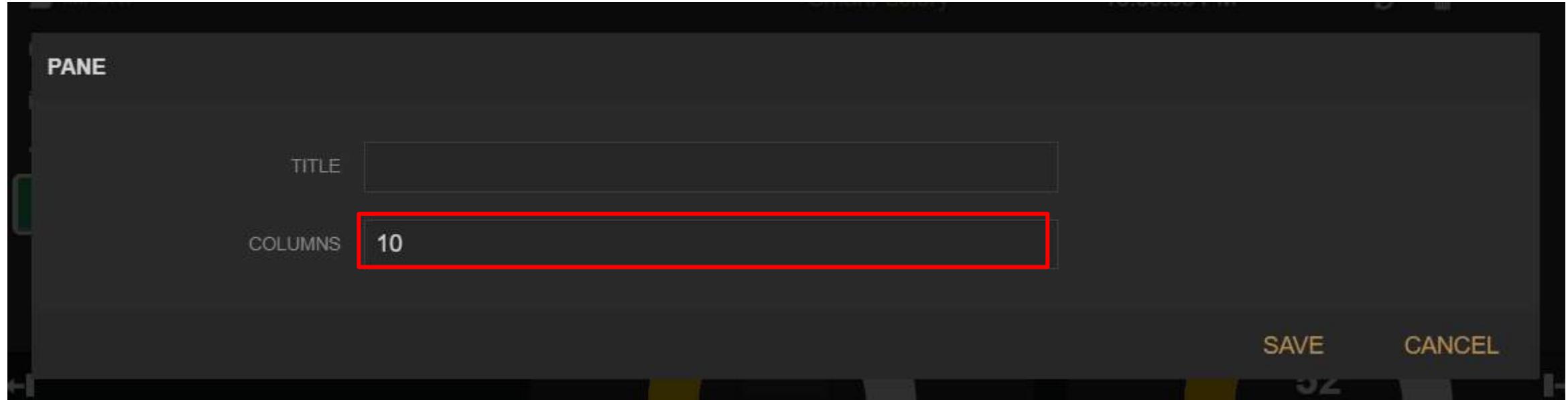
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



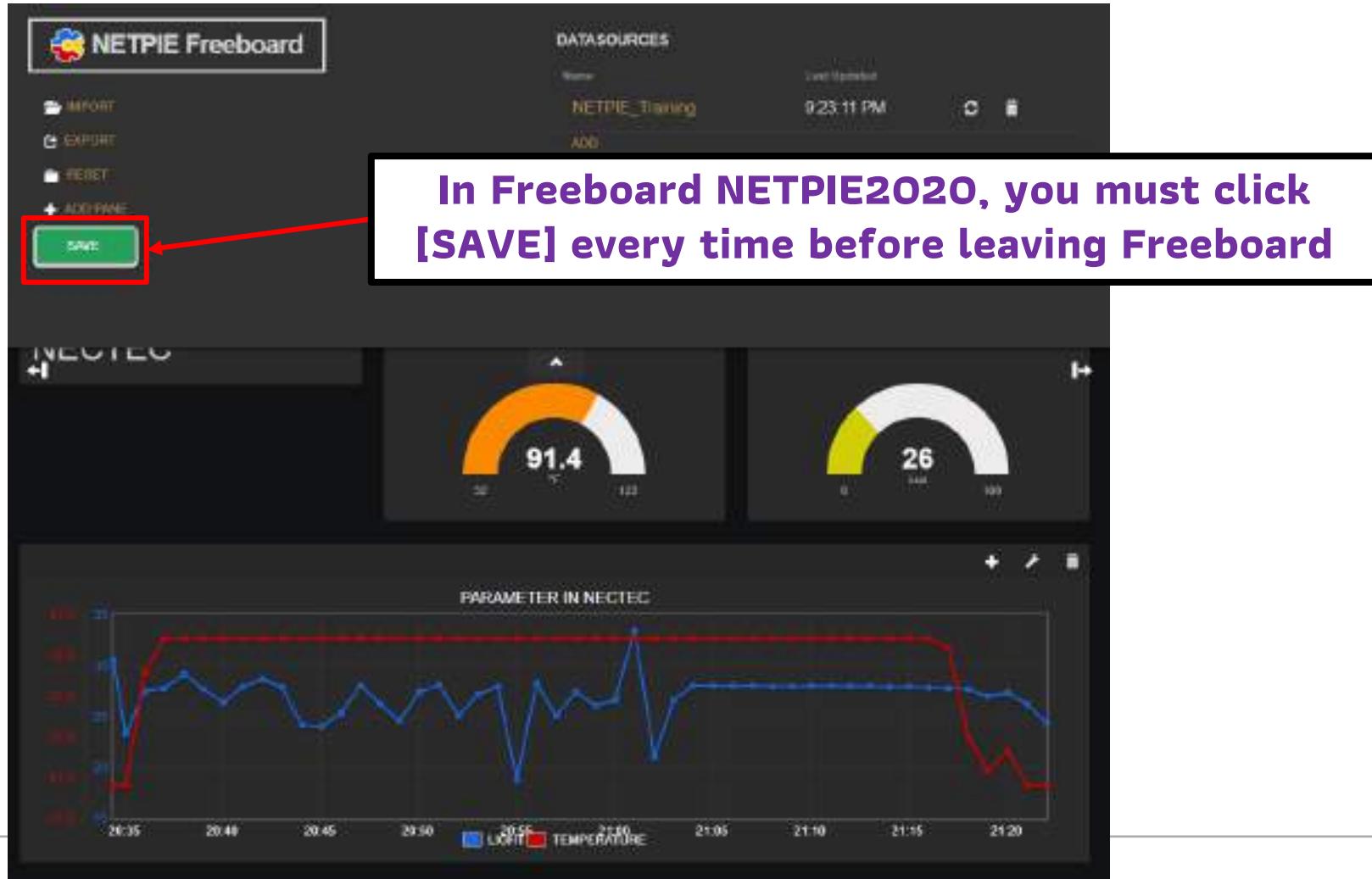
# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



# 5 - Freeboard in NETPIE2020

## Exercise 6 : Construct a Freeboard on NETPIE2020



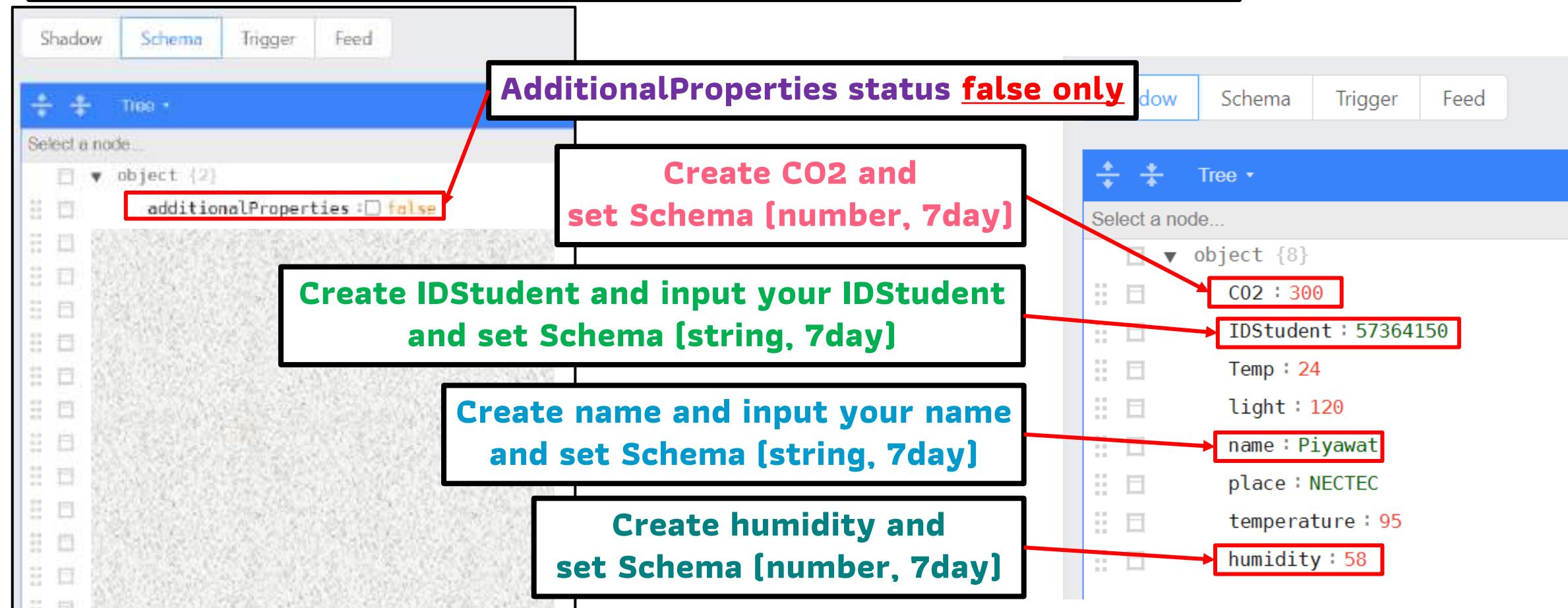
6

# Workshop



# 6 - Workshop

## Create Shadow & Freeboard follow by my Image



# 6 - Workshop

Create Shadow & Freeboard follow by my Image



# 6 - Workshop

---

## How to send workshop???

1

**Copy your device schema and paste on text file (.txt)**

2

**Capture your Freeboard Screen (.img or .png)**

3

**Drop 2 File in Google Drive**



Drop your workshop at  
Link --> <https://bit.ly/3m4isJj>

Set File Name is W follow by your IDStudent  
Such as W1\_57364150, W2\_57364150  
(W1 = Schema File, W2 = Freeboard File)





# Internet of Things & NETPIE2020 - Workshop

[For 11 Sep'21]

**Mr. Piyawat Jomsathan [Tae]**  
**Cyber-Physical Systems [CPS]**  
**National Electronics and Computer Technology Center [NECTEC], Thailand**

# Download Document & Software

Link --> <https://bit.ly/3BXJzeh>



# Main Topic

1

Data Management in NETPIE2020 (Trigger & Event Hook)

2

MQTT Library in Arduino IDE (Pubsubclient)

3

Freeboard in NETPIE2020 (Control Widget)

4

RESTful API

5

Workshop



1

# Data Management in NETPIE2020 (Trigger & Event Hook)



# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Device Trigger and Event Hook

It is a system that binds to changing Device Shadow information to external actions [Event Hook], such as setting alerts according to different status. According to the condition of the device set by the Trigger will be declared in JSON format:

```
{  
    "enabled": true,  
    "trigger": [  
        {  
            "action": "EVENT_HOOK_NAME",  
            "event": "SHADOW.UPDATED or DEVICE.STATUSCHANGED",  
            "condition": "Operation List ==, !=, >, >=, <, <=, in",  
            "msg": "text",  
            "option": {}  
        }  
    ]  
}
```

# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Device Trigger and Event Hook

Trigger Format consists of 2 parts

1. enable is used to turn the Trigger on/off

```
{  
  "enabled": true,  
  "trigger": [  
    {  
      "action": "EVENT_HOOK_NAME",  
      "event": "SHADOW.UPDATED or DEVICE.STATUSCHANGED",  
      "condition": "Operation List ==, !=, >, >=, <, <=, in",  
      "msg": "text",  
      "option": {}  
    }  
  ]  
}
```

# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Device Trigger and Event Hook

Trigger Format consist of 2 parts

2. trigger is for Trigger settings

```
{  
    "enabled": true,  
    "trigger": [  
        {  
            "action": "EVENT_HOOK_NAME",  
            "event": "SHADOW.UPDATED or DEVICE.STATUSCHANGED",  
            "condition": "Operation List ==, !=, >, >=, <, <=, in",  
            "msg": "text",  
            "option": {}  
        }  
    ]  
}
```

- action : what Trigger has to do when an event occurs. Specify Event Hook name.

- event : corresponds to two types of change in Device Shadow  
**SHADOW.UPDATED** : happens when Device Shadow Data changes according to certain condition (in this case we have to specify some condition, otherwise the Trigger would not fire)  
**DEVICE.STATUSCHANGED** : happens when the Device changes its platform connection status from Online to Offline, or vice versa.

# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Device Trigger and Event Hook

Trigger Format consist of 2 parts

```
{  
  "enabled": true,  
  "trigger": [  
    {  
      "action": "EVENT_HOOK_NAME",  
      "event": "SHADOW.UPDATED or DEVICE.STATUSCHANGED",  
      "condition": "Operation List ==, !=, >, >=, <, <=, in",  
      "msg": "text",  
      "option": {}  
    }  
  ]  
}
```

Condition : conditional change in Device Shadow Data . Use in case of SHADOW.UPDATED

msg : a message to notify the user when Trigger fires

option : use for specify some other parameters

For reference, variables in Trigger can be found in

<https://docs.nexpie.io/device-config.html#device-trigger-and-event-hook>

# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Device Trigger and Event Hook

### Trigger and Event Hook example

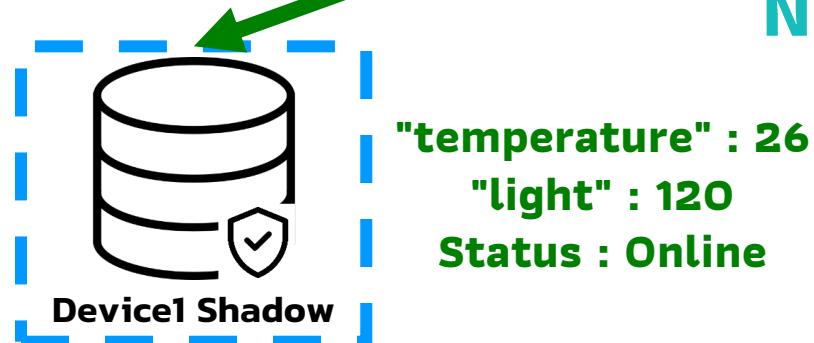
Event Hooks  
LINE NOTIFY



Publish : @shadow/data/update



NETPIE2020

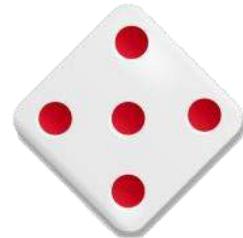


Trigger : DEVICE.STATUSCHANGE

# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Device Trigger and Event Hook

### Trigger and Event Hook example



MQTTBox1

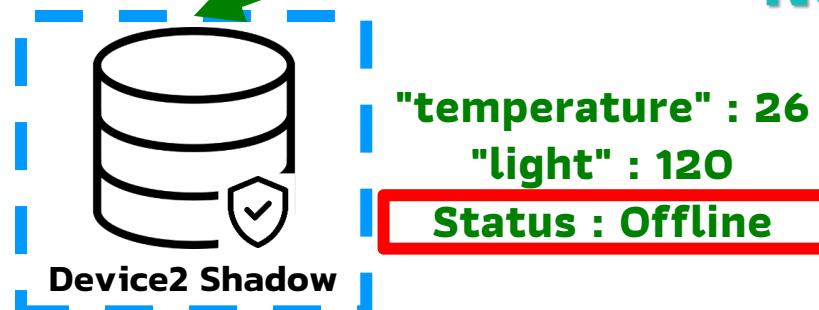
Publish : @shadow/data/update



Event Hooks  
LINE NOTIFY



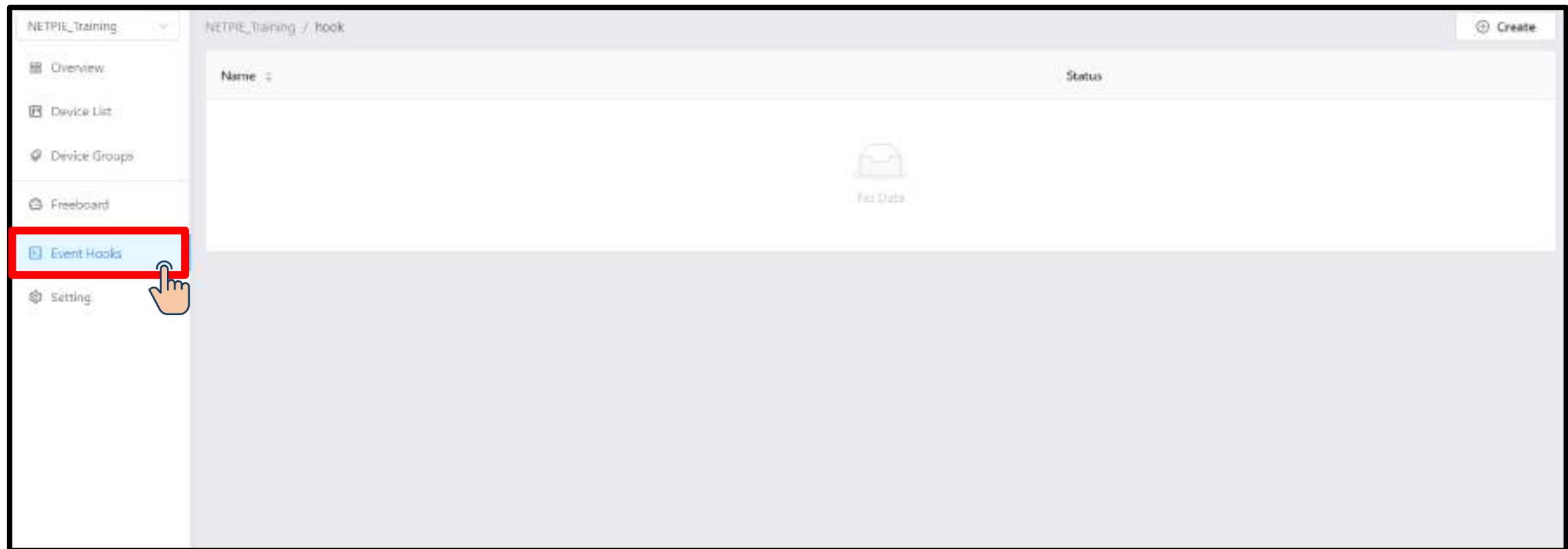
NETPIE2020



Trigger : DEVICE.STATUSCHANGE

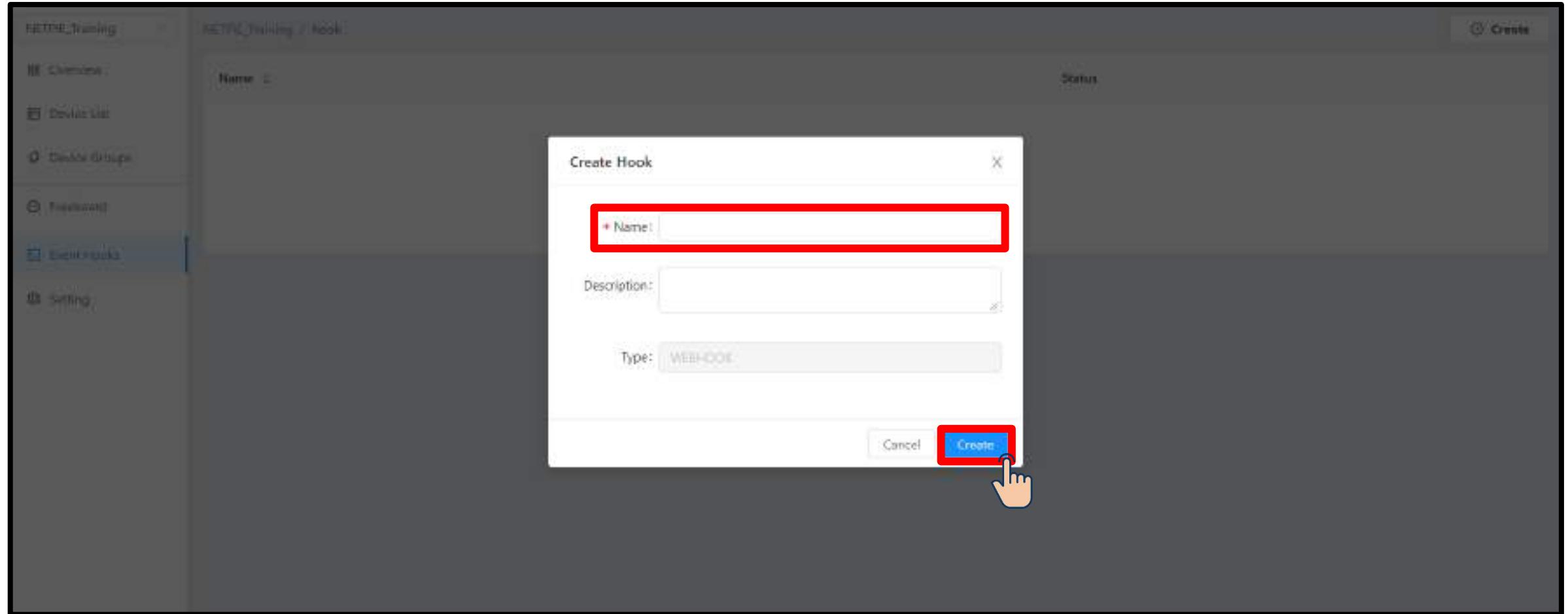
# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Create Event Hook



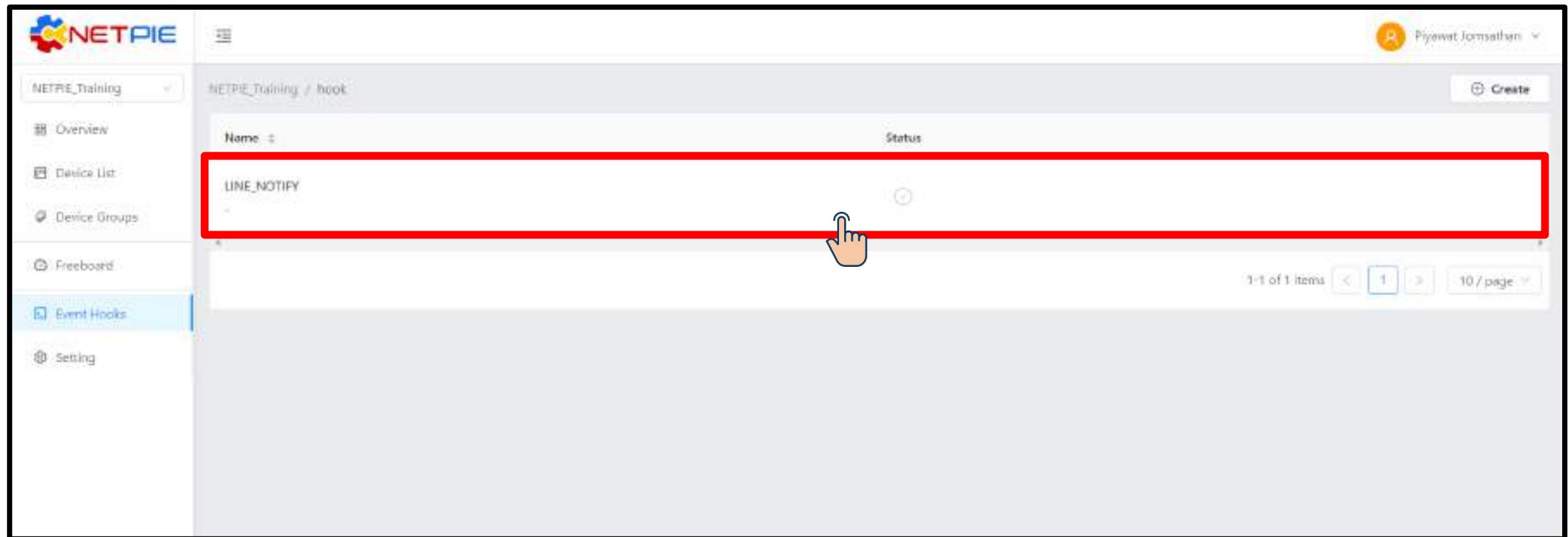
# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Create Event Hook



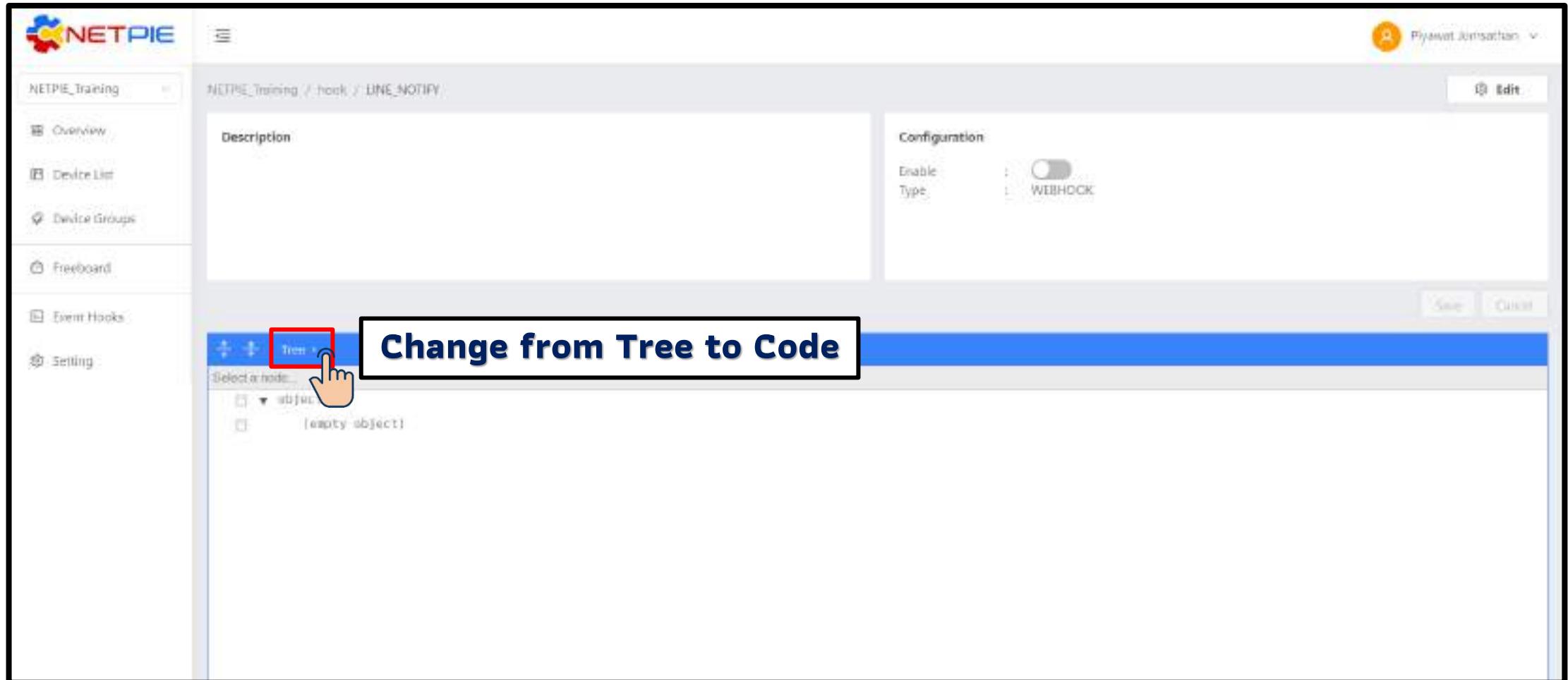
# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Create Event Hook



# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Create Event Hook



# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Create Event Hook

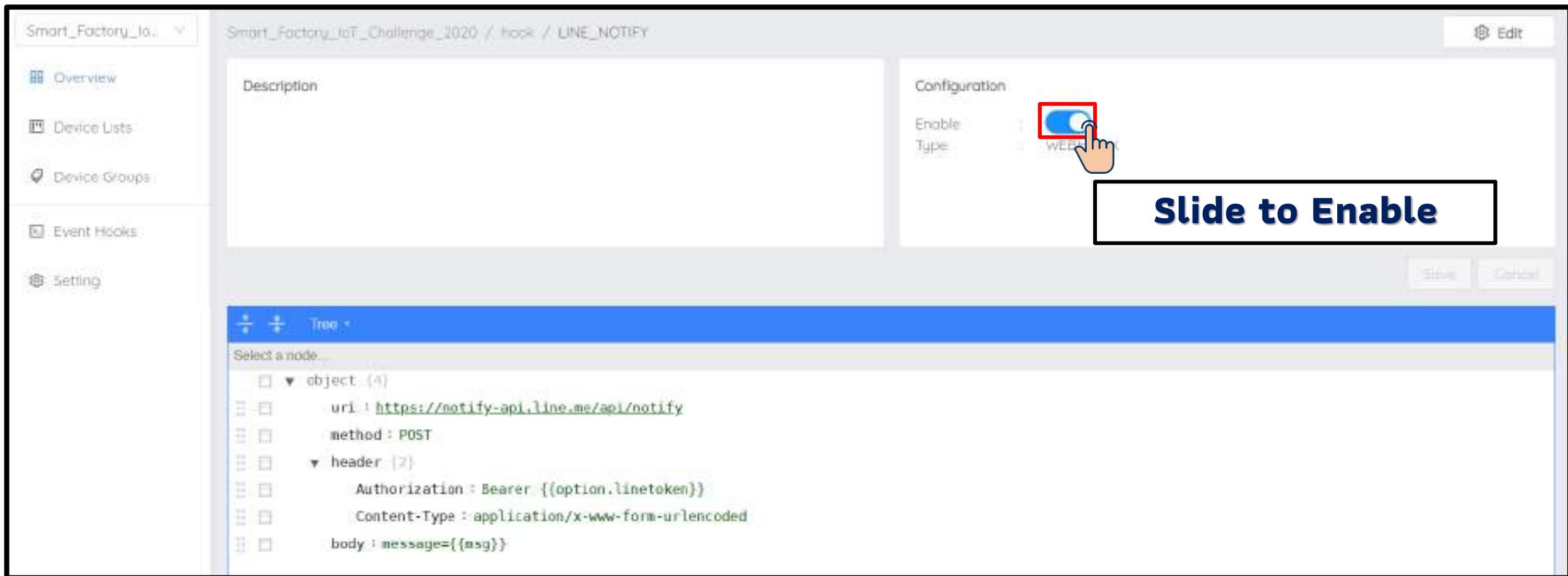
### Event Hook Example of a Line Notify that specifies 4 Attributes

```
{  
  "body": "message={{msg}}",  
  "header": {  
    "Authorization": "Bearer {{option.linetoken}}",  
    "Content-Type": "application/x-www-form-urlencoded"  
  },  
  "method": "POST",  
  "uri": "https://notify-api.line.me/api/notify"  
}
```

**Copy and paste to Event Hook on  
NETPIE2020**

# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

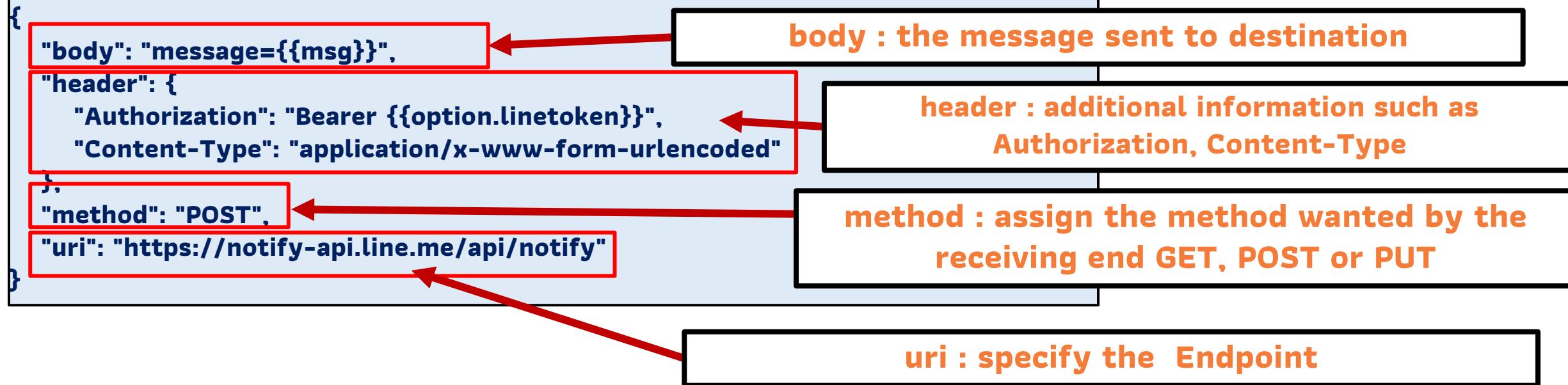
## Create Event Hook



# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Create Event Hook

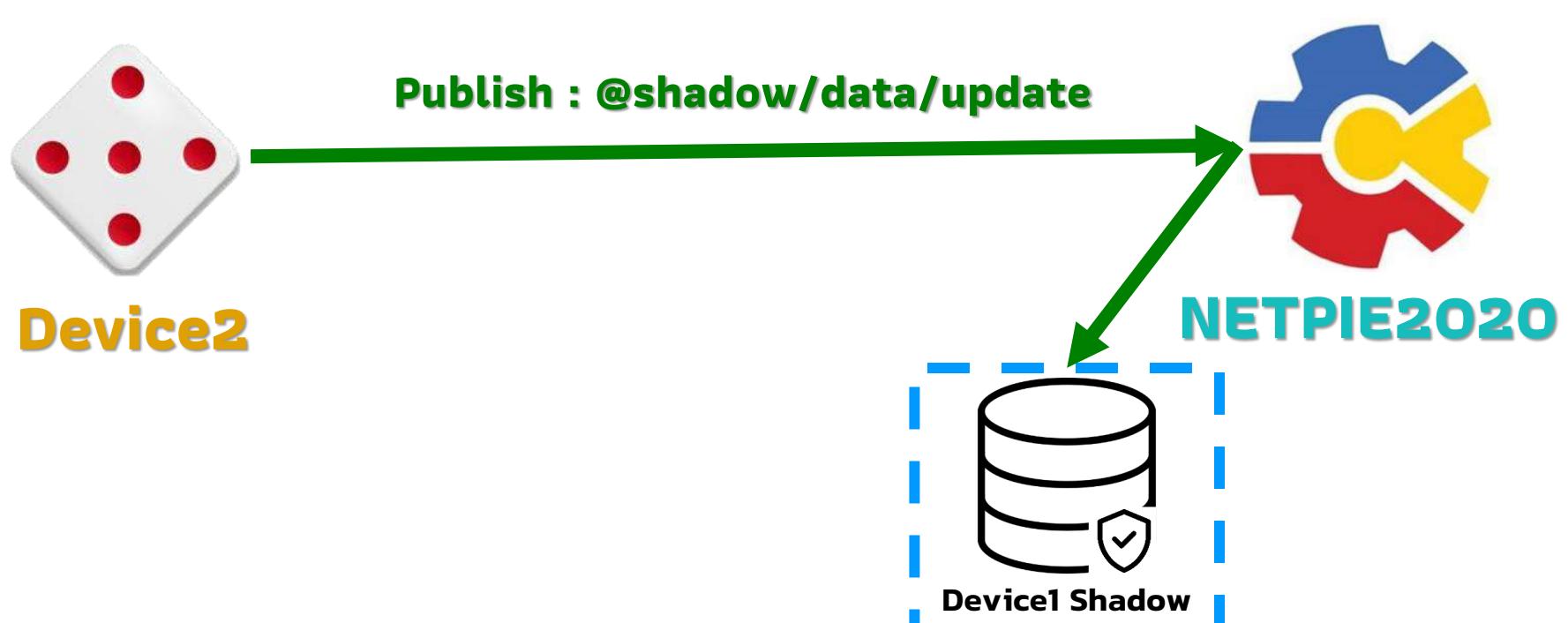
### Event Hook Example of a Line Notify that specifies 4 Attributes



\*\*\* In Event Hook, variables sent from Trigger can be referenced by using {{...}} symbols around them. For example, if we want to reference msg from Trigger, use {{msg}} Or in the option of linetoken, use {{option.linetoken}}

# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Exercise 1 : Create Device Trigger



**Event Hooks**  
**LINE NOTIFY**

LINE

**Trigger : SHADOW UPDATE & DEVICE.STATUSCHANGE**

# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

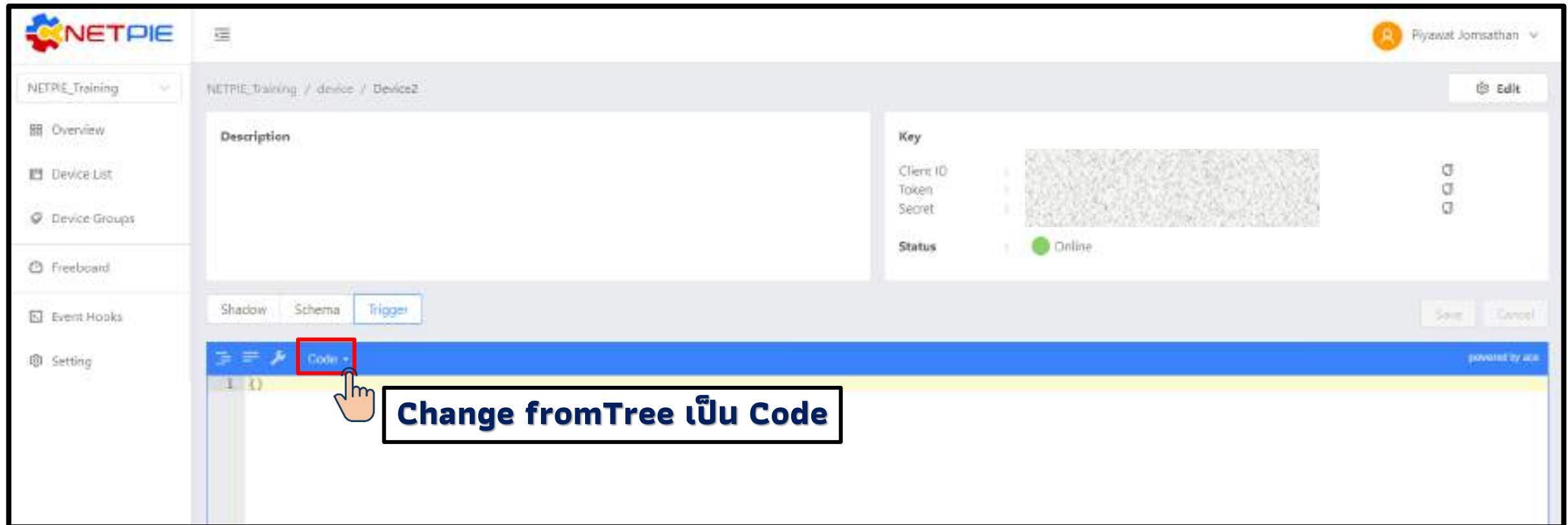
## Exercise 1 : Create Device Trigger

Select Device1 to create Trigger



# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Exercise 1 : Create Device Trigger



# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Exercise 1 : Create Device Trigger

```
{  
    "enabled": true,  
    "trigger": [  
        {  
            "action": "LINE_NOTIFY",  
            "event": "SHADOW.UPDATED",  
            "condition": "$.temperature!= $$.temperature",  
            "msg": "My temperature 2 was change from {{ $$.temperature }} to {{ $.temperature }}",  
            "option": {  
                "linetoken": "Your_Token_LINE"  
            }  
        },  
        {  
            "action": "LINE_NOTIFY",  
            "event": "DEVICE.STATUSCHANGED",  
            "msg": "My Device {{ $.statustext }}, statuscode: {{ $.status }}",  
            "option": {  
                "linetoken": "Your_Token_LINE"  
            }  
        }  
    ]  
}
```

# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Exercise 1 : Create Device Trigger

The screenshot shows the NETPIE2020 interface with the 'Trigger' tab selected. The main area displays a JSON configuration for two triggers:

```
1-1
2- "enabled": true,
3- "trigger": [
4-   {
5-     "action": "LINE_NOTIFY",
6-     "event": "SHADOW.UPDATED",
7-     "condition": "{$.temperature}<{$.temperature}",
8-     "msg": "My temperature 2 was change from {{$.temperature}} to {{$.temperature}}",
9-     "option": [
10-       "linetoken": "Your_Token_Line"
11-     ]
12-   },
13-   {
14-     "action": "LINE_NOTIFY",
15-     "event": "DEVICE.STATUSCHANGED",
16-     "msg": "My Device {{$.statusText}}, statuscode: {{$.status}}",
17-     "option": [
18-       "linetoken": "Your_Token_Line"
19-     ]
20-   }
21- ]
22- }
```

The interface includes a sidebar with navigation links: Overview, Device List, Device Groups, Fileboard, Event Hooks, and Setting. The 'Event Hooks' link is currently selected. On the right side, there is a 'Key' section with Client ID, Token, and Secret fields, and a 'Status' section indicating the device is Online.

# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Exercise 1 : Create Device Trigger

```
{  
  "enabled": true, Turn on Trigger  
  "trigger": [  
    {  
      "action": "LINE_NOTIFY",  
      "event": "SHADOW.UPDATED",  
      "condition": "$.temperature!= $$.temperature",  
      "msg": "My temperature 2 was change from {{ $$.temperature }} to {{ $.temperature }}.",  
      "option": {  
        "linetoken": "Your_Token_LINE"  
      }  
    },  
    {  
      "action": "LINE_NOTIFY",  
      "event": "DEVICE.STATUSCHANGED",  
      "msg": "My Device {{$statustext}}, statuscode: {{$status}}.",  
      "option": {  
        "linetoken": "Your_Token_LINE"  
      }  
    }  
  ]  
}
```

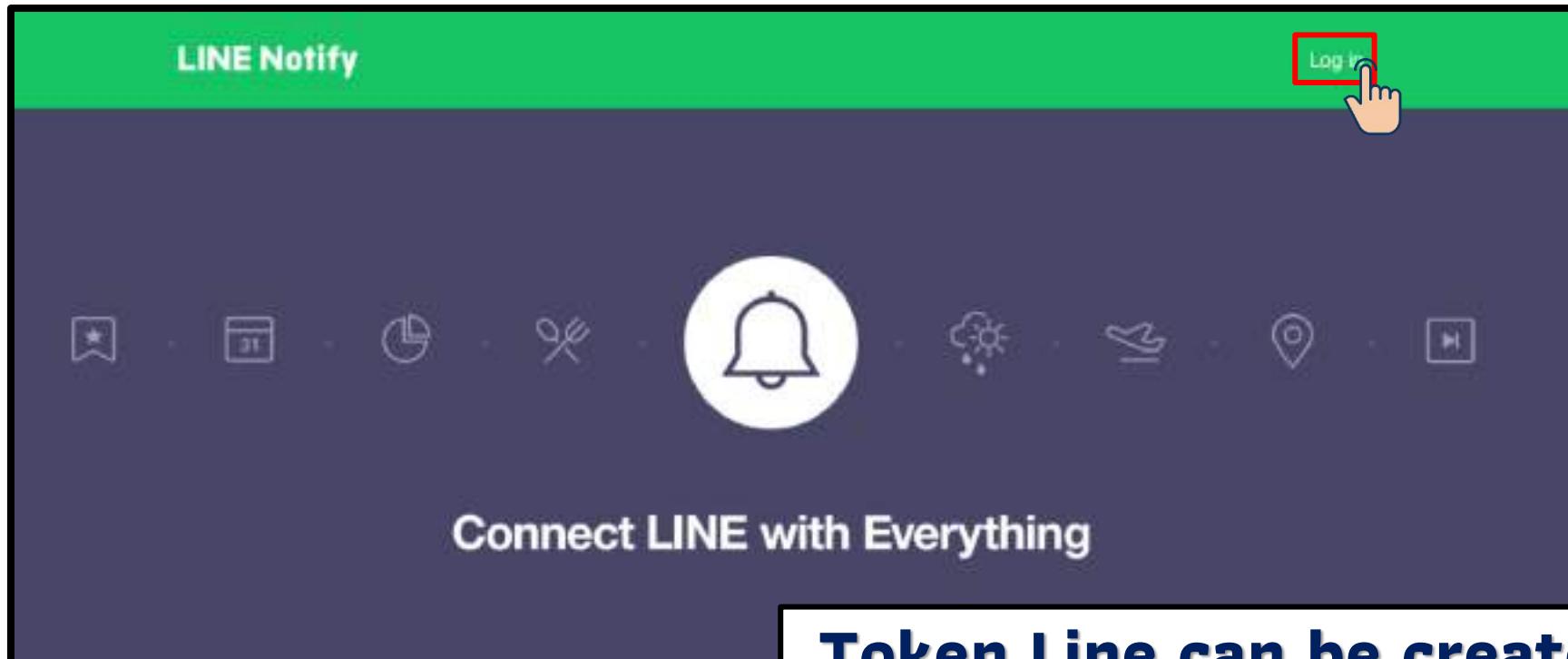
**Copy this trigger code and paste to Device Trigger**

**Trigger Shadow Update : the condition is to take action whenever the temperature changes, with action : “LINE\_NOTIFY” and the message is “My temperature 2 was changed from xx to xx.”**

**Trigger Status Change : The condition is to take action whenever device status changes, with action: “LINE\_NOTIFY” and the message is “My device xx statucode: xx”**

# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

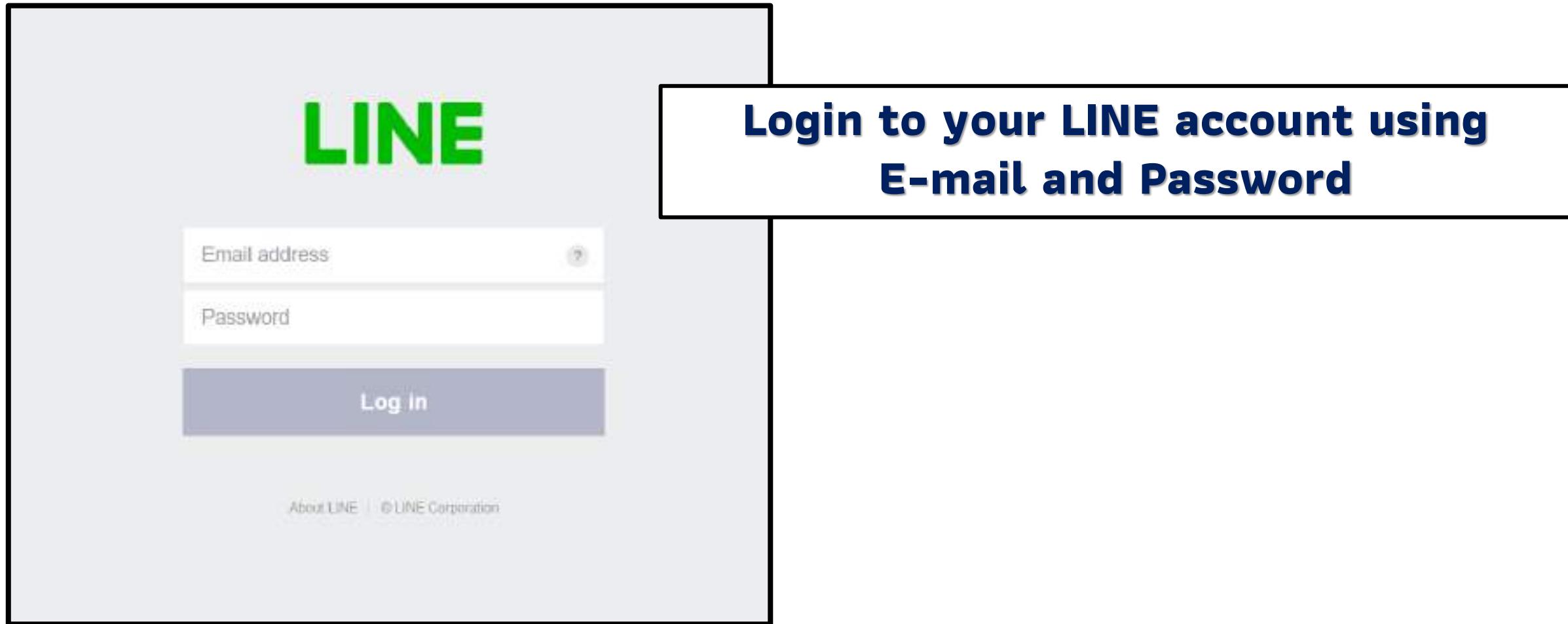
## Exercise 1 : Create Device Trigger



Receive web service notifications on LINE

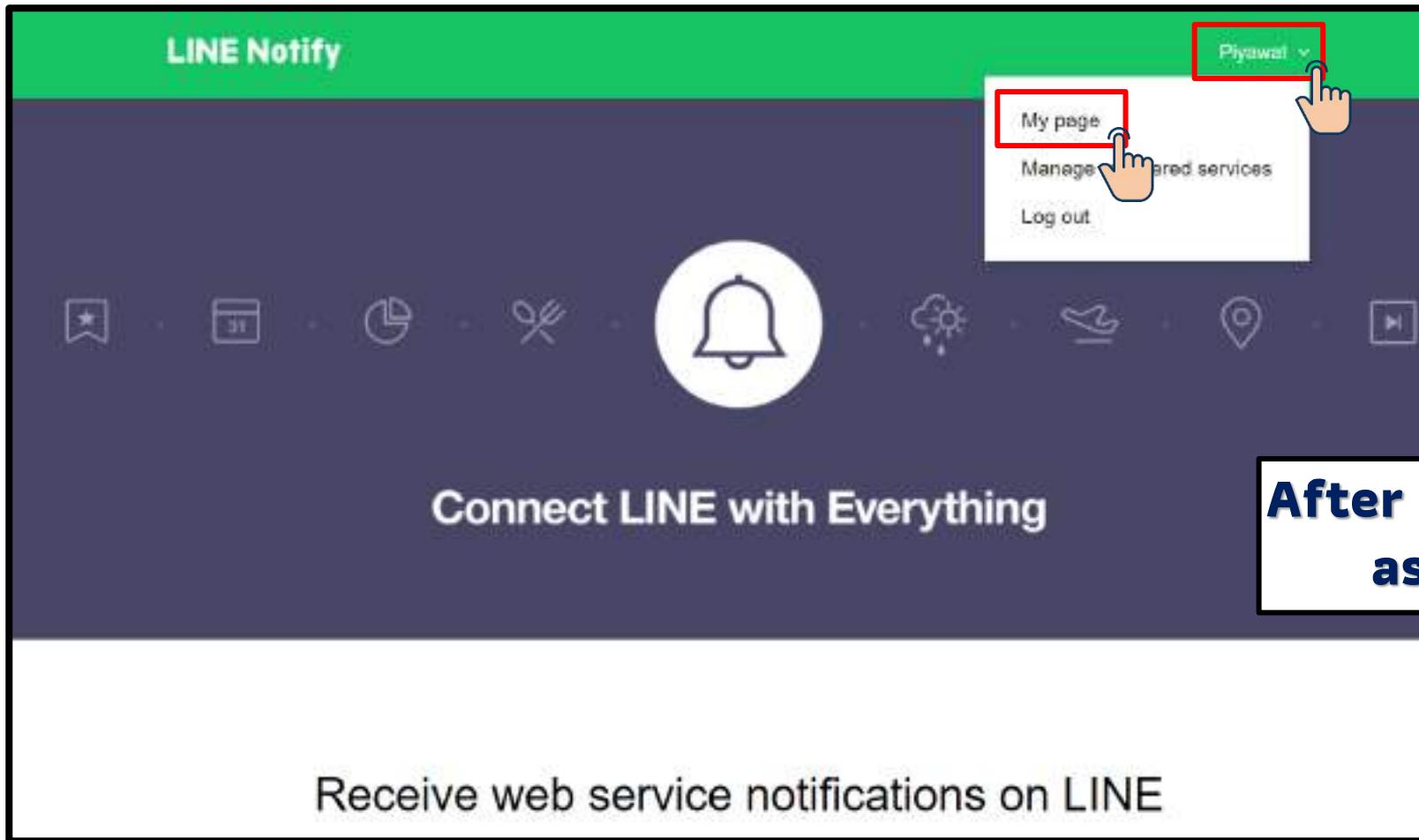
# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Exercise 1 : Create Device Trigger



# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Exercise 1 : Create Device Trigger



**After login, select >> My page  
as shown in the figure**

# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Exercise 1 : Create Device Trigger

**Scroll down and click on  
[Generate Token]**

Generate access token (For developers)

By using personal access tokens, you can configure notifications without having to add a web service.

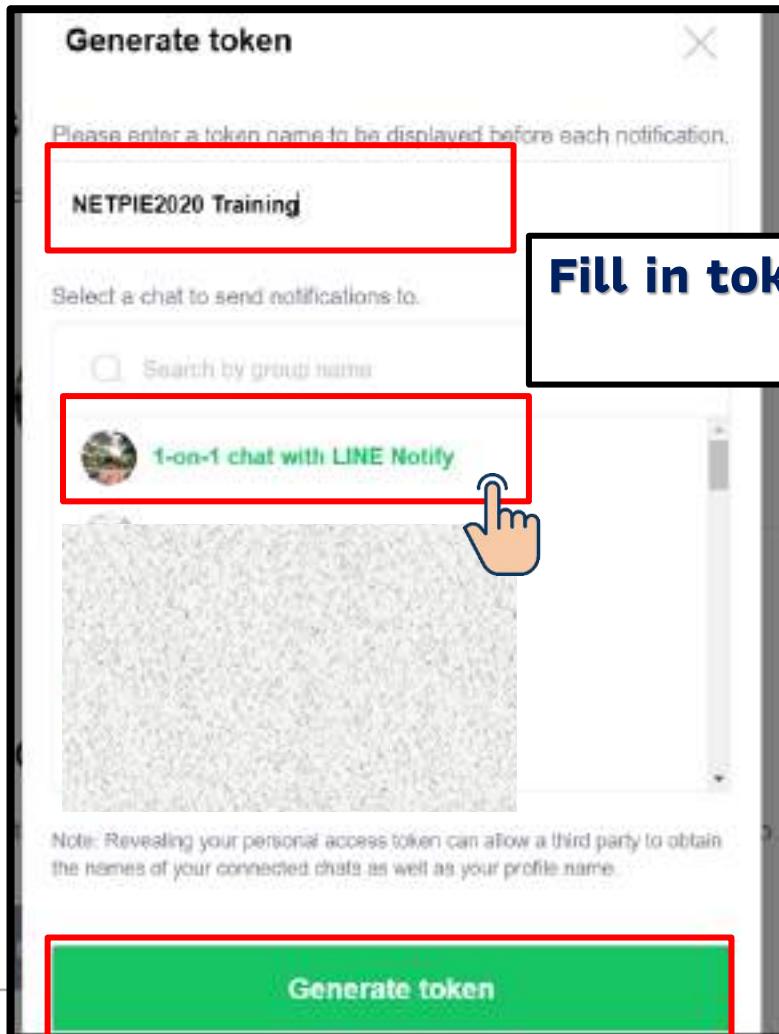
Generate token

LINE Notify API Document



# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

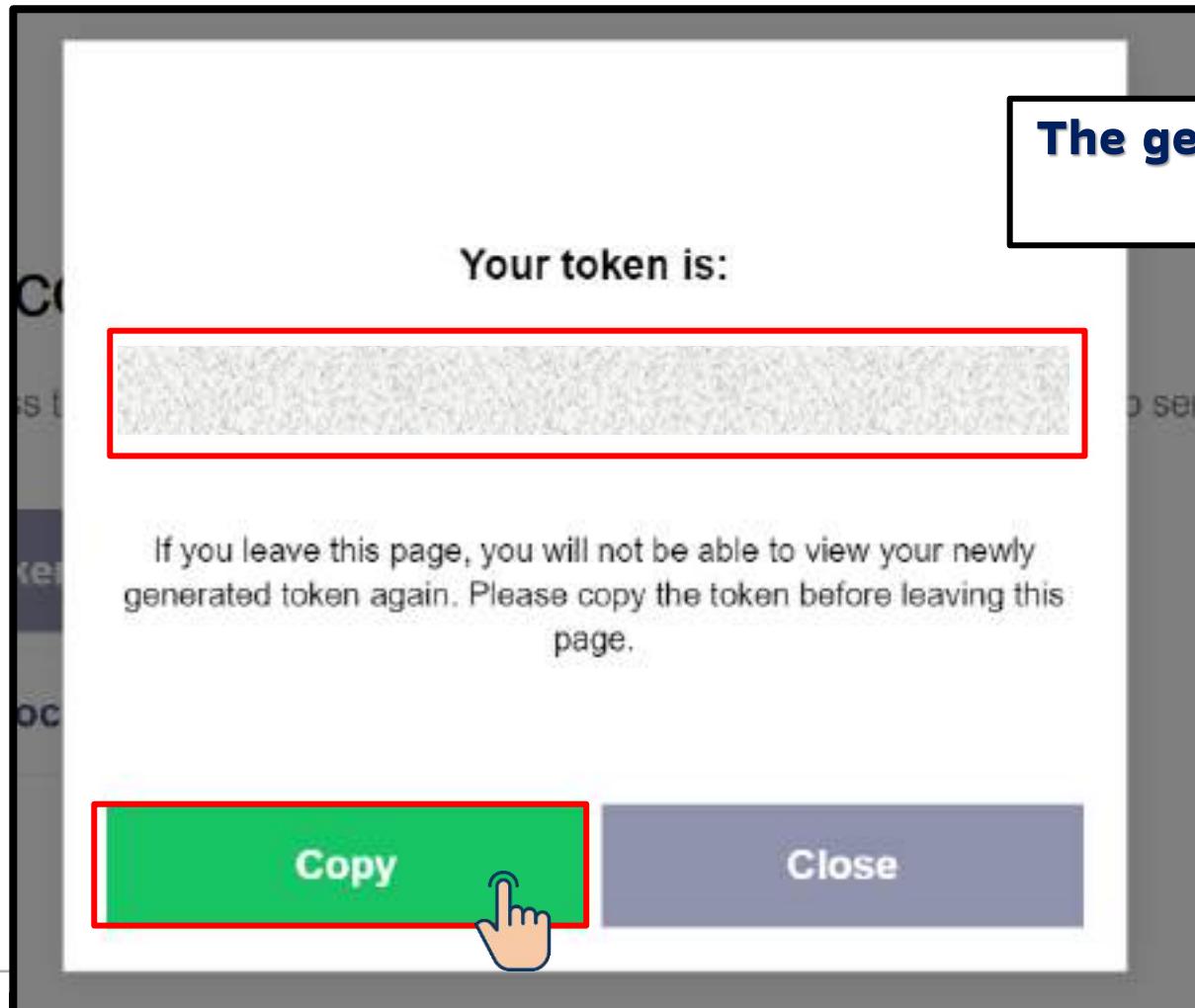
## Exercise 1 : Create Device Trigger



**Fill in token name, and select 1-on-1 chat with LINE notify.  
Click [Generate Token]**

# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

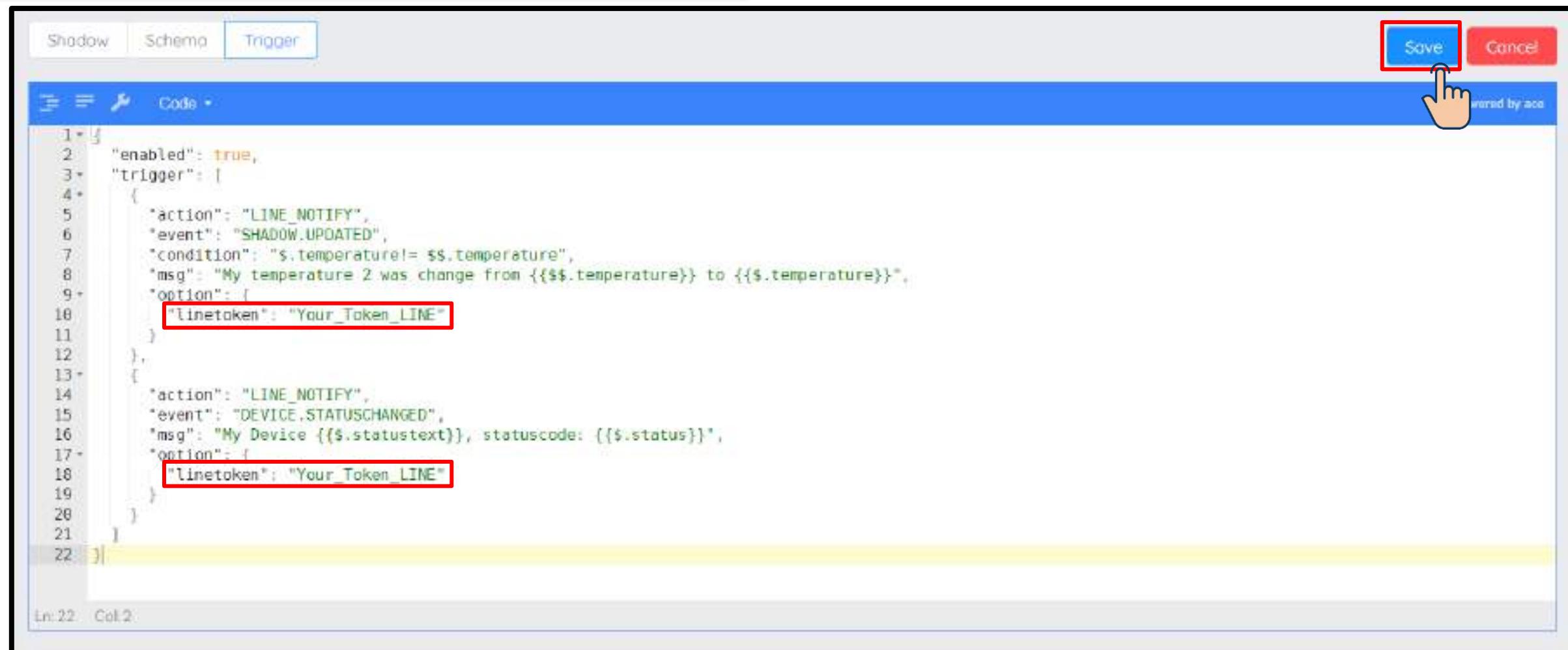
## Exercise 1 : Create Device Trigger



**The generated token is displayed. Copy and paste to linetoken in Device Trigger**

# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Exercise 1 : Create Device Trigger

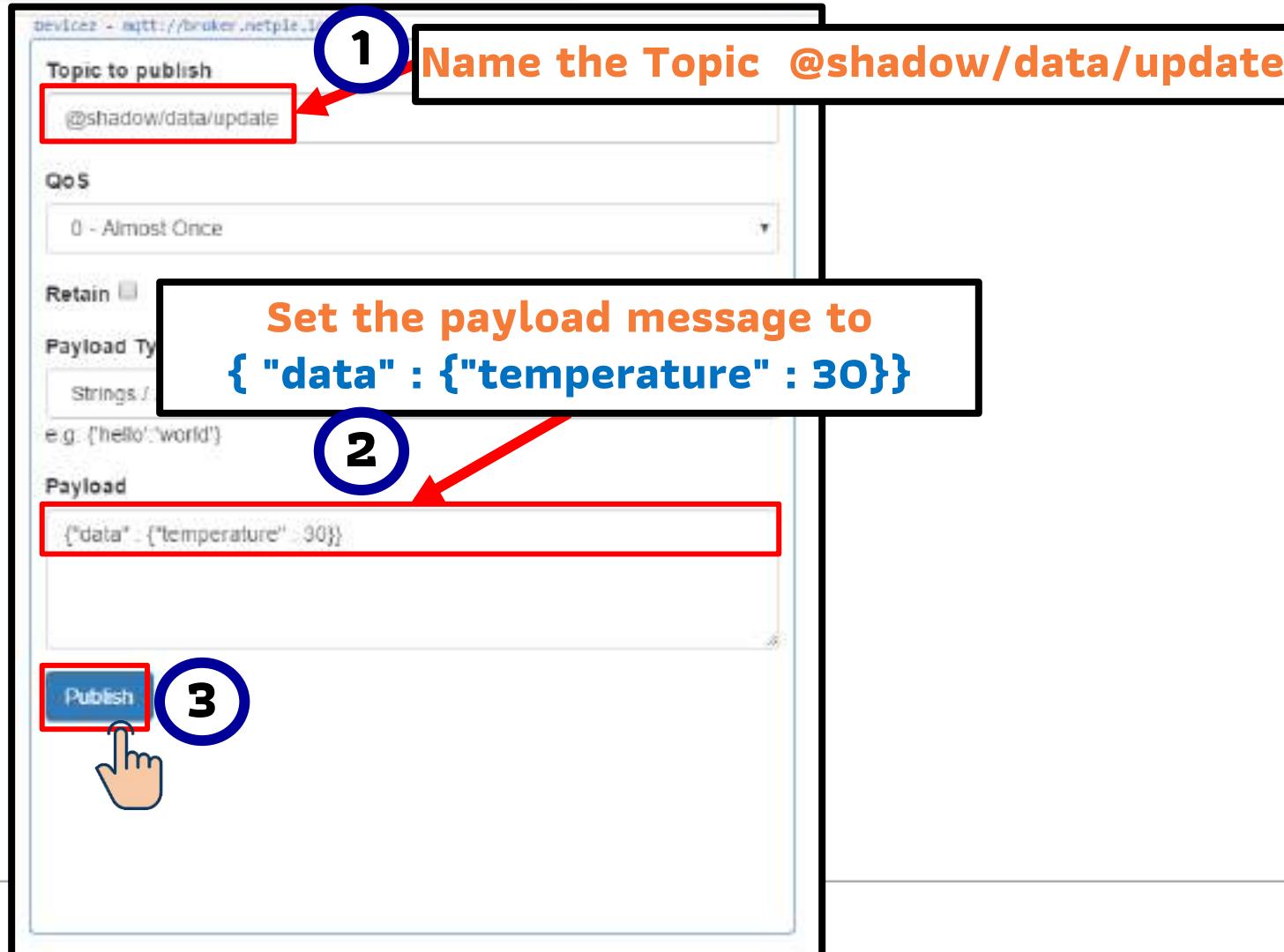


The screenshot shows the NETPIE2020 configuration interface with the 'Trigger' tab selected. The main area displays a JSON configuration file for device triggers. Two specific lines of code are highlighted with red boxes: 'linetoken': 'Your\_Token\_LINE' in both trigger definitions. A hand cursor is hovering over the 'Save' button, which is also highlighted with a red box.

```
1 "enabled": true,
2 "trigger": [
3     {
4         "action": "LINE_NOTIFY",
5         "event": "SHADOW.UPDATED",
6         "condition": "$.temperature!= $$.temperature",
7         "msg": "My temperature 2 was change from {{$$.temperature}} to {{$temperature}}",
8         "option": [
9             {
10                 "linetoken": "Your_Token_LINE"
11             }
12         ],
13     },
14     {
15         "action": "LINE_NOTIFY",
16         "event": "DEVICE.STATUSCHANGED",
17         "msg": "My Device {{$statustext}}, statuscode: {{$status}}",
18         "option": [
19             {
20                 "linetoken": "Your_Token_LINE"
21             }
22         ]
23     }
24 ]
```

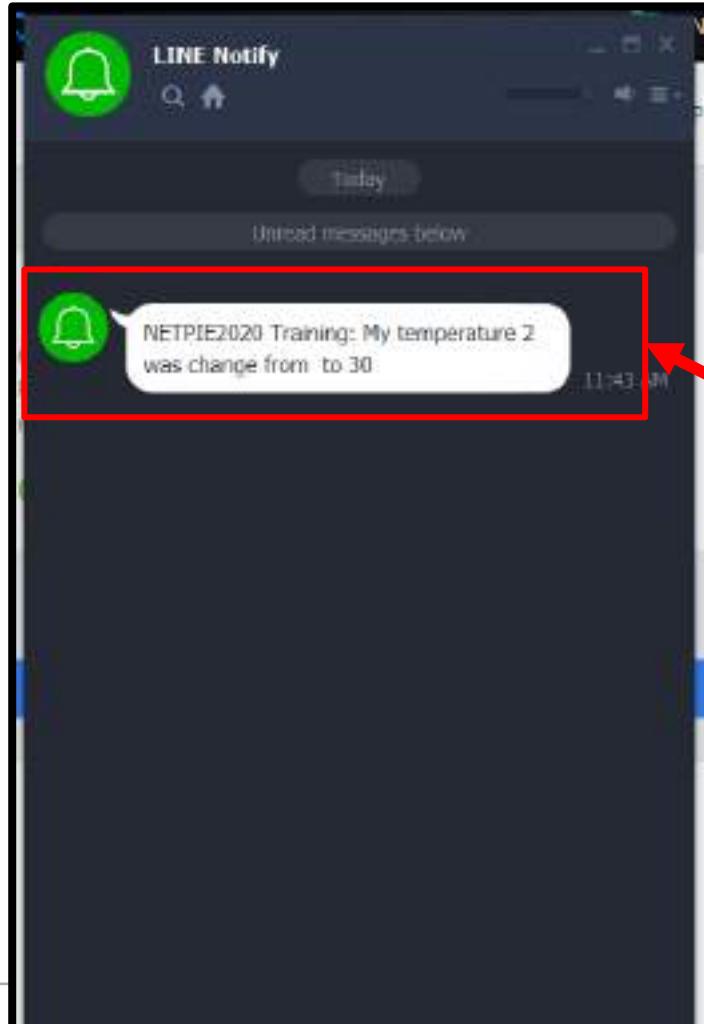
# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Exercise 1 : Create Device Trigger



# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Exercise 1 : Create Device Trigger



Temperature change notification from LINE

# 1 - Data Management in NETPIE2020 (Trigger & Event Hook)

## Exercise 1 : Create Device Trigger

The image shows two screenshots illustrating the creation of a device trigger. On the left, the MQTTBox interface is displayed. A red box highlights the 'Not Connected' status indicator, which is being clicked by a cursor. A red arrow points from this click to a callout box labeled 'Device status change test'. The MQTTBox interface includes fields for Topic to publish (@shadow/data/update), QoS (0 - Almost Once), Retain, Payload Type (Strings / JSON / XML / Characters), and Payload ({"data": {"temperature": 30}}). At the bottom, a preview of the MQTT message is shown: {"data": {"temperature": 30}}, topic:@shadow/data/update, qos:0, retain:false. On the right, a LINE Notify screenshot shows a message: 'NETPIE2020 Training: My temperature 2 was change from 30 to 30'. Another message at the bottom is highlighted with a red box and a red arrow pointing to a callout box labeled 'Device status change notification LINE'. This message reads: 'NETPIE2020 Training: My Device offline, statuscode: 0'.

Device status change test

Device status change notification LINE

MQTTBox

MQTTBox Edit Help

Not Connected

Add publisher Add subscriber

Topic to publish: @shadow/data/update

QoS: 0 - Almost Once

Retain

Payload Type: Strings / JSON / XML / Characters

e.g. {hello:world}

Payload: {"data": {"temperature": 30}}

Publish

{"data": {"temperature": 30}}, topic:@shadow/data/update, qos:0, retain:false

LINE Notify

NETPIE2020 Training: My temperature 2 was change from 30 to 30

NETPIE2020 Training: My Device offline, statuscode: 0

By Piyawat Jomsathan

Page 33

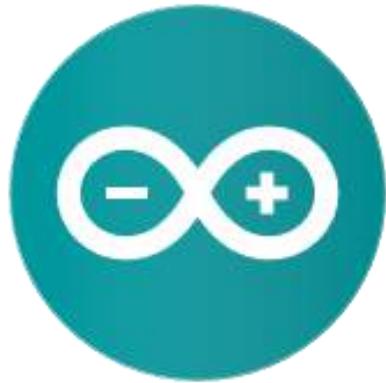
2

# MQTT Library in Arduino IDE (Pubsubclient)



## 2 - MQTT Library in Arduino IDE (Pubsubclient)

In order for ESP32 or ESP8266 to connect to NETPIE2020, it needs to be programmed on the Arduino IDE to connect ESP32 or ESP8266 to internet and then to NETPIE2020.



**Arduino IDE**

**ESP32 or ESP8266**



**Internet  
connection**  
→  
**through WiFi**



**NETPIE2020**

## 2 - MQTT Library in Arduino IDE (Pubsubclient)



**ESP32 or ESP8266**

**In order for ESP32 or ESP8266 to connect to NETPIE2020, the following libraries are needed**

**1. WiFi**

**Used to allow ESP32 or ESP8266 to connect to the Internet via WiFi network.**

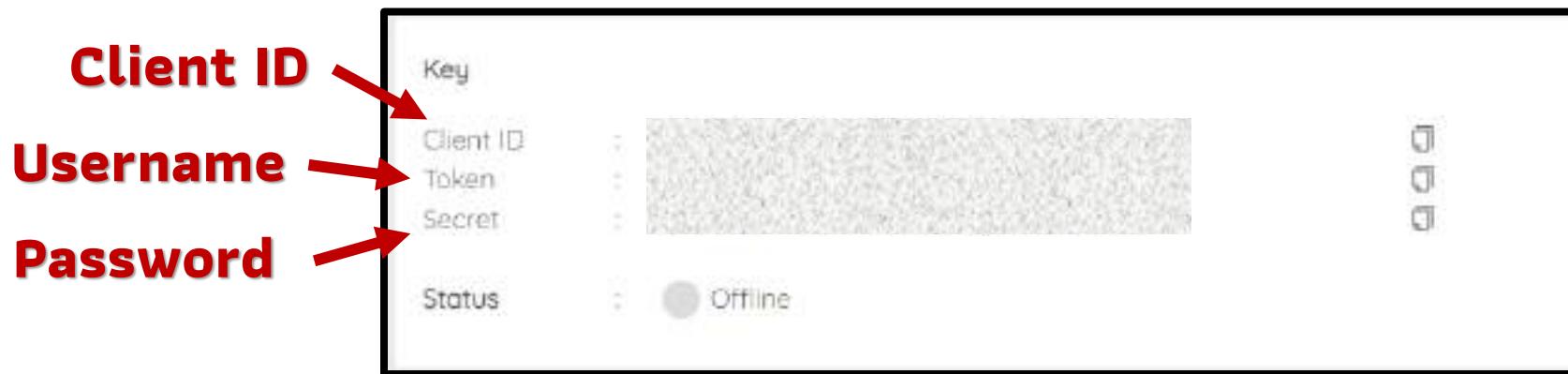
**2. PubSubClient**

**Used for ESP32 or ESP8266 to connect and communicate on NETPIE 2020 platform.  
[MQTT Protocol]**

## 2 - MQTT Library in Arduino IDE (Pubsubclient)

**NETPIE2020 connection requires 4 parameters as follows:**

1. Host : broker.netpie.io
2. Client ID : Client ID of Device created in NETPIE2020 portal.
3. Username : Token of Device created in NETPIE2020 portal.
4. Password : Secret of Device created n NETPIE2020 portal.  
(used for added security)



## 2 - MQTT Library in Arduino IDE (Pubsubclient)

### Exercise 2 : Connecting the NETPIE2020 with ESP32 or ESP8266



**ESP32 or ESP8266**

**Connect**



**NETPIE2020**

**Please check condition**

- 1. Install Arduino IDE**
- 2. Install ESP32 or ESP8266 Hardware.**
- 3. Install PubSubClient Library**

# 2 - MQTT Library in Arduino IDE (Pubsubclient)

## Exercise 2 : Connecting the NETPIE2020 with ESP32 or ESP8266

Coding in connect to NETPIE2020 consists of 3 sections

1

Section 1 : Library and variable declaration

Call to various libraries.

Variable declaration for connecting to WiFi

Variable declaration for NETPIE2020 connection

Create instances for NETPIE2020 connection

```
#include <WiFi.h>
#include <PubSubClient.h>
```

```
const char* ssid = "Your WiFi SSID";
const char* password = "Your Password";
```

```
const char* mqtt_server = "broker.netpie.io";
const int mqtt_port = 1883;
const char* mqtt_Client = "Your Client-ID";
const char* mqtt_username = "Your Token";
const char* mqtt_password = "Your Secret";
```

```
WiFiClient espClient;
PubSubClient client[espClient];
```

# 2 - MQTT Library in Arduino IDE (Pubsubclient)

## Exercise 2 : Connecting the NETPIE2020 with ESP32 or ESP8266

2

### Section 2 : functions

Function for NETPIE2020 connection

```
void reconnect() {  
    while (!client.connected()) {  
        Serial.print("Attempting NETPIE2020 connection...");  
        if [client.connect[mqtt_Client, mqtt_username, mqtt_password]] {  
            Serial.println("NETPIE2020 connected");  
        }  
        else {  
            Serial.print("failed, rc=");  
            Serial.print(client.state());  
            Serial.println("try again in 5 seconds");  
            delay(5000);  
        }  
    }  
}
```

Enter NETPIE2020 connection.

- If connected successfully, it will display "connected".
- If the connection is not successful, it will display "failed..." and automatically reconnect.

# 2 - MQTT Library in Arduino IDE (Pubsubclient)

## Exercise 2 : Connecting the NETPIE2020 with ESP32 or ESP8266

2

### Section 2 : functions

```
void setup() {  
    Serial.begin(115200);  
    Serial.println("Starting...");  
    if (WiFi.begin(ssid, password)) {  
        while (WiFi.status() != WL_CONNECTED) {  
            delay(1000);  
            Serial.print(".");  
        }  
        Serial.println("WiFi connected");  
        Serial.println("IP address: ");  
        Serial.println(WiFi.localIP());  
        client.setServer(mqtt_server, mqtt_port);  
    }  
}
```

Performing initialization

In this function, it will connect to WiFi and NETPIE2020 according to the settings.

# 2 - MQTT Library in Arduino IDE (Pubsubclient)

## Exercise 2 : Connecting the NETPIE2020 with ESP32 or ESP8266

3

### Section 3 : loop() function

```
void loop() {
```

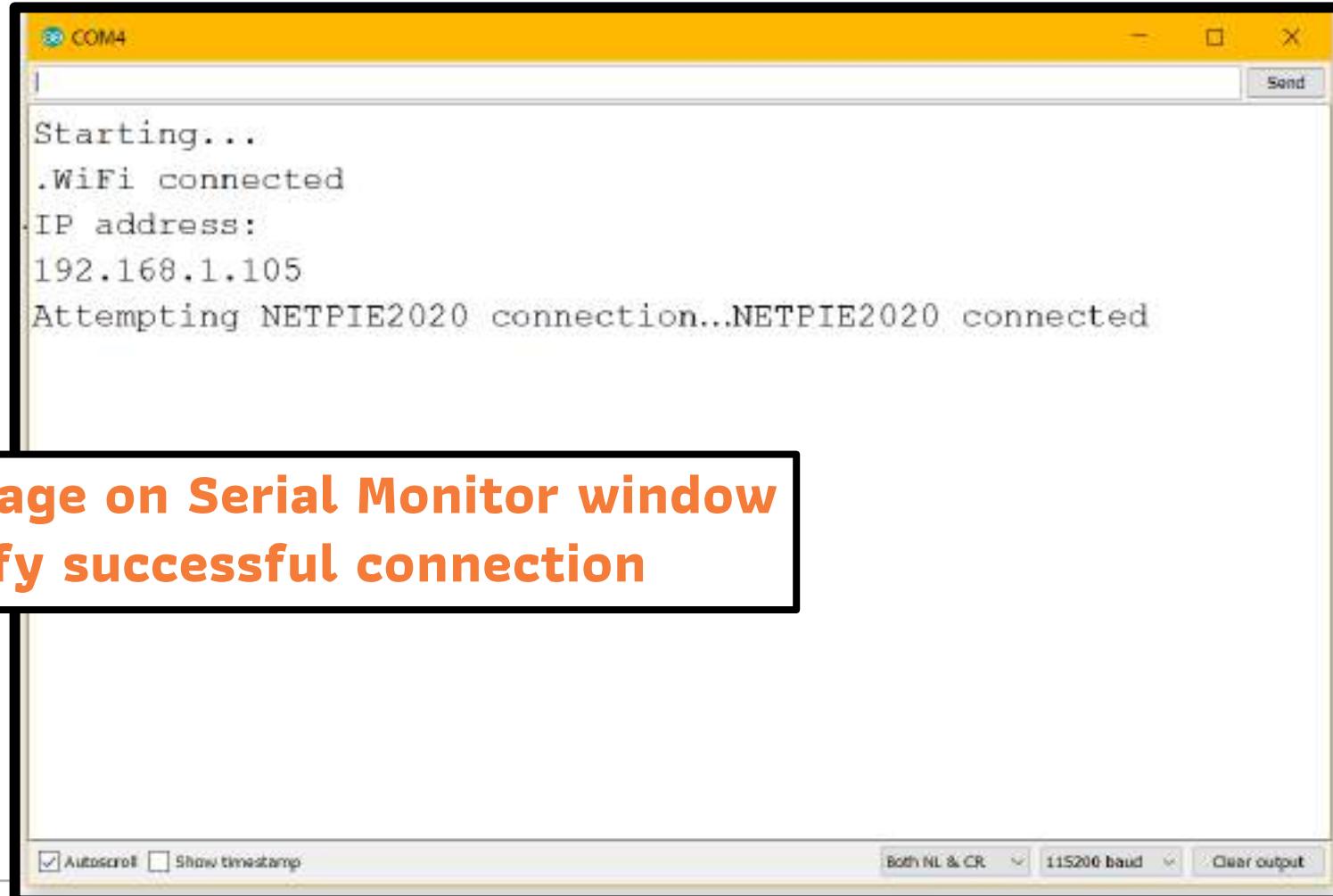
Main function that runs  
iteratively

```
    if (!client.connected()) {  
        reconnect();  
    }  
    client.loop();  
}
```

Command set that maintains the  
connection status and functions of  
NETPIE2020.

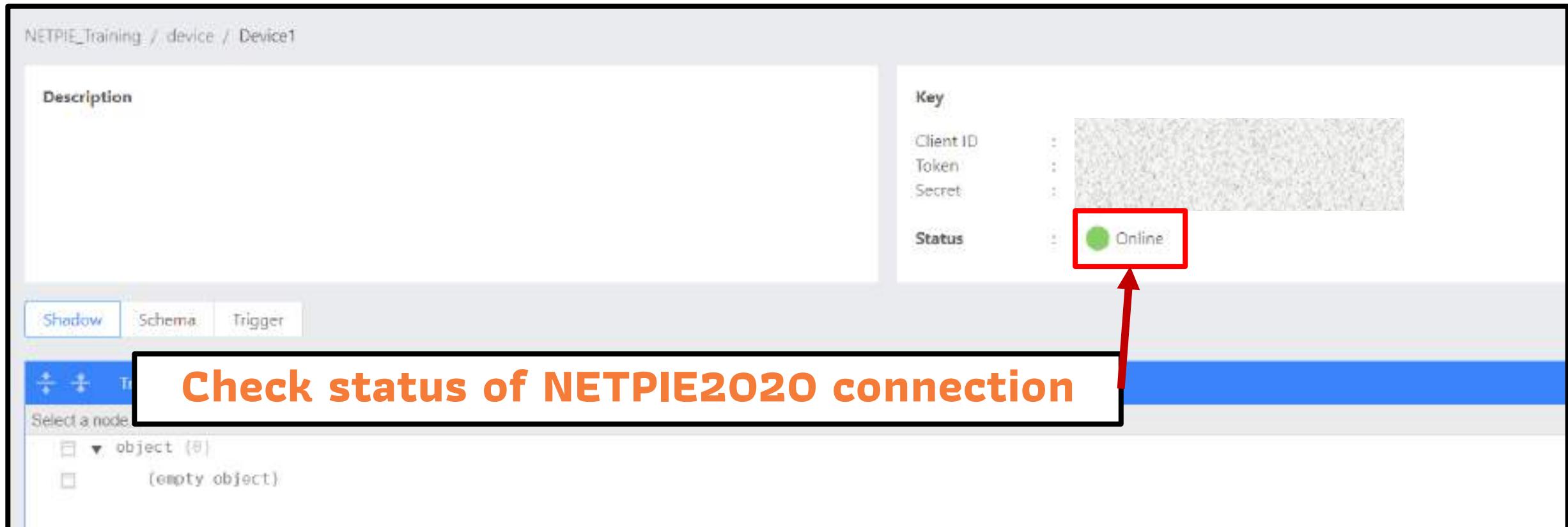
## 2 - MQTT Library in Arduino IDE (Pubsubclient)

### Exercise 2 : Connecting the NETPIE2020 with ESP32 or ESP8266



# 2 - MQTT Library in Arduino IDE (Pubsubclient)

## Exercise 2 : Connecting the NETPIE2020 with ESP32 or ESP8266



## 2 - MQTT Library in Arduino IDE (Pubsubclient)

Exercise 3 : Communication on NETPIE2020 with ESP32 or ESP8266 [Publish]



**ESP32 or ESP8266**

**“Hello NETPIE2020”**



**NETPIE2020**

## 2 - MQTT Library in Arduino IDE (Pubsubclient)

### Exercise 3 : Communication on NETPIE2020 with ESP32 or ESP8266 [Publish]

Coding in Exercise2 has the beginning part WiFi and NETPIE2020 connection similar Exercise2.  
Only the void loop[] part is added.

client.publish is a command to publish messages to NETPIE2020 using format  
**client.publish ["Topic", "Message"];**

```
void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();

    client.publish("@msg/test", "Hello NETPIE2020");
    Serial.println("Hello NETPIE2020");
    delay(2000);
}
```

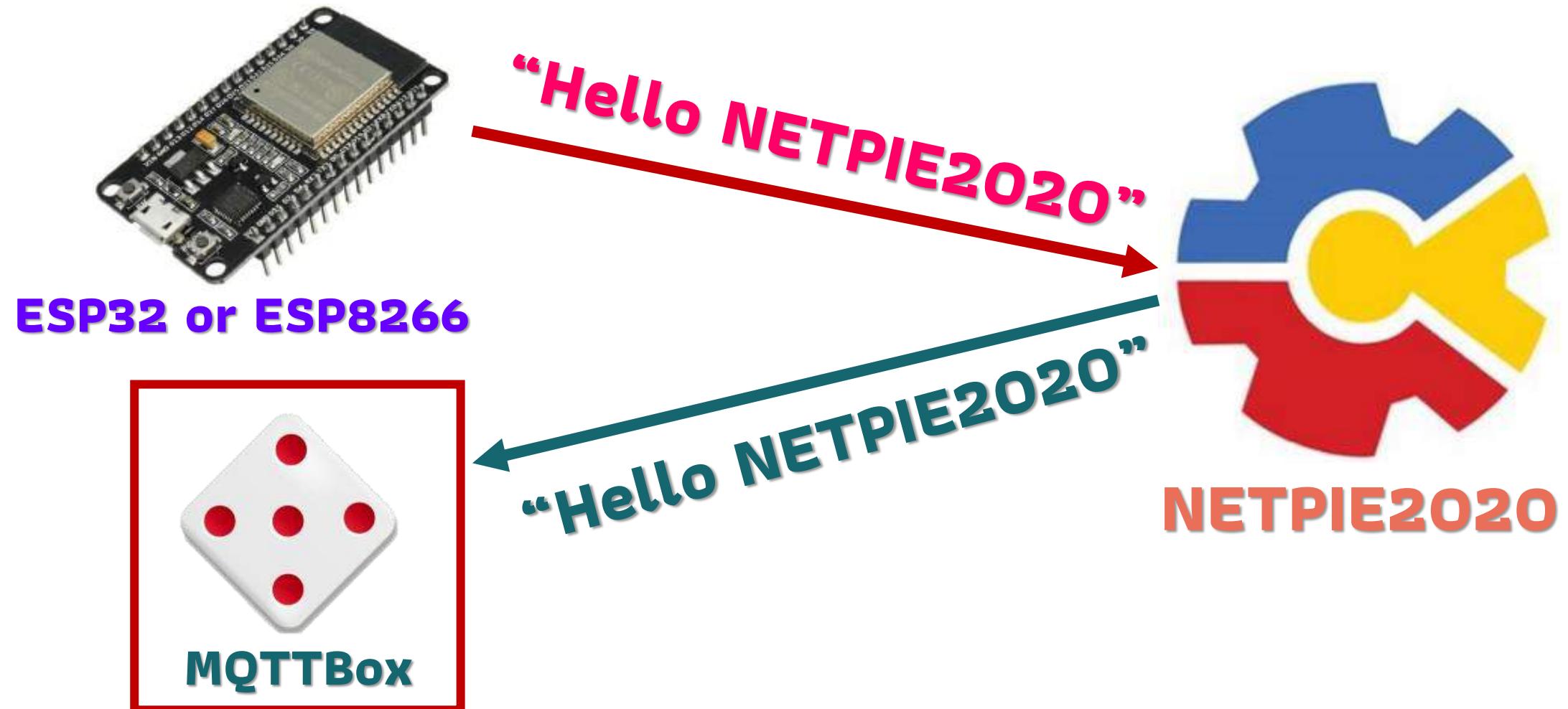
## 2 - MQTT Library in Arduino IDE (Pubsubclient)

### Exercise 3 : Communication on NETPIE2020 with ESP32 or ESP8266 [Publish]



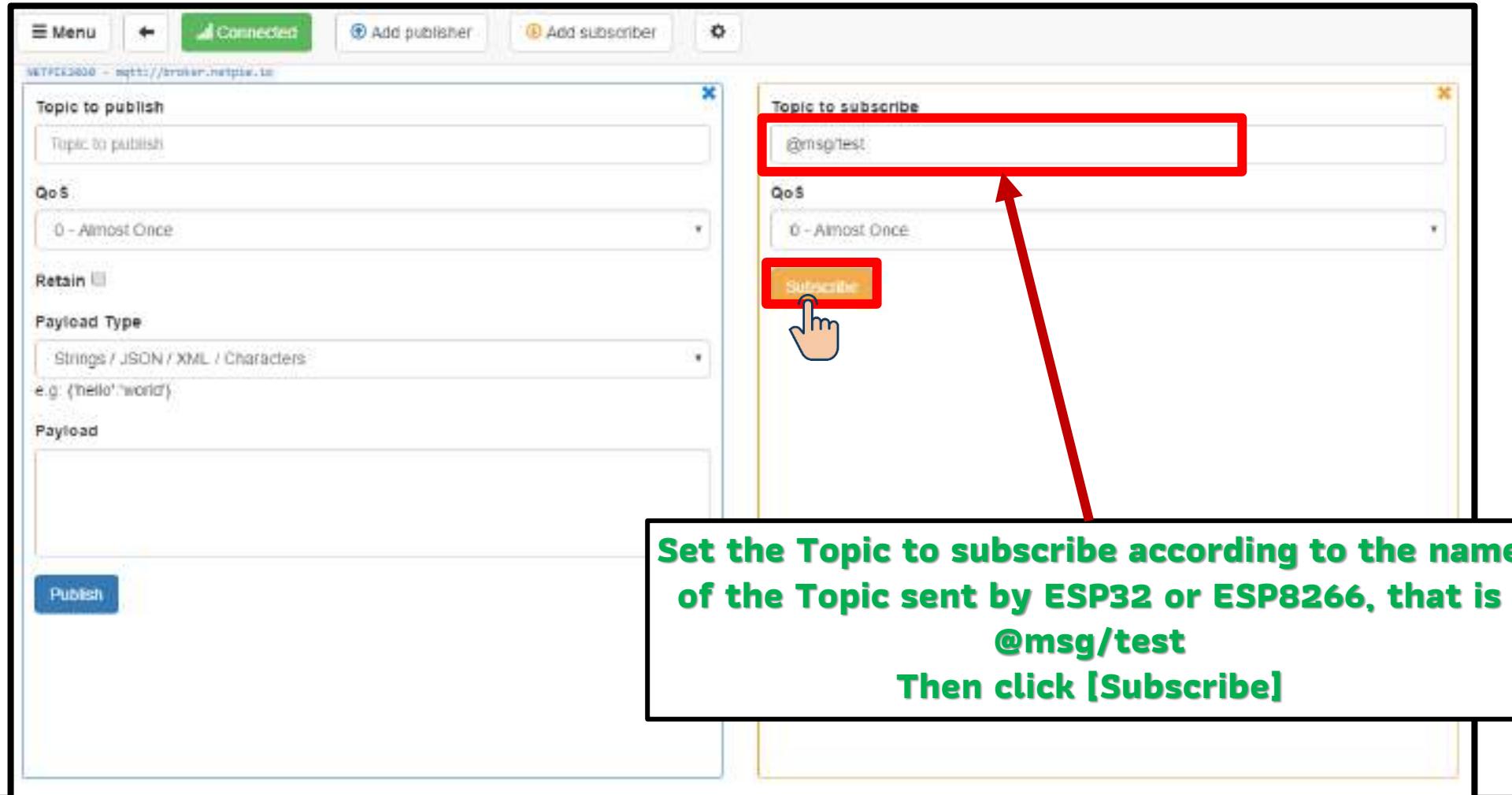
## 2 - MQTT Library in Arduino IDE (Pubsubclient)

Exercise 4 : Communication on NETPIE2020 with ESP32 or ESP8266 [Subscribe]



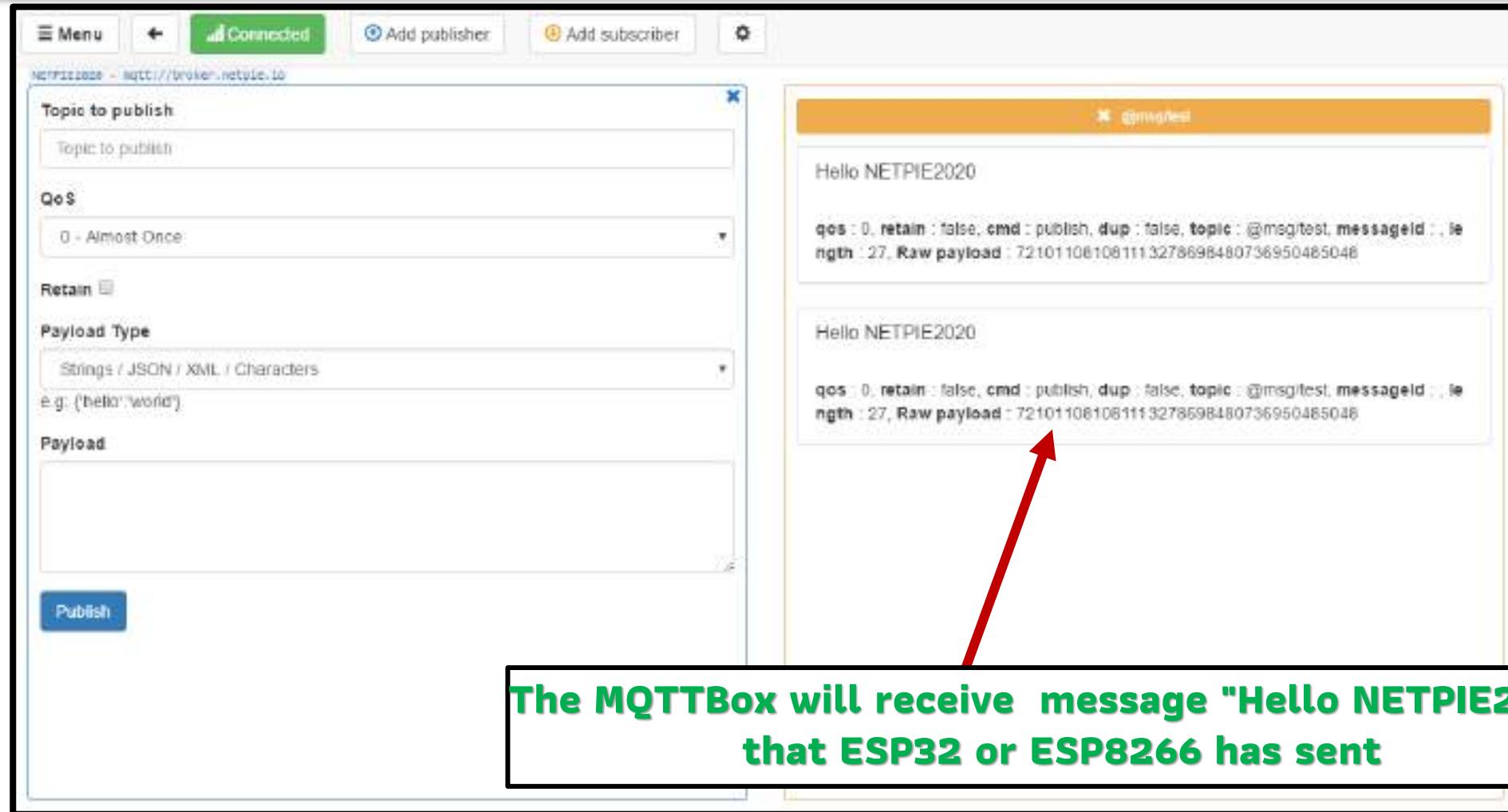
## 2 - MQTT Library in Arduino IDE (Pubsubclient)

### Exercise 4 : Communication on NETPIE2020 with ESP32 or ESP8266 [Subscribe]



## 2 - MQTT Library in Arduino IDE (Pubsubclient)

### Exercise 4 : Communication on NETPIE2020 with ESP32 or ESP8266 [Subscribe]



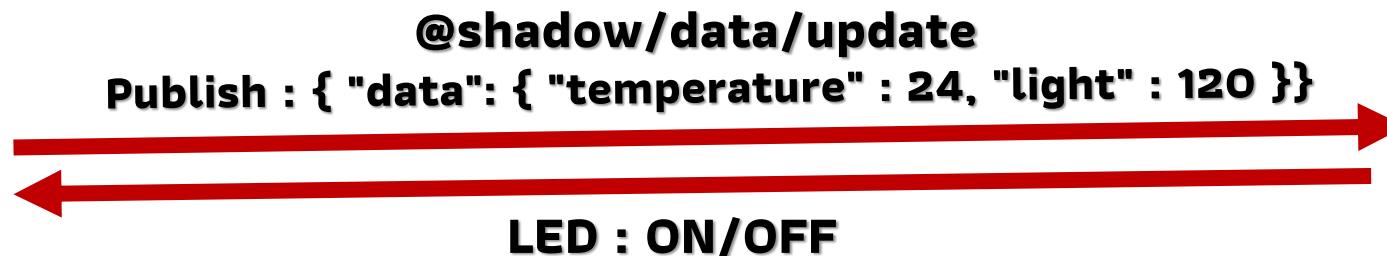
3

# Freeboard in NETPIE2020 (Control Widget)

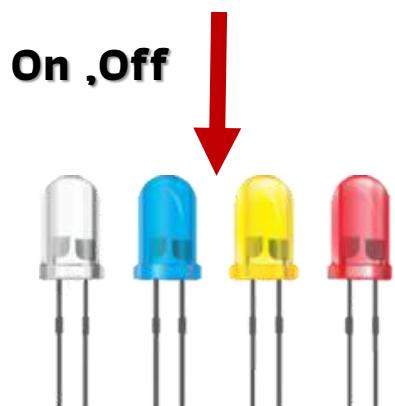


# 3 - Freeboard in NETPIE2020 (Control Widget)

Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard



**ESP32 or ESP8266**



**Freeboard**



# 3 - Freeboard in NETPIE2020 (Control Widget)

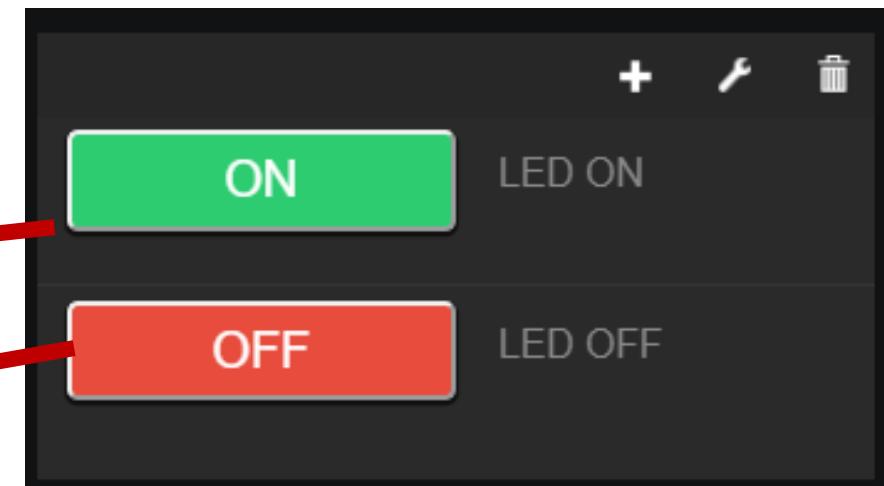
## Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard

LED on ESP32 or ESP8266 can be controlled via Freeboard by 2 methods

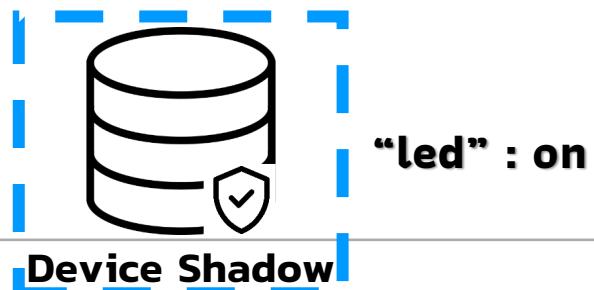
### 1. Command via Shadow



@shadow/data/update  
“led” : “on”  
@shadow/data/update  
“led” : “off”



ESP32 or ESP8266



# 3 - Freeboard in NETPIE2020 (Control Widget)

## Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard

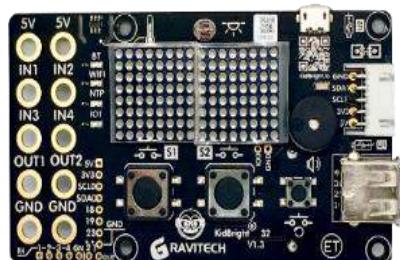
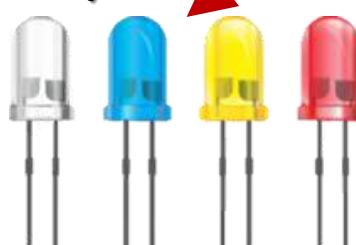
LED on ESP32 or ESP8266 can be controlled via Freeboard by 2 methods

### 1. Command via Shadow

Message sent via shadow has the following format

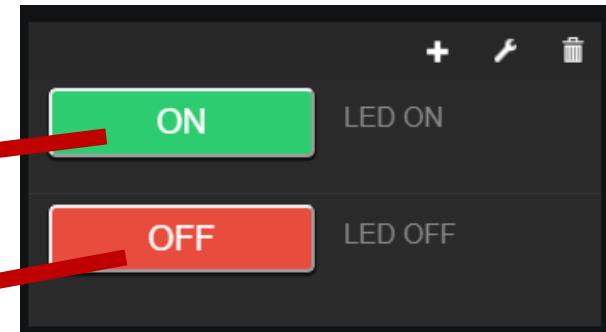
```
b'{"deviceid": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx", "data": {"led": "on"}, "rev": 3, "modified": 1579864929867'}
```

On ,Off



@shadow/data/update  
"led" : "on"

@shadow/data/update  
"led" : "off"

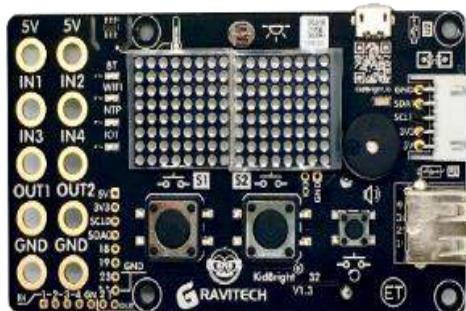


# 3 - Freeboard in NETPIE2020 (Control Widget)

## Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard

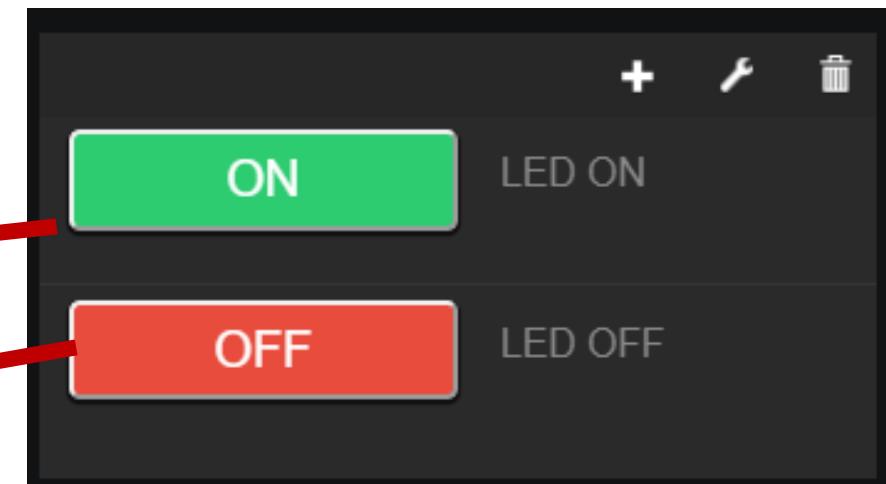
LED on ESP32 or ESP8266 can be controlled via Freeboard by 2 methods

### 2. Command by Message



@msg/led  
“on”

@msg/led  
“off”



@shadow/data/update  
“led” : “on”



“led” : “on”

# 3 - Freeboard in NETPIE2020 (Control Widget)

## Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard

2

### Part 2 various functions

### In Coding

```
void reconnect() {
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        if (client.connect(mqtt_Client, mqtt_username, mqtt_password)) {
            Serial.println("connected");
            client.subscribe("@msg/led");
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println("try again in 5 seconds");
            delay(5000);
        }
    }
}
```

MQTT connection function

Subscribe Topic sent by Freeboard

# 3 - Freeboard in NETPIE2020 (Control Widget)

## Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard

2

### Part 2 various functions

#### MQTT message receive function

```
void callback(char* topic, byte* payload, unsigned int length) {
```

```
Serial.print("Message arrived [");  
Serial.print(topic);  
Serial.print("] ");  
String message;  
for (int i = 0; i < length; i++) {  
    message = message + (char)payload[i];  
}  
Serial.println(message);
```

#### Commands to display MQTT topic and message

```
if(String(topic) == "@msg/led") {  
    if (message == "on"){  
        digitalWrite(LED1,0);  
        client.publish("@shadow/data/update", "{\"data\" : {\"led\" : \"on\"}}");  
        Serial.println("LED ON");  
    }  
    else if (message == "off") {  
        digitalWrite(LED1,1);  
        client.publish("@shadow/data/update", "{\"data\" : {\"led\" : \"off\"}}");  
        Serial.println("LED OFF");  
    }  
}
```

LED on / off condition :starting by checking whether topic is @ msg / led or not And check the received message whether it is on or off .Because the LED on the ESP8266 or ESP32 is a pull-up circuit. Thus we must send 0 to the LED pin to turn on, and send 1 to turn off.

# 3 - Freeboard in NETPIE2020 (Control Widget)

## Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard

2

### Part 2 various functions

```
void setup() {  
    Serial.begin(115200);  
    Wire1.begin(4, 5);  
    pinMode(LED, OUTPUT);  
    digitalWrite(LED, 1);  
  
    Serial.println("Starting...");  
    if (WiFi.begin(ssid, password)) {  
        while (WiFi.status() != WL_CONNECTED) {  
            delay(1000);  
            Serial.print(".");  
        }  
    }  
    Serial.println("WiFi connected");  
    Serial.println("IP address: ");  
    Serial.println(WiFi.localIP());  
    client.setServer(mqtt_server, mqtt_port);  
    client.setCallback(callback);
```

### Setup function

Declare LED pin as OUTPUT  
And start by turning it off

Assign callback function to MQTT

# 3 - Freeboard in NETPIE2020 (Control Widget)

## Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard

3

### Part 3 loop function

```
void loop() {  
    if (!client.connected()) {  
        reconnect();  
    }  
    client.loop();  
    temperature = readTemperature();  
    light = readlight();  
    String place = "NECTEC";
```

Use millis[] for timing instead of delay[] to avoid blocking the execution while waiting Freeboard message

```
long now = millis();  
if (now - lastMsg > 5000) {  
    lastMsg = now;  
    ++value;  
    String data = "{\"data\": {\"light\": " + String(light) + ", \"temperature\": " + String(temperature) + ", \"place\": \"" + String(place) + "\"}}";  
    Serial.println(data);  
    data.toCharArray(msg, [data.length() + 1]);  
    client.publish["@shadow/data/update", msg]; }  
delay[1];
```

# 3 - Freeboard in NETPIE2020 (Control Widget)

## Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard

Device Schema in JSON format

```
{  
  "additionalProperties": false,  
  "properties": {  
    "light": {  
      "operation": {  
        "store": {  
          "ttl": "7d"  
        }  
      },  
      "type": "number"  
    },  
    "temperature": {  
      "operation": {  
        "store": {  
          "ttl": "7d"  
        }  
      },  
      "transform": {  
        "expression": "[$.temperature]*1.8 + 32"  
      },  
      "type": "number"  
    }  
  }  
}
```

Add code on Schema for LED status to record

```
  "place": {  
    "operation": {  
      "store": {  
        "ttl": "7d"  
      }  
    },  
    "type": "string"  
  },  
  "led": {  
    "operation": {  
      "store": {  
        "ttl": "7d"  
      }  
    },  
    "type": "string"  
  }  
}
```

Record **led** data with property  
Store for 7 days  
Variable type is string

# 3 - Freeboard in NETPIE2020 (Control Widget)

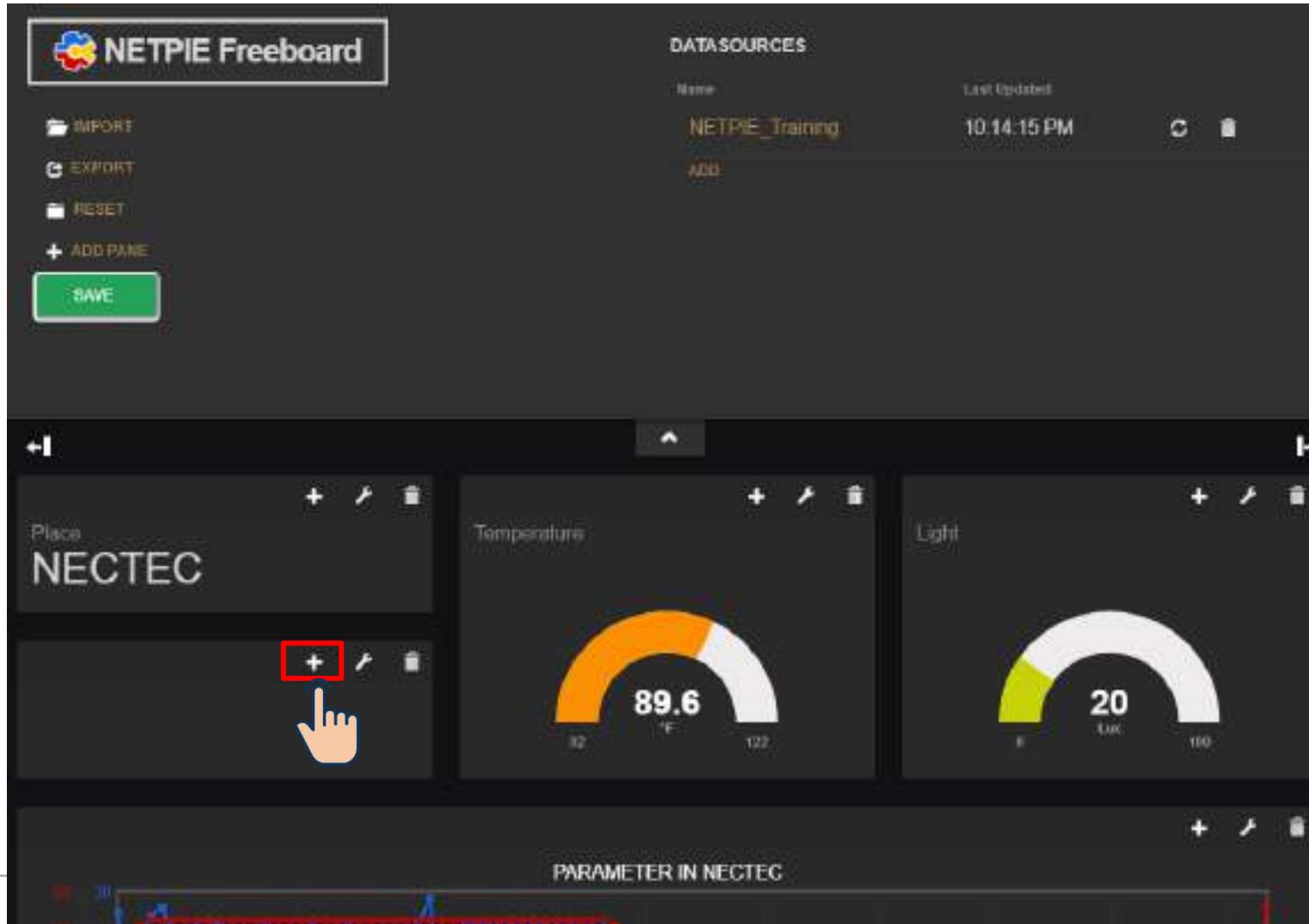
## Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard

Create a button widget to control LED

The screenshot shows the NETPIE Freeboard interface. At the top left is the logo and title 'NETPIE Freeboard'. To the right is a 'DATA SOURCES' section showing a single entry named '/NETPIE\_Training' last updated at 10:14:15 PM. Below this is a 'ADD' button. The main area displays a dashboard titled 'NECTEC' with three gauges: 'Temperature' (89.6 °F) and 'Light' (20 lux). At the bottom is a section titled 'PARAMETER IN NECTEC' showing a waveform graph. On the left side of the dashboard, there is a sidebar with buttons for 'EXPORT', 'RESET', '+ ADD PAGE', and 'SAVE'.

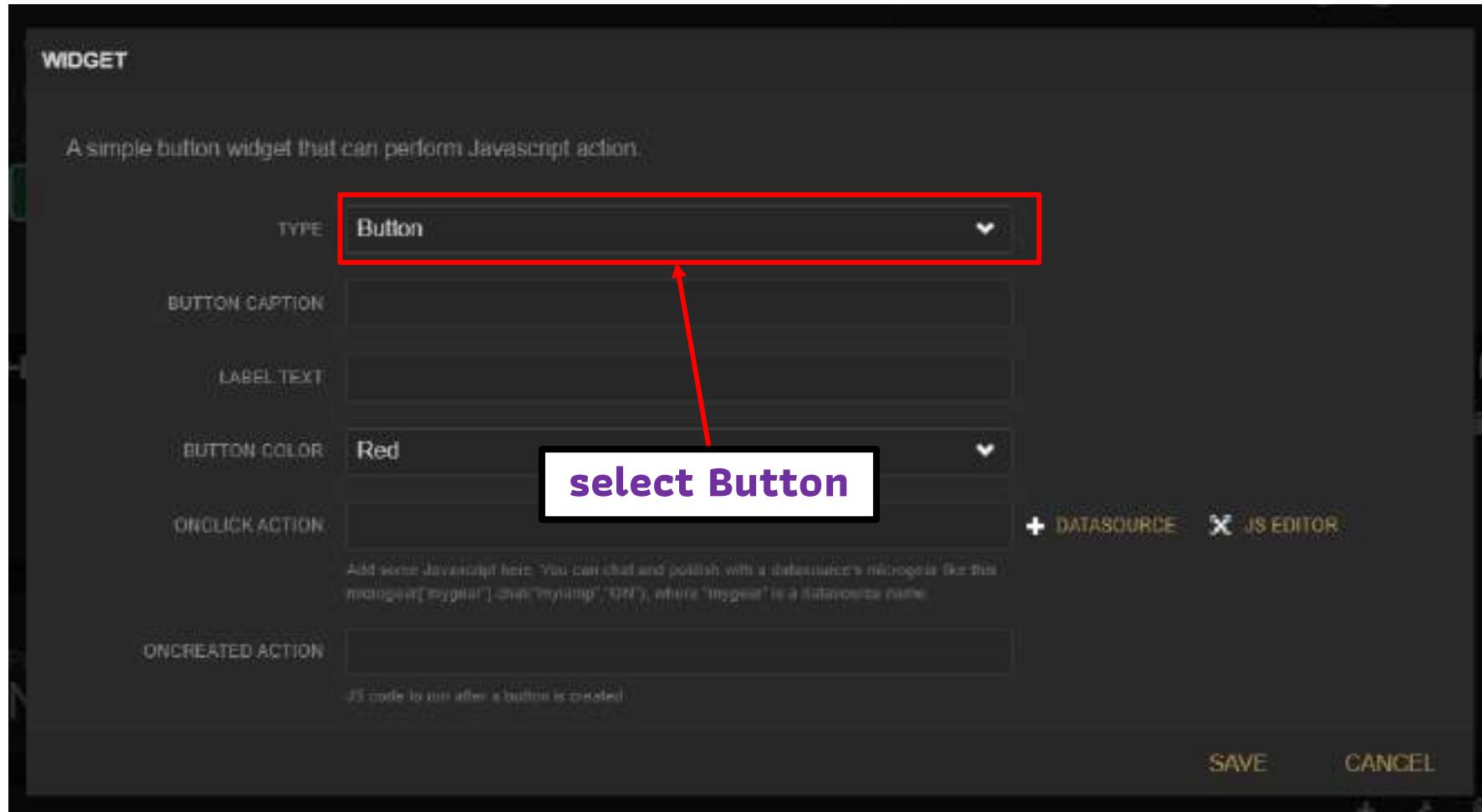
# 3 - Freeboard in NETPIE2020 (Control Widget)

Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard



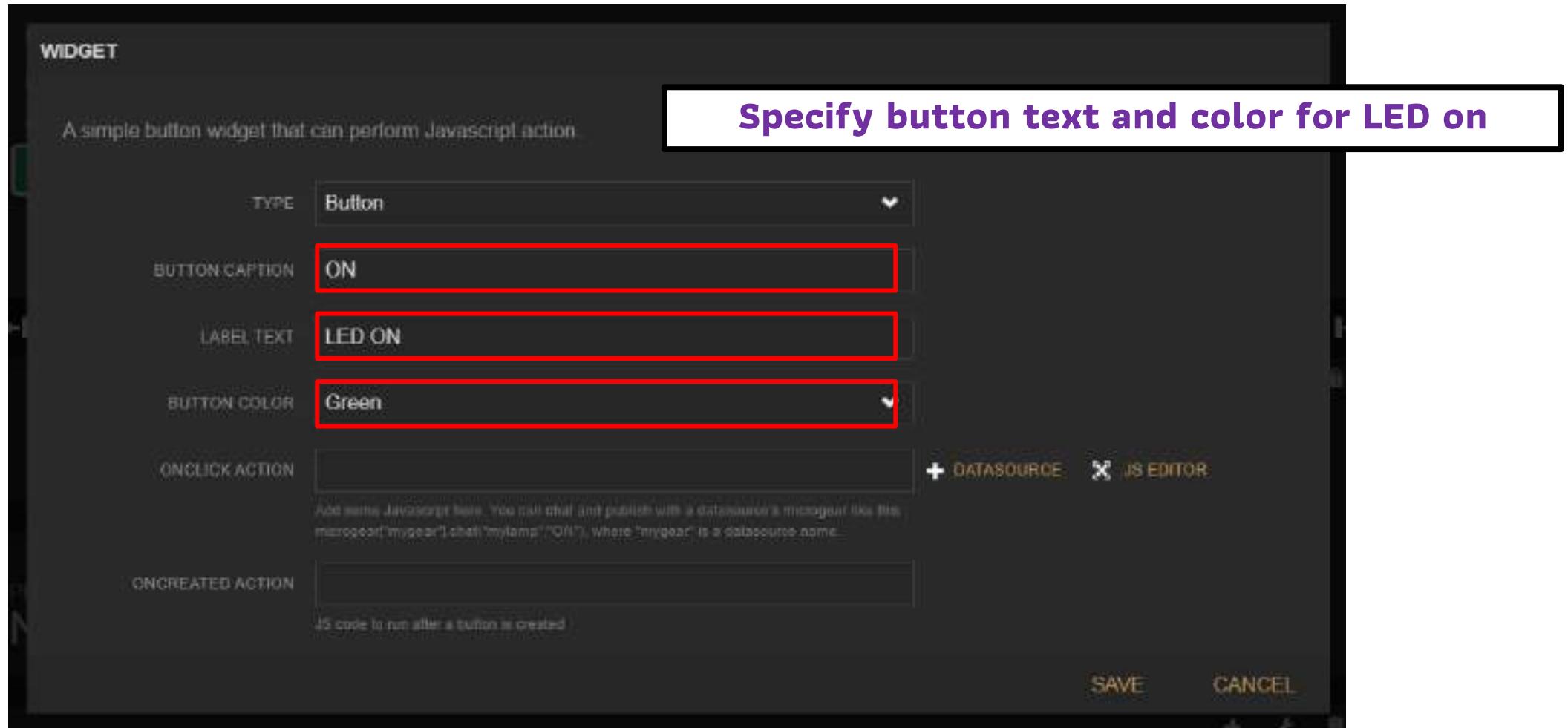
# 3 - Freeboard in NETPIE2020 (Control Widget)

## Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard



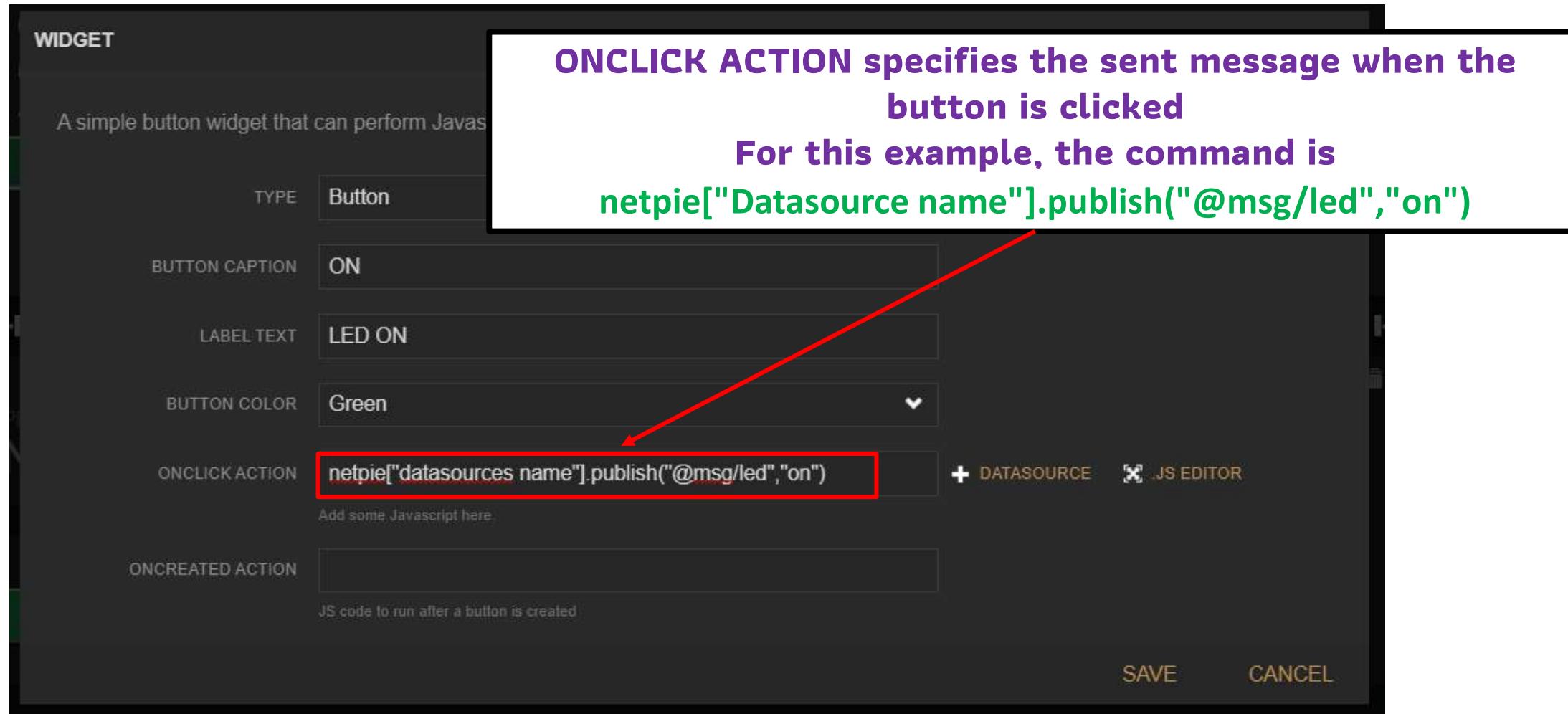
# 3 - Freeboard in NETPIE2020 (Control Widget)

## Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard



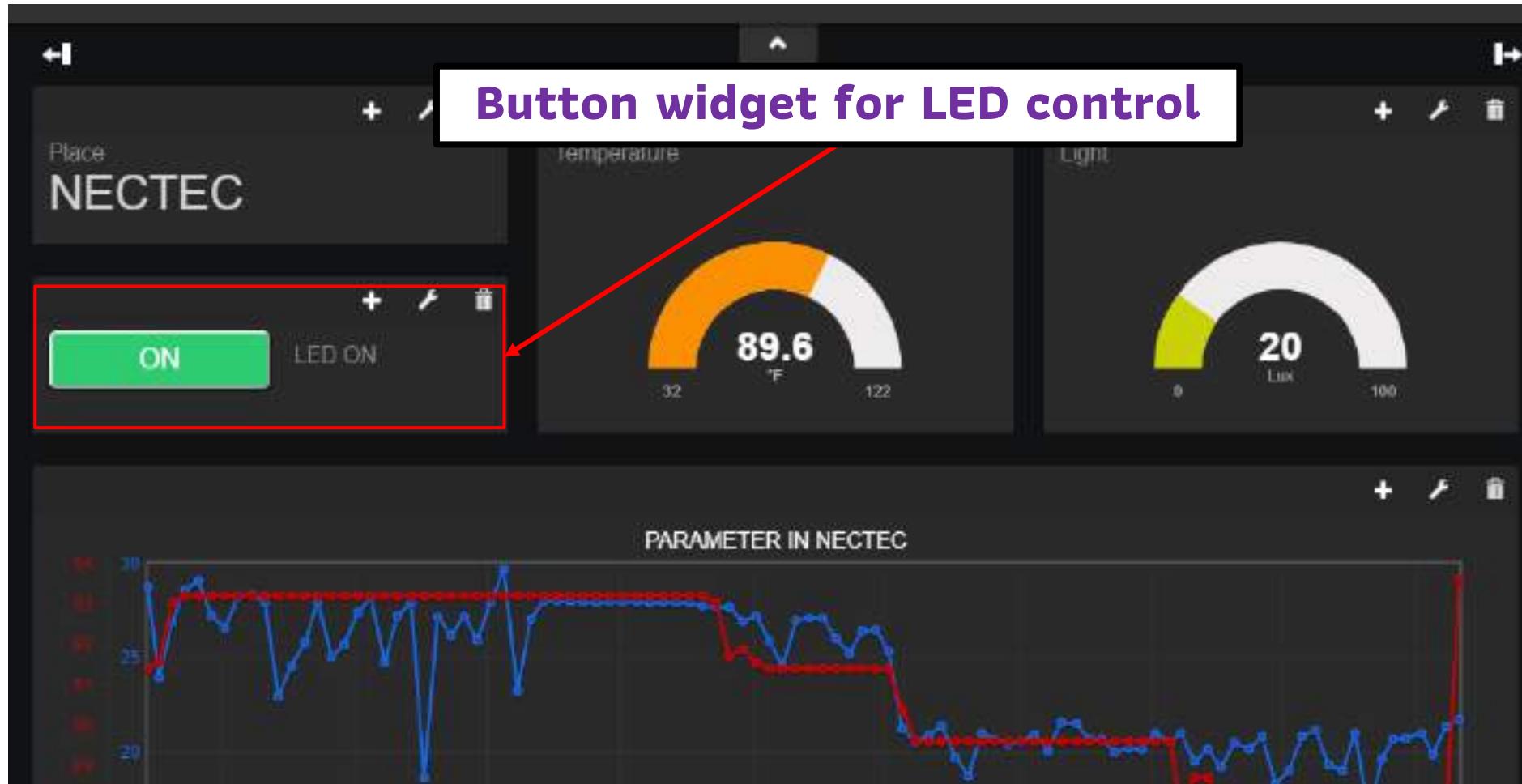
# 3 - Freeboard in NETPIE2020 (Control Widget)

## Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard



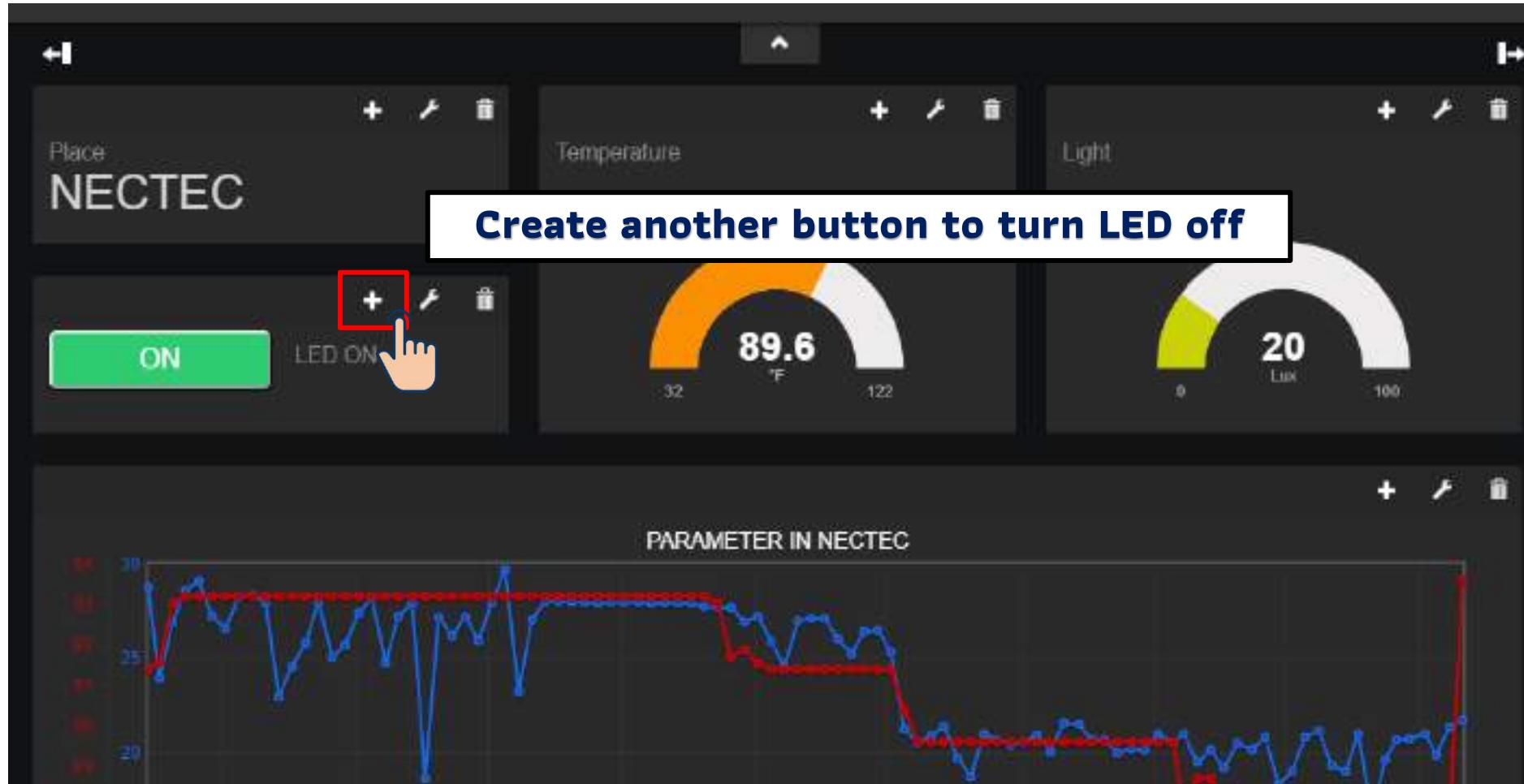
# 3 - Freeboard in NETPIE2020 (Control Widget)

Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard



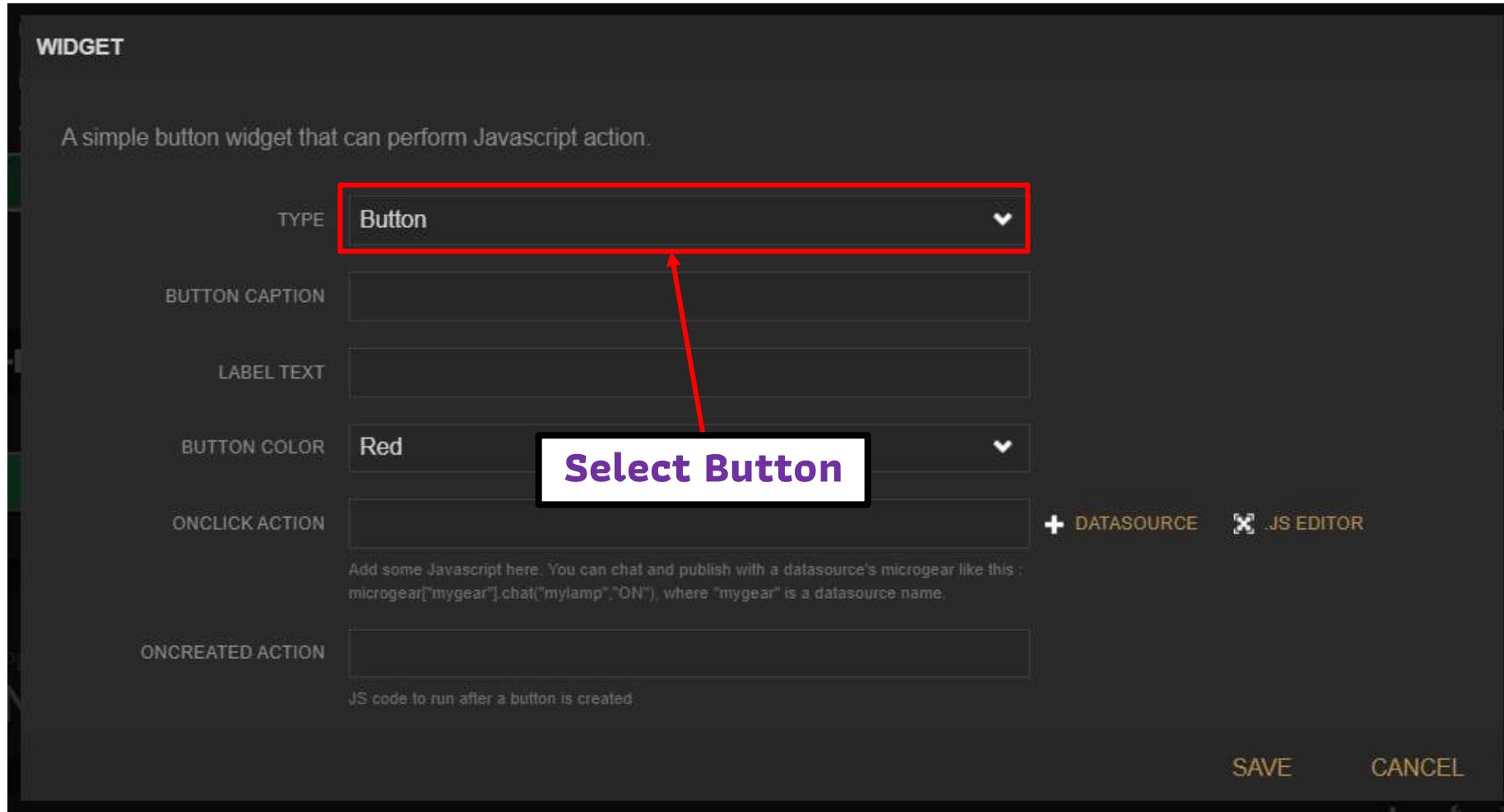
# 3 - Freeboard in NETPIE2020 (Control Widget)

Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard



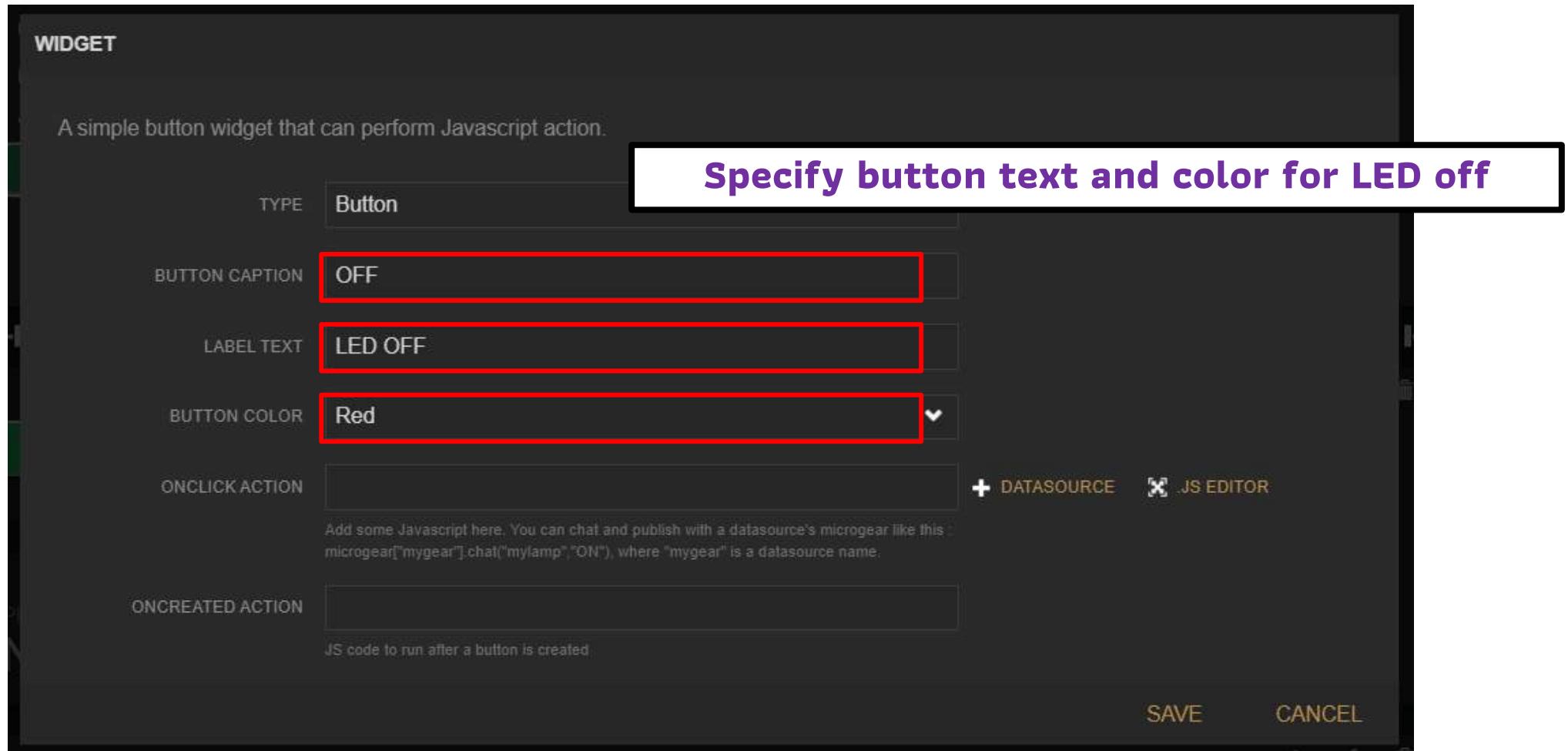
# 3 - Freeboard in NETPIE2020 (Control Widget)

## Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard



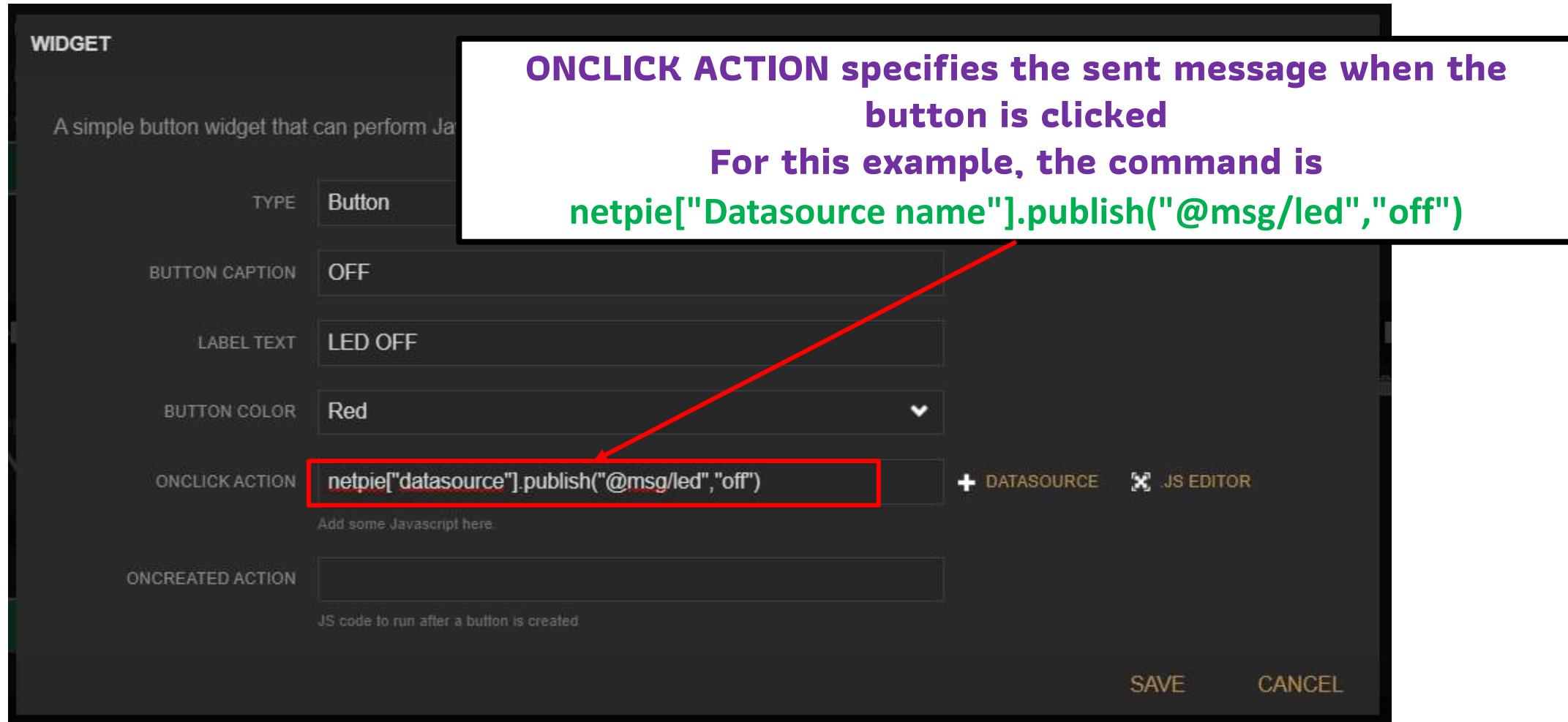
# 3 - Freeboard in NETPIE2020 (Control Widget)

## Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard



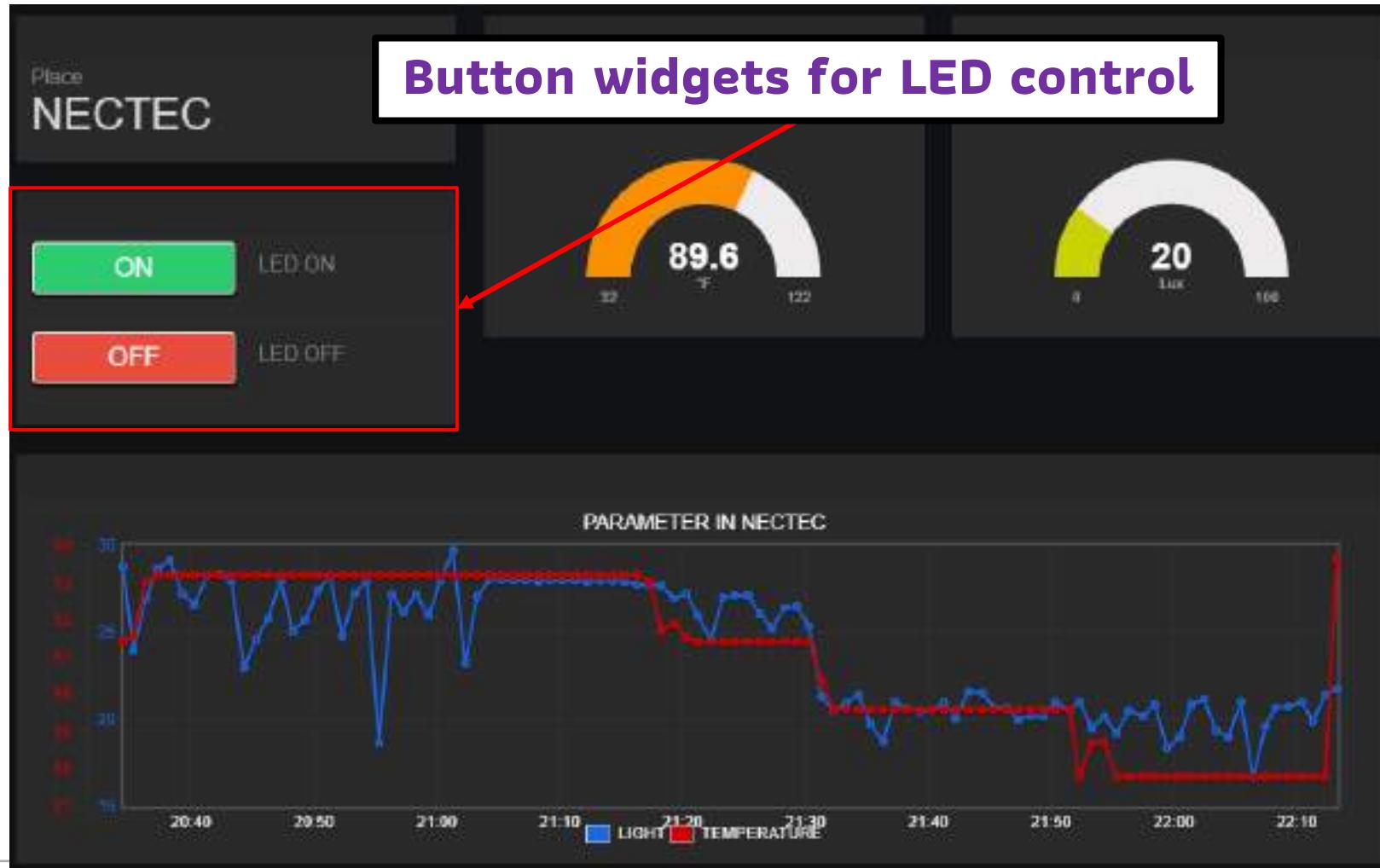
# 3 - Freeboard in NETPIE2020 (Control Widget)

## Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard



# 3 - Freeboard in NETPIE2020 (Control Widget)

Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard



# 3 - Freeboard in NETPIE2020 (Control Widget)

## Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard

The screenshot shows the Freeboard interface with the 'Shadow' tab selected. A red box highlights the 'led : off' entry under the 'object' node. A callout box labeled 'LED status on Shadow' points to this highlighted entry.

Shadow   Schema   Trigger

Tree ▾

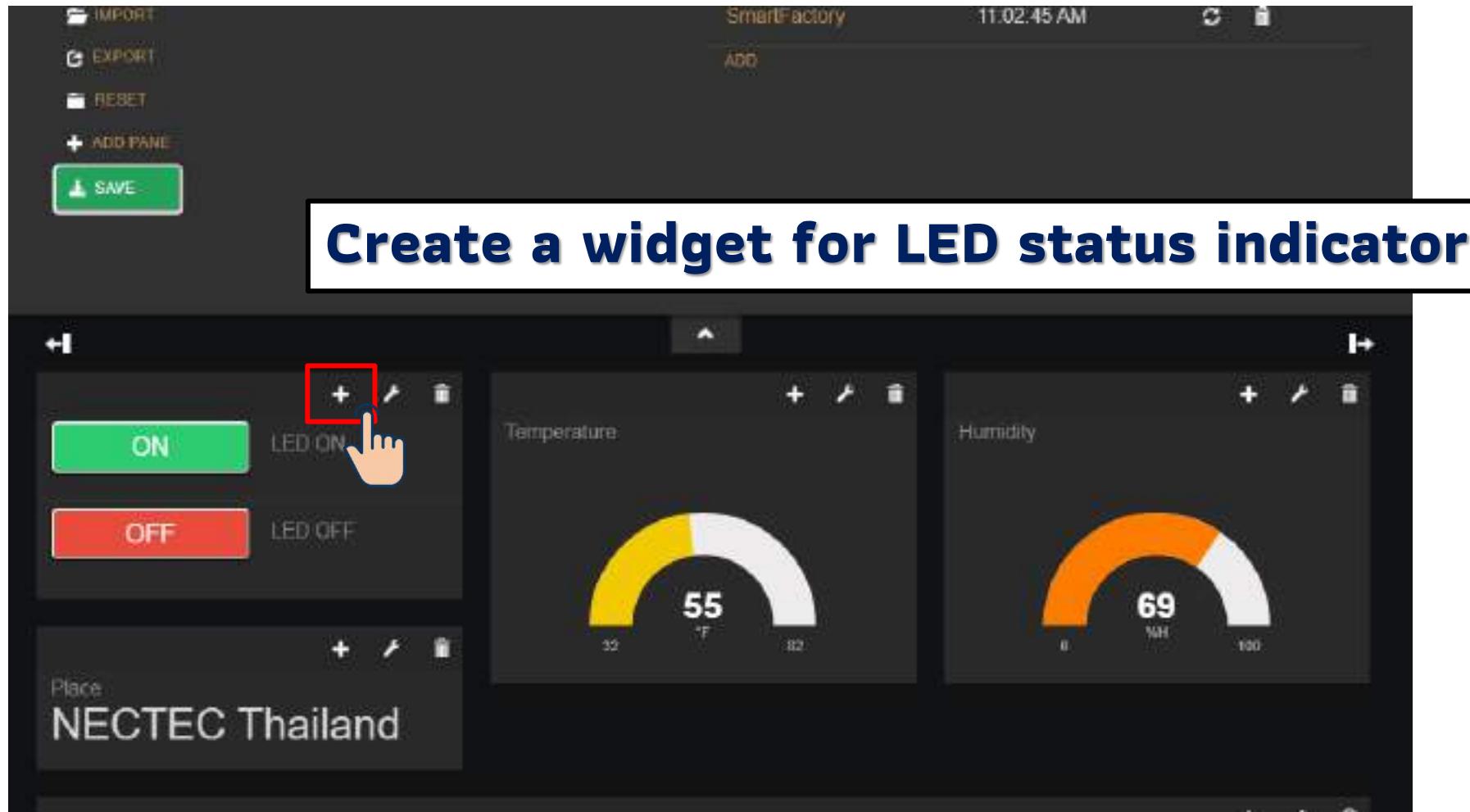
Select a node...

- object {4}
  - led : off
  - light : 20
  - place : NECTEC
  - temperature : 89.6

**LED status on Shadow**

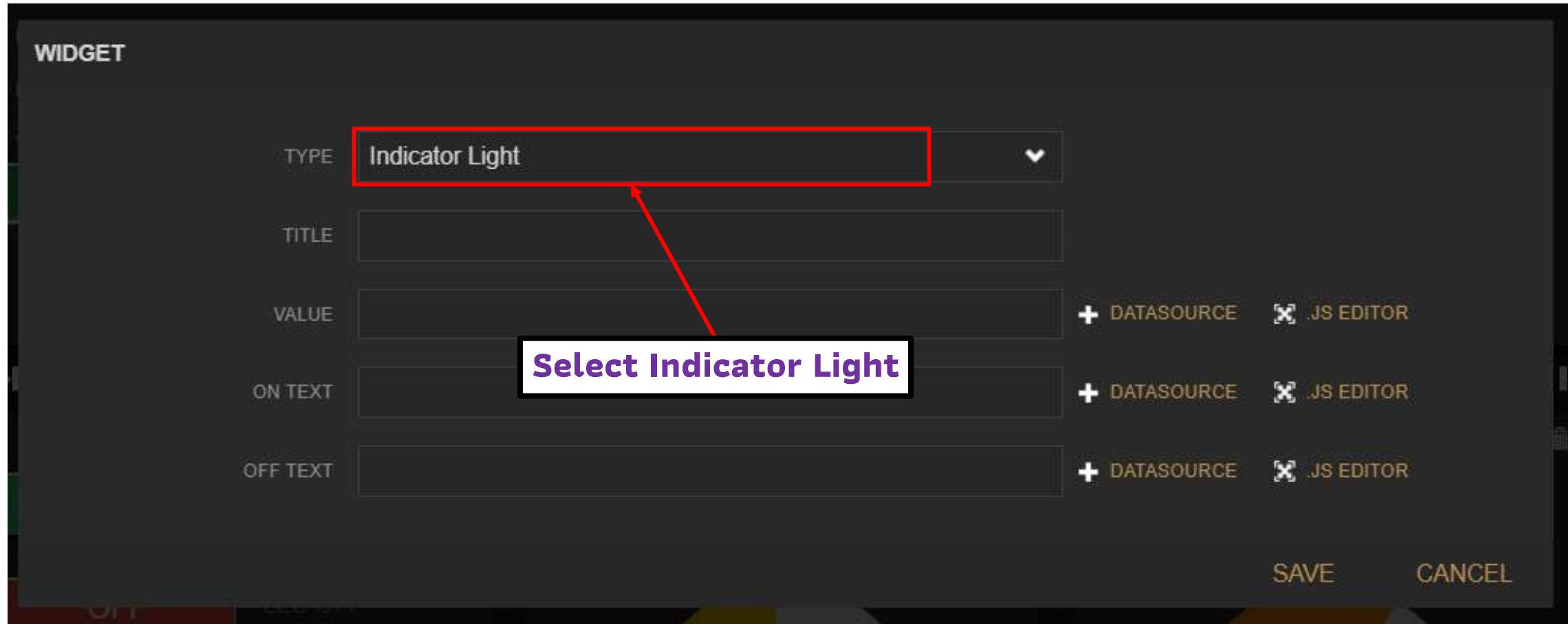
# 3 - Freeboard in NETPIE2020 (Control Widget)

Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard



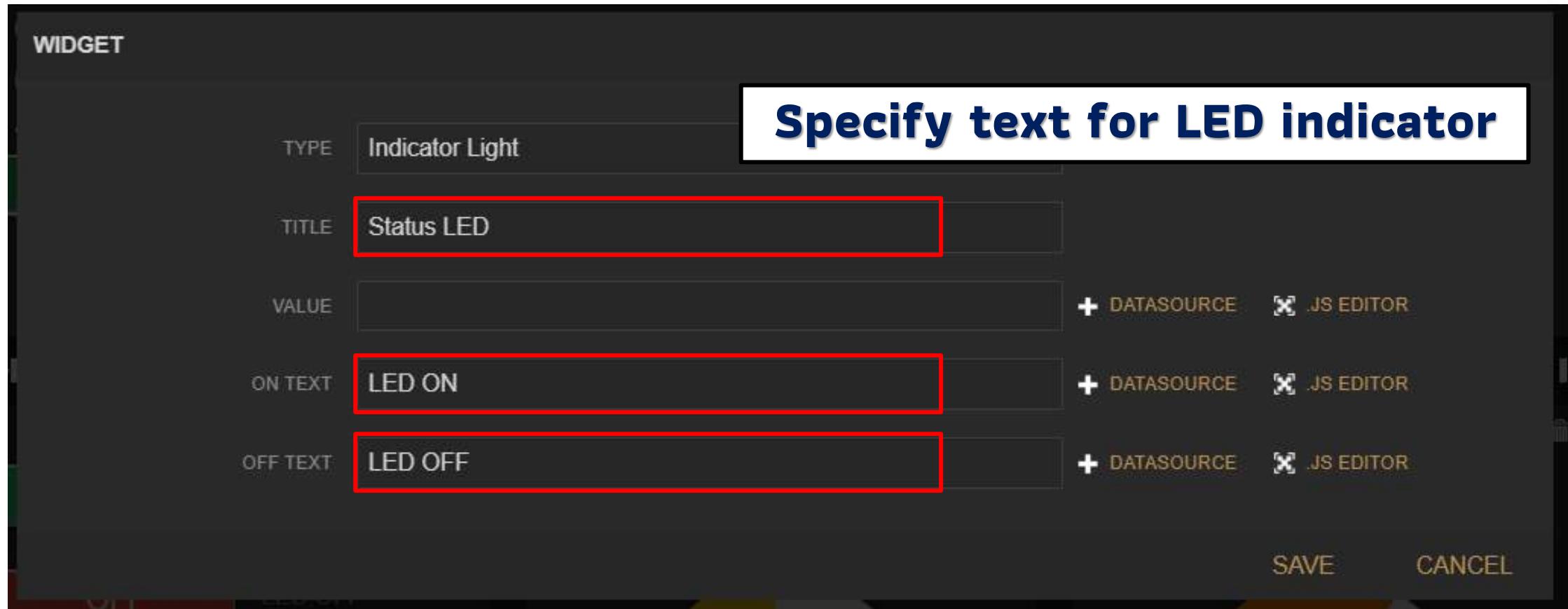
# 3 - Freeboard in NETPIE2020 (Control Widget)

## Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard



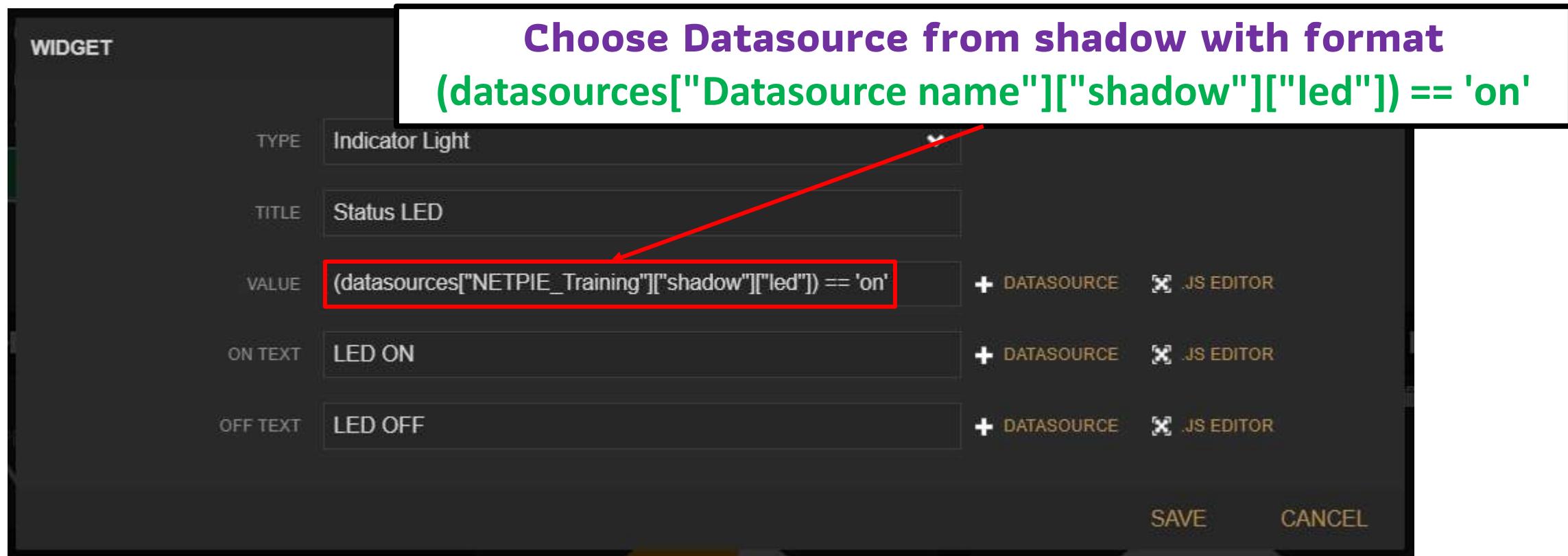
# 3 - Freeboard in NETPIE2020 (Control Widget)

## Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard



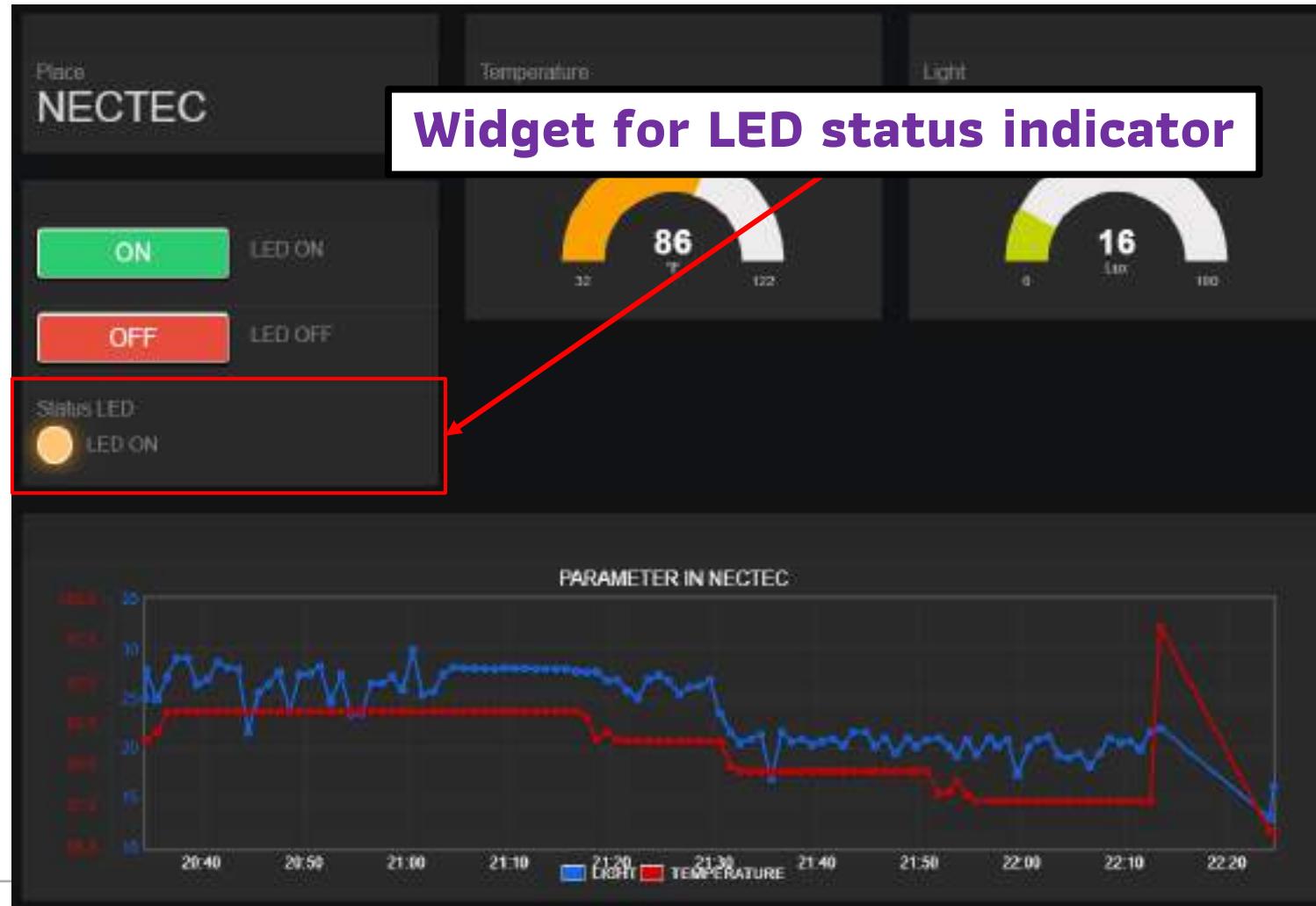
# 3 - Freeboard in NETPIE2020 (Control Widget)

## Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard



# 3 - Freeboard in NETPIE2020 (Control Widget)

Exercise 5 : Control LED on ESP32 or ESP8266 via Freeboard



4

# RESTful API

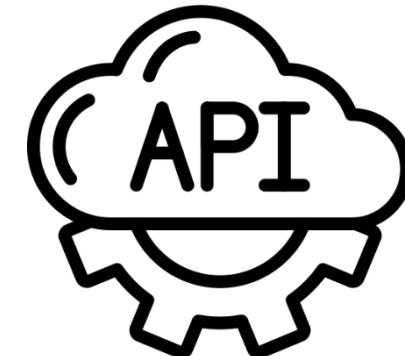
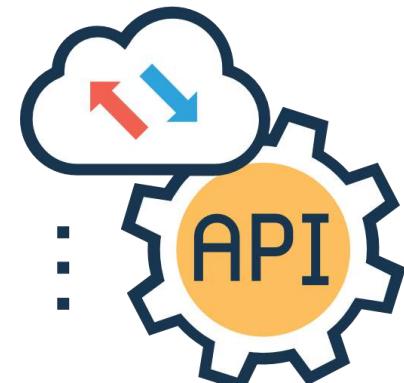
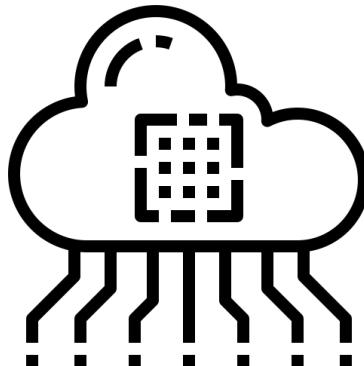


# 4 - RESTful API

## What is RESTful API?

**It is a channel for the device to call the Platform service via the RESTful API which uses HTTP protocol, suitable for integrating various systems, either existing or new development, without limitation by programming language or hardware. There are two groups of APIs:**

1. Device API
2. Data Store API



# 4 - RESTful API

## Device API

It is an API related to device management or calling. The EndPoint is <https://api.netpie.io/v2/device>

### Publish message to a topic

```
curl -X PUT "https://api.netpie.io/v2/device/message?topic=%40msg%2Ftest" -H  
"Authorization: Basic ClientID:Token" -H "Content-Type: text/plain" -d "message"
```

### Read Shadow Data of Device

```
curl -X GET "https://api.netpie.io/v2/device/shadow/data" -H "Authorization:  
Basic ClientID:Token"
```

### Write Shadow Data using Merge method

```
curl -X PUT "https://api.netpie.io/v2/device/shadow/data" -d "{data:{humid:63.7,  
temp:25.2}}" -H "Authorization: Device ClientID:Token"
```

# 4 - RESTful API

---

## Device API

### Write Shadow Data using Overwrite method

```
curl -X POST "https://api.netpie.io/v2/shadow/data" -d "{data:{temp:31.7}}" -H  
"Authorization: Device ClientID:Token"
```

### Write to Shadow with specified time in the past

```
curl -X PUT " https://api.netpie.io/v2/shadow/data" -d "{data:{humid:61.9,  
temp:28.6,timestamp:1566863843262}}" -H "Authorization: Device ClientID:Token"
```

# 4 - RESTful API

## Data Store API

It is an API that deals with retrieving data stored in Time series Database, where the Domain name of API is

<https://api.netpie.io/v2/feed> . The database is stored in KairosDB format, so querying the parameters can be done using the same format as KairosDB

### Read data from Timeseries Database of Device

```
curl -X POST "https://api.netpie.io/v2/feed/api/v1/datapoints/query" -H "Content-Type: application/json" -H "Authorization: Bearer userToken" -d ' {  
  "start_relative": { "value": 1, "unit": "days" },  
  "metrics": [ { "name": "ClientID", "tags": [ { "attr": "temperature" } ],  
    "limit": 50,  
    "group_by": [ { "name": "tag", "tags": [ "attr" ] } ],  
    "aggregators": [ { "name": "avg", "sampling": { "value": 1, "unit": "minutes" } } ] } }'
```

### Webpage to Generate Code for API calls

<https://trial-api.netpie.io/>

5

# Workshop



Drop your workshop at

Link --> <https://bit.ly/3m4isJj>

Set File Name is W follow by your IDStudent

Such as W3\_57364150, W4\_57364150

(W3 = Schema File, W4 = Freeboard File, W5 = Arduino File,  
W6 = Trigger File, W7 = Event Hook)





# **ITCS447**

## **Lectures 14-15**

**Our Lab: IoT System Integration  
IoT Security  
Battery Life & Power Management**

**Asst. Prof. Dr. Thitinan Tantidham**



มหาวิทยาลัยมหิดล  
Mahidol University

ห้ามมิให้นักศึกษาทำข้า ดัดแปลง หรือใช้ประโยชน์จากการอันมีลิขสิทธิ์ปราภูอยู่ในระบบการศึกษาอิเล็กทรอนิกส์ (E-Learning) ของมหาวิทยาลัย ไม่ว่าจะทั้งหมดหรือบางส่วน โดยไม่ได้รับอนุญาตจากมหาวิทยาลัย นอกจากนี้จากการศึกษาส่วนบุคคล ทั้งนี้ การทำข้า ดัดแปลง หรือเผยแพร่ต่อสาธารณะชื่องานอันมีลิขสิทธิ์ จะมีโทษปรับตั้งแต่ 20,000 บาท ถึง 200,000 บาท และหากเป็นการกระทำเพื่อการค้า จะมีโทษจำคุกตั้งแต่ 6 เดือน ถึง 10 ปี หรือปรับตั้งแต่ 100,000 บาท ถึง 800,000 บาท หรือทั้งจำทั้งปรับ

Unless allowed by Mahidol University, the User shall not copy, modify, or exploit in part or in whole of the copyrighted materials on the Platform, other than for your own individual study. The copy, modify or communication to public of the copyrighted materials shall be inflicted with a fine from 20,000 Baht up to 200,000 Baht. If the offence is committed with the commercial purpose, the offender shall be inflicted with imprisonment for a term from 6 months up to 10 years or a fine from 100,000 Baht up to 800,000 Baht or both imprisonment and fine.

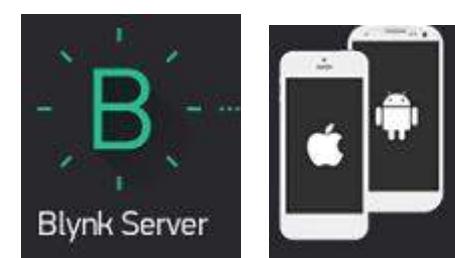
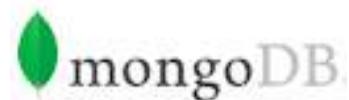
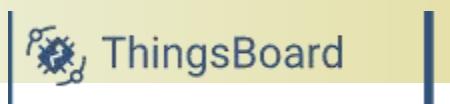




# Part 1

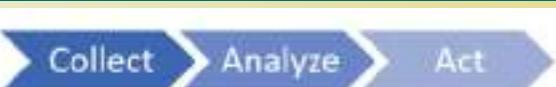
# IoT System Integration

# Our Lab: IoT System Integration

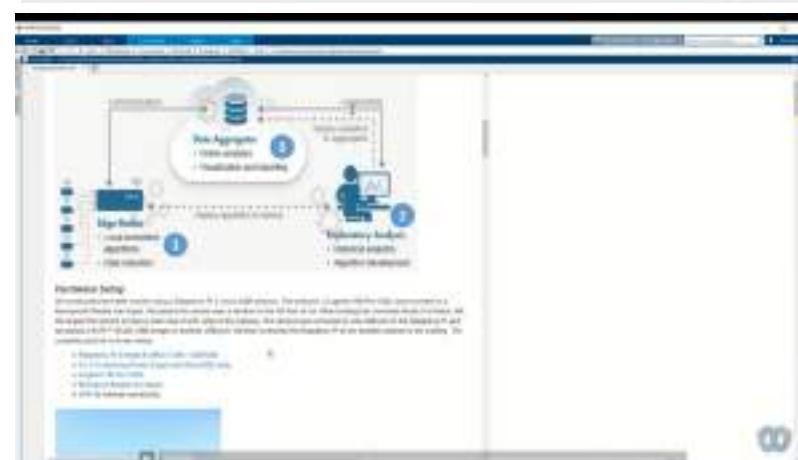


# Thingspeak

## IoT Workflow



- Acquire data from the field using sensors – often in remote locations!
- Monitor the data in near real time
- Analyses the raw data in the cloud to gain further insight
- React to what the data is telling you
  - Send commands back to the devices
    - e.g., turn on irrigation, turn off motor
  - Send an email or tweet
  - Trigger an HTTP request
    - e.g., to IFTTT or other web services



### How to use ThingSpeak?



### What is ThingSpeak?

- Online IoT analytics platform
  - Typically used to collect data from sensors ("Things")
  - Provides instant visualization of the data
  - Popular for people experimenting in IoT
  - Easy to get started
- Can be used to act on data
  - E.g. Turn a message when the temperature in your house goes above 32 degrees
- Can be used to analyse data
  - MATLAB Integration allows users to run native MATLAB code on data coming into ThingSpeak



Collect  
Send sensor data to the cloud

Analyze  
Analyze and visualize your data

Act  
Trigger an action

### Using ThingSpeak and MATLAB for IoT Key Takeaways

- ThingSpeak allows you to **collect, analyses and act** on your IoT data
  - Without setting up the web servers
  - Without having expertise in web technology
- ThingSpeak **natively run MATLAB** for more advanced analysis of your IoT data



### Building a Car Counter

#### Objectives

- Measure, explore, discover traffic patterns

#### Solution

- RaspberryPi 2+ webcam
- Automated deployment of vision algorithms on embedded sensor





# Part 2

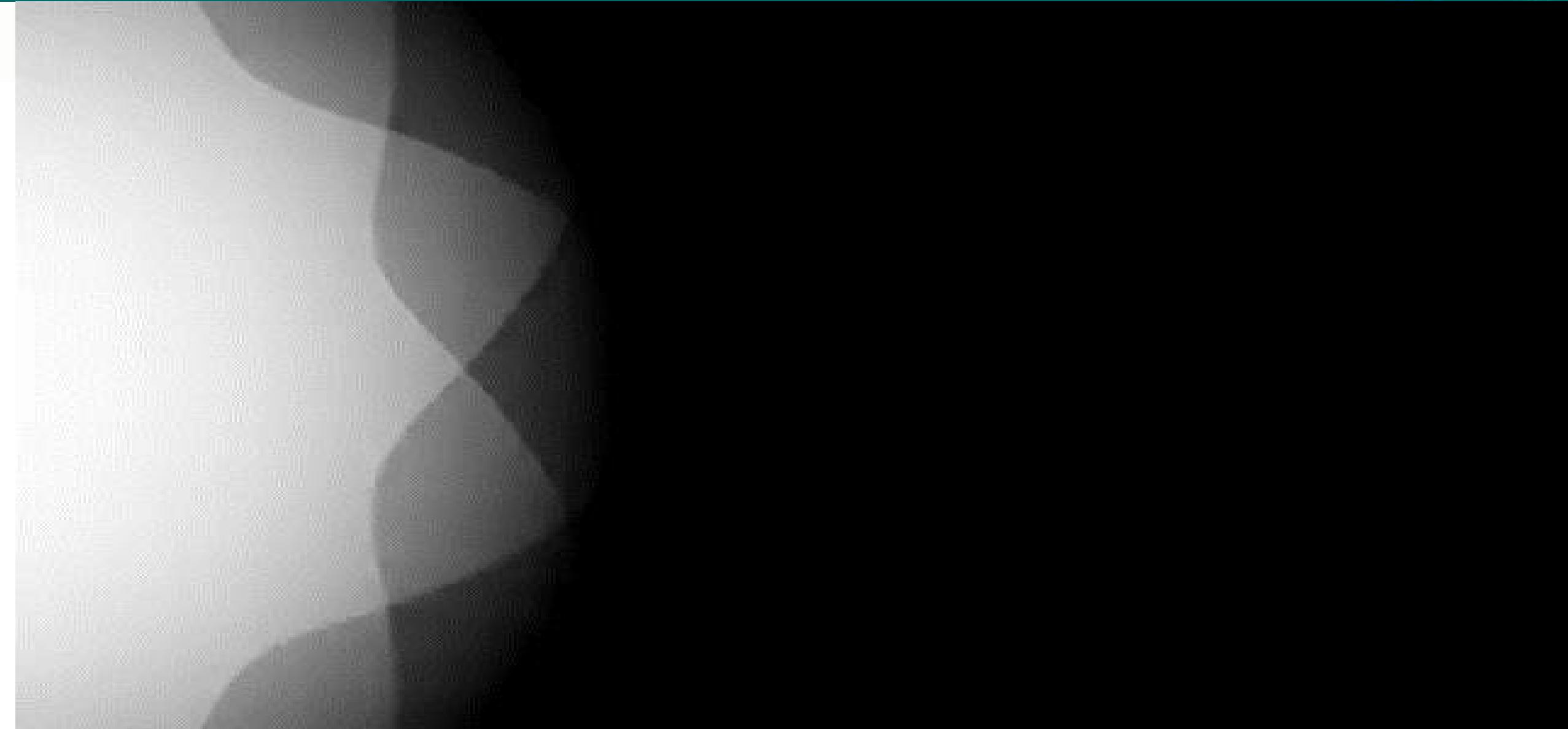
# IoT Security



- IoT Security – Related Terminologies
- IoT Characteristics
- Data in IoT
- Data Flow in IoT
- Case Studies
- News
- IoT Attacks
- IoT Reference Model
- IoT Vulnerabilities and attacks
- Edge Computing Security Strategies



[https://www.sei.cmu.edu/our-work/projects/display.cfm?customel\\_datapageid\\_4050=334957](https://www.sei.cmu.edu/our-work/projects/display.cfm?customel_datapageid_4050=334957)  
<https://www.iotworldtoday.com/guide/striking-back-an-iot-security-guide-for-critical-infrastructure/>  
NETACAD Fundamental IoT: IoT Security





## IoT Security vs Cybersecurity vs CPS

- IoT Security is not traditional cybersecurity, but a fusion of cybersecurity with other engineering disciplines.
  - It addresses much more than mere data, servers, network infrastructure, and information security. Rather, it includes the direct or distributed monitoring and/or control of the state of physical systems connected over the Internet.
  - It requires a unique application for each system and system-of-systems in which IoT devices participate.
- Cybersecurity, if you like that term at all, generally does not address the physical and security aspects of the hardware device or the physical world interactions it can have.
- Cyber-Physical Systems (CPS) are a huge. A CPS, comprising connected sensors, actuators, and monitoring/control systems, does not necessarily have to be connected to the Internet and still achieves its business objective.



## Vulnerability - Threat - Attack – Adversary - Countermeasure

- **Vulnerability:** A **flaw or weakness** in a system's design, implementation, operation, or management that could be exploited to violate the system's confidentiality, integrity, or availability
- **Threat:** Any circumstance or event with the potential to exploit a vulnerability and adversely affect a system through **unauthorized access, destruction, disclosure, or modification of data, denial-of-service**, etc.
- **Attack:** An intentional assault on system security that derives from an intelligent threat.
  - **Active attacks** attempt to **alter system resources or affect their operation**
  - **Passive attacks** attempt to learn or make **use of information** from a system but does not affect that system.
- **Adversary:** An entity that **attacks** a system or is a **threat** to a system.  
synonyms: intruder, attacker, cyber attacker, cracker, hacker, etc.
- **Countermeasure:** An action, device, procedure, or technique that meets or opposes (i.e., counters) a threat, a vulnerability, or an attack by eliminating or preventing it, by minimizing the harm it can cause, or by discovering and reporting it so that corrective action can be taken.

- **Heterogeneity of devices** – the large number of devices used means high diversity in their calculation and communication capabilities;
- **Scalability** – addressing, naming, connectivity of a growing number of devices being used every day;
- **Wide scale use of wireless data transfer technology** – problems connected with overcoming this issue are related to transfer speeds and delays in delivery of data;
- **Optimum energy use** – the issue of power use is crucial;
- **User configuration capabilities** – the number as well as the complexity of systems make it necessary to provide mechanisms allowing the users to configure the systems themselves;
- **Data management** – in IoT it will be crucial to utilize appropriate data models and semantic descriptions of their content, appropriate language and format;
- **Privacy protection** – because of its close relationship with the real world, IoT technologies must ensure an appropriate level of security and privacy.

Ref: A. Magruk, "The Most Important Aspects of Uncertainty in the Internet of Things Field – Context of Smart Buildings", Procedia Engineering, Elsevier, 2015.



## 1: Data Ingestion: IoT devices/sensors collect data from the environment.

The data can be as simple as temperature/humidity or it can be as complex as a full video feed. "Almost 5 quintillion bytes of data produced every day by IoT devices."

**The data needs to be sent to cloud to be analyzed. But it needs a way to get there.**

## 2: Data Transmission: The data is transmitted to the cloud via Gateways (Telemetry Devices).

The gateways use both the cellular as well as satellite communication to transmit the data.

To ensure the data security protocols such as Bluetooth, Sig Fox, LoRa, NB-IoT, ZigBee, COAP, REST, MQTT, etc are used.

## 3: Data Processing: Once the data gets to the cloud, IoT platform process it.

The processing can be as simple as checking if the temperature is within the acceptable range or it could be very complex, such as using computer vision on video to identify objects.

## 4: Data Visualization: The processed data (information) is made useful to the end-user by providing alerts to the user (E-mails, text, notification).

The user might have an application (interface) that allows him to proactively check-in to the system.

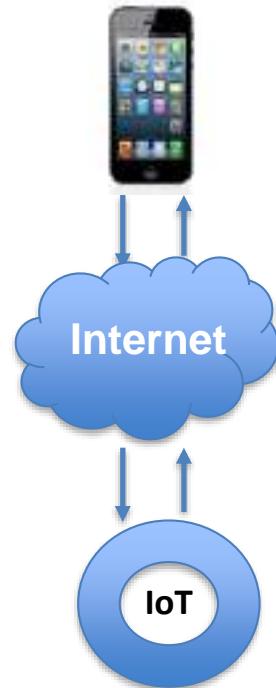
## 5: Data Analysis and Prediction: To utilize the data collected over the time, data analytics makes use of the historical data to provide actionable insights. Insights helps in predicting the future events that may occur. For example, by analyzing the data, we can predict the possible future malfunctioning of a machinery.

# Data Flow in IoT - Operational Models

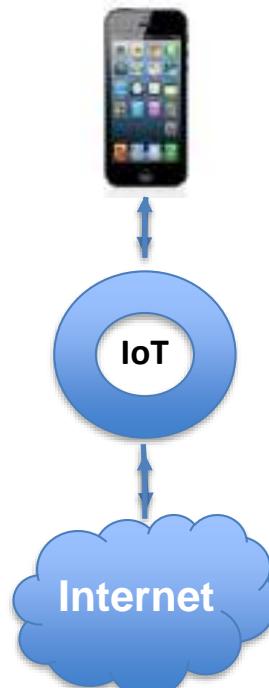


## Security and Privacy Risks

External Server



Direct Access



Transit



Eg: Nest Protect Smoke Alarm

<https://www.youtube.com/watch?v=81jg6SlxnXM>

Eg: Philips Hue Lamps/WeMo

Eg: Fitbit Flex



External Access: the user has no direct interaction with the IoT device. IoT device communicates directly with the external server and the only way for the user to retrieve relevant data, such as current status via this external server.

Direct Access: The IoT device updates the server of the current status. It is also possible to control the IoT device via the web portal provided by the manufacturer.

Transit Access: these devices (e.g. Fitbit) do not have Wi-Fi interface, the user's phone is used as a relay/bridge for data exchange.

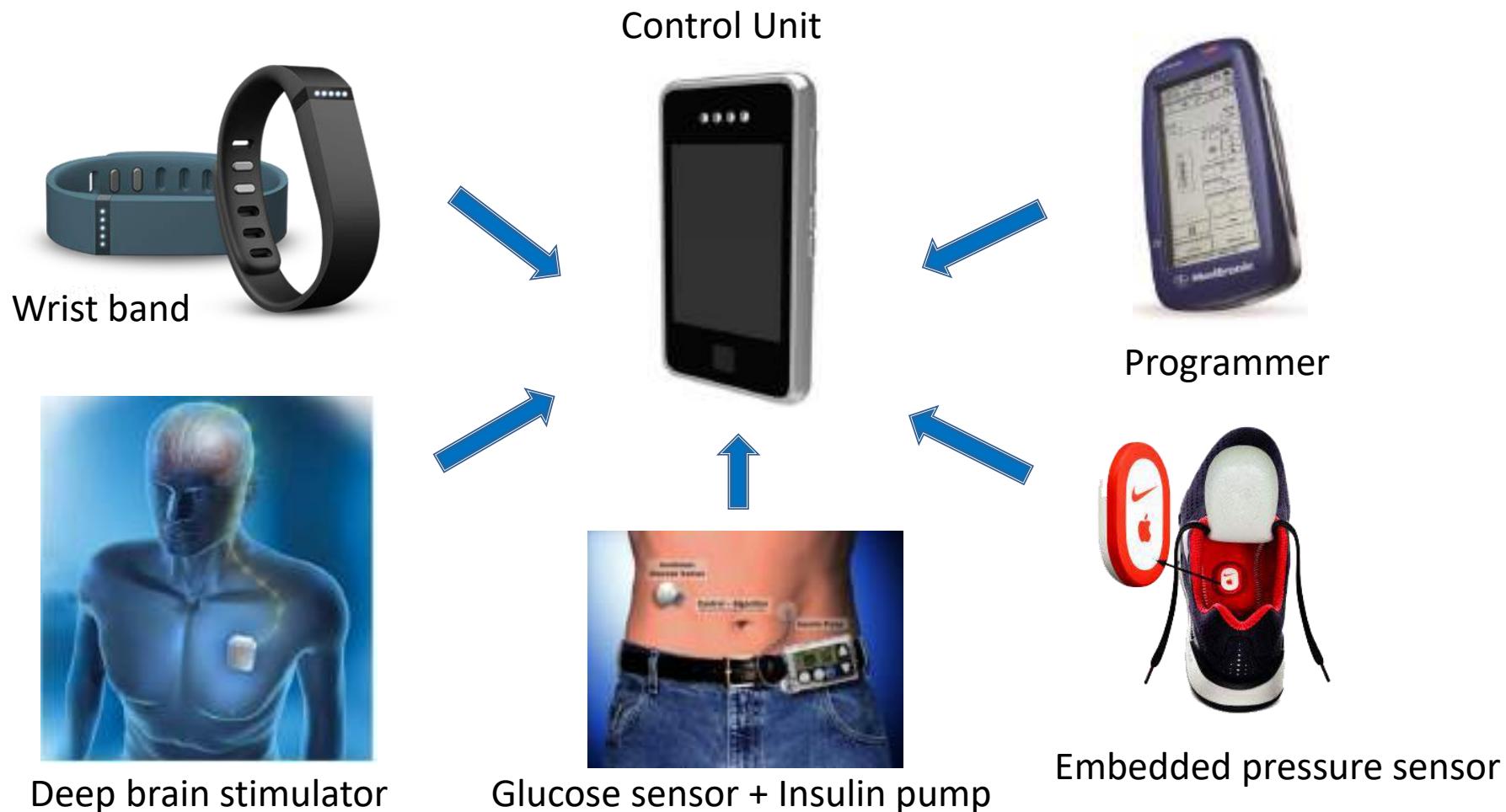
# Case Studies: Phillips Hue Lamps



- One of the oldest IoT devices on the market (since 2011).
- Ability to control lights via a smartphone app.
- Highly Customizable and work with a lot of **3<sup>rd</sup> party services** like IFTTT (eg: blink the light if someone sends me a message on facebook)
- Phone talks directly to the hue bridge and bridge then relays appropriate commands to the lights using zigbee.  
All Communications between the phone and the bridge are **in plain text**.



# Case Studies: E-Health Applications



# News in the IoT Security Fails



- “Lack of Trust in Internet Privacy and Security May Deter Economic and Other Online Activities ”, NTIA May 2016.
- How long will consumers put up with the IoT's failures? – IoT support panel, CES 2016.
- IoT “plug and pray” all over again, says security consultant David Alexander, PA Consulting, CRESTcon & IISP 2016.
- Three quarters of UK’s information security professionals think IoT device manufacturers aren’t implementing enough security on their products and 73% said there’s a general lack of industry standards – ISACA 2015 poll
- 72% of Americans see cyberattacks as a major threat, coming 2<sup>nd</sup> after ISIS – Pew Research poll, April 2016.
- “All of the potential weaknesses that could afflict IoT systems, such as authentication and traffic encryption, are already well known to the security industry... ”, Insecurity in the Internet of Things, Symantec, March 2015

<https://cismag.eccouncil.org/10-iot-security-incidents-that-make-you-feel-less-secure/>

<https://portswigger.net/daily-swig/iot>



## The IoT Risk and Security Awareness

- Target's Heating and Cooling System
  - Hackers gained access through HVAC account, and were able to install card skimming s/w on POS terminals
- Wink's IoT Hubs (<http://www.wink.com/>)
  - Consumers found their devices bricked when the Hub security certificate unexpectedly expired
- Insteon connected homes
  - Reporter able to turn lights on and off whilst chatting with home owners over the phone
- Home routers
  - Open to man in the middle attacks when people use default or easy to guess passwords
- Spammy refrigerators
  - Default passwords allowed attacker to use connected refrigerators as part of a `bot net
- TrendNet's nanny cams
  - Easy remote access once you have the camera's IP address
- Samsung's smart TVs
  - Easy to commandeer to view people's living rooms
- Nest thermostat
  - Easy to hack if you can get physical access for a few minutes

<https://www.w3.org/Talks/2016/0614-iot-security.pdf>

<https://www.forbes.com/sites/kashmirhill/2013/07/26/smart-homes-hack/?sh=7dedc710e426>



## The IoT Risk and Security Awareness



### Vehicle Hacking

<http://bit.ly/1s0m4Hv>  
<http://bit.ly/1TOt2h5>



### Global Positioning System Spoofing

[https://www.youtube.com/watch?v=y4pr5\\_ea5hw](https://www.youtube.com/watch?v=y4pr5_ea5hw)



### Healthcare Device & Information Hacking

<http://bit.ly/1EJnTjv>



### Industrial Hacking



### Smart Home Hacking

<https://www.youtube.com/watch?v=-JVrFI138kE>



### National Transportation Safety Board (NTSB) Connected-Car Mandate

[https://ecfsapi.fcc.gov/file/103101033822776/Final\\_Security%20Considerations%20for%20Connected%20Vehicles%20and%20DSRC\\_3\\_19\\_2017.pdf](https://ecfsapi.fcc.gov/file/103101033822776/Final_Security%20Considerations%20for%20Connected%20Vehicles%20and%20DSRC_3_19_2017.pdf)

<https://www.youtube.com/watch?v=iRQPfISsG9k>  
<https://www.youtube.com/watch?v=Fki7MCRWgdo>

- Breaches of privacy
- Cybercrime
- Physical safety in the home, across the city and within businesses
- Threats to national infrastructure
- Looming risks of cyberwar

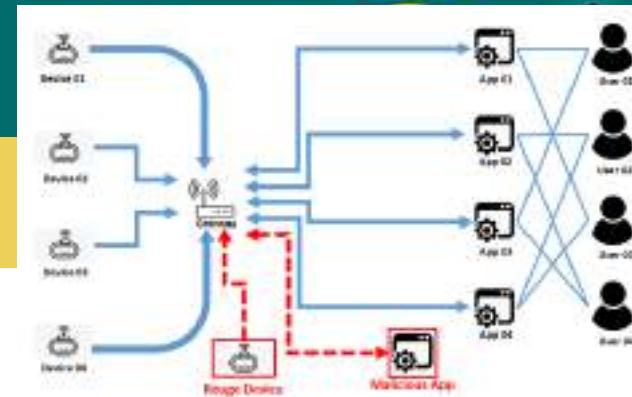
- Default, weak, and hardcoded credentials
- Difficult to update firmware and OS
- Lack of vendor support for repairing vulnerabilities
- Vulnerable web interfaces (SQL injection, XSS)
- Coding errors (buffer overflow)
- Clear text protocols and unnecessary open ports
- DoS / DDoS
- Physical theft and tampering

## Types of Attacks

- **Wireless Reconnaissance and Mapping**
  - Similar to the war dialing days of old where hackers scanned through telephone switching networks to identify electronic modems
  - Network scanning tool such as [Nmap](#) is commonly utilized by hackers to gather intelligence about hosts, subnets, ports, and protocols in networks, similar paradigms are being used against IoT devices
- **Security Protocol Attacks**
  - Many security protocols can sustain attacks against vulnerabilities introduced either in the protocol design (specification), implementation and even configuration stages (in which different, viable protocol options are set).
  - As an example, researchers found while testing a ZigBee-based consumer IoT implementation that the protocol was designed for easy setup and usage but lacked configuration possibilities for security and performed vulnerable device pairing procedures.
- **Physical Security Attacks**
  - Physical security attacks include those in which the attacker(s) physically penetrate the enclosure of a host, embedded device, or other type of IoT computing platform to gain access to its processor, memory devices, and other sensitive components.
- **Application Security Attacks**
  - IoT devices and connections can be exploited through attacks against application endpoints. Application endpoints include web servers as well as mobile device applications (for example, iPhone, Android) that have a role in controlling the device.

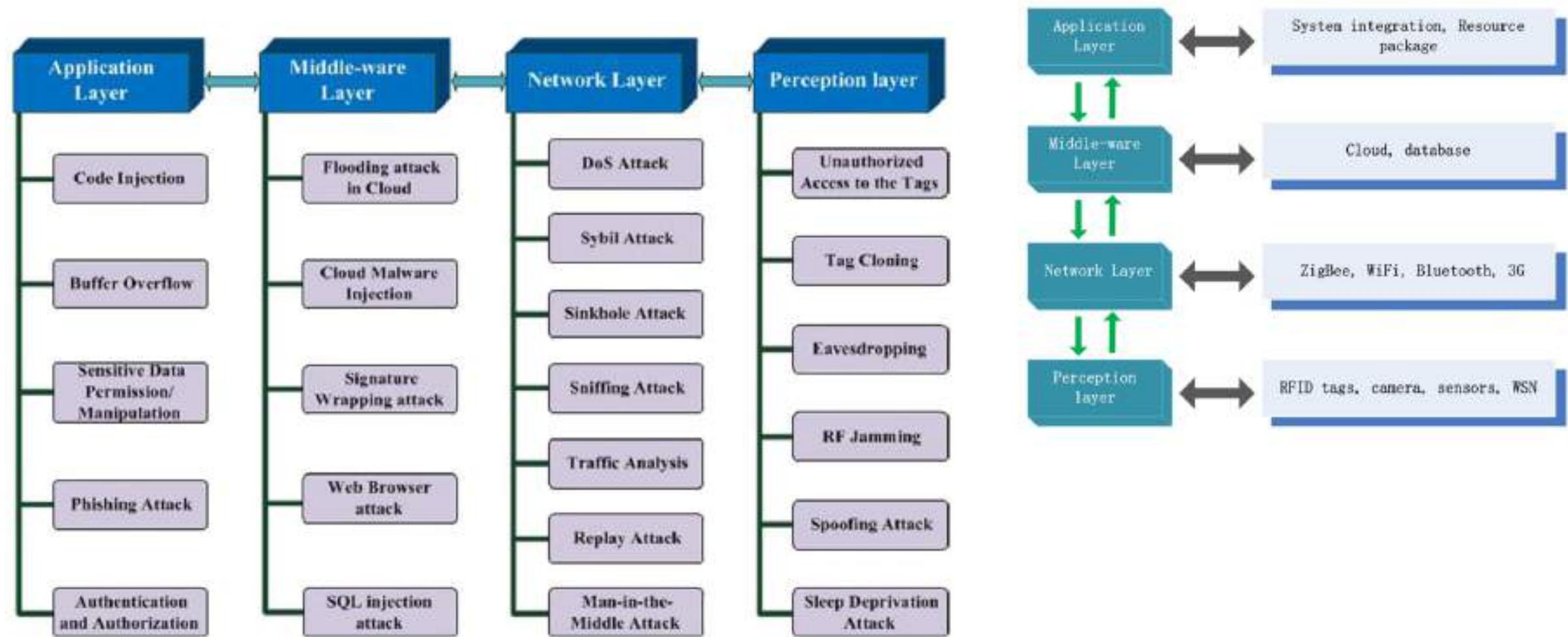
# Common IoT Attacks

- Lack of compliance on the part of IoT manufacturers (e.g.: fitness trackers, smart refrigerators, etc.)
- Lack of user knowledge and awareness
- IoT security problems in device update management
- Lack of physical hardening
- Botnet attacks: IoT devices are highly vulnerable to malware attacks. A victim's system can be remotely take control and confidential data can be storlen.
- Industrial espionage and eavesdropping: Attackers can sniff the traffic generated by IoT data flow to gather users' critical information (usernames/passwords, identifiers, other useful data) by setting similar IoT devices.
- Highjacking IoT devices (ransomware does not destroy sensitive files but blocks access to them by way of encryption)
- Data integrity risks of IoT security in healthcare
- Rogue IoT devices (without authorization)
- Cryptomining with IoT Bots (Mining cryptocurrency demands colossal CPU and GPU resources)
- Malicious Data Injection: False sensor data injection is a form of attack in which the sensor data used in IoT applications are forged or modified for malicious purposes.
- Sybil Attack: The malicious nodes in this can have multiple identities of a genuine node by either impersonating it or with a fake identity through duplication. One such malicious node may have several identities simultaneously or at different instances or an attacker can be in more than one place at once.
- Disclosure of Critical Information: Sensors used in IoT gadgets can disclose sensitive information such as passwords, secret keys, credit card credentials, and so on.
- Sinkhole attack: all traffic is lured from an area through a compromised node, where selective forwarding can follow with the attacker deciding what data to allow through.
- Wormhole attack – an attacker tunnels messages received in one part of the network over a low latency link and replays them in a different part.
- HELLO flood – here the attacker causes every node to mark it as their parent. Most nodes will be out of range and this causes a lot of packets to be lost. Routing loops can be set up via spoofing routing updates, with two nodes being attacked and redirecting packets to each other.
- Acknowledgement spoofing - used for a selective for-warding attack, where an attacker strengthens/weakens networks links so packets are lost from a node.
- Denial of Service (DoS) attacks – jamming radio signals, using malicious nodes to refuse to route messages or redirect them to unwanted locations.
- MITM(Man-in-the-Middle): a system that listens on traffic between smart device and a gateway.
- Ransomware: a hacker uses malware to encrypt data that may be required for business operations. An attacker will decrypt critical data only after receiving a ransom.



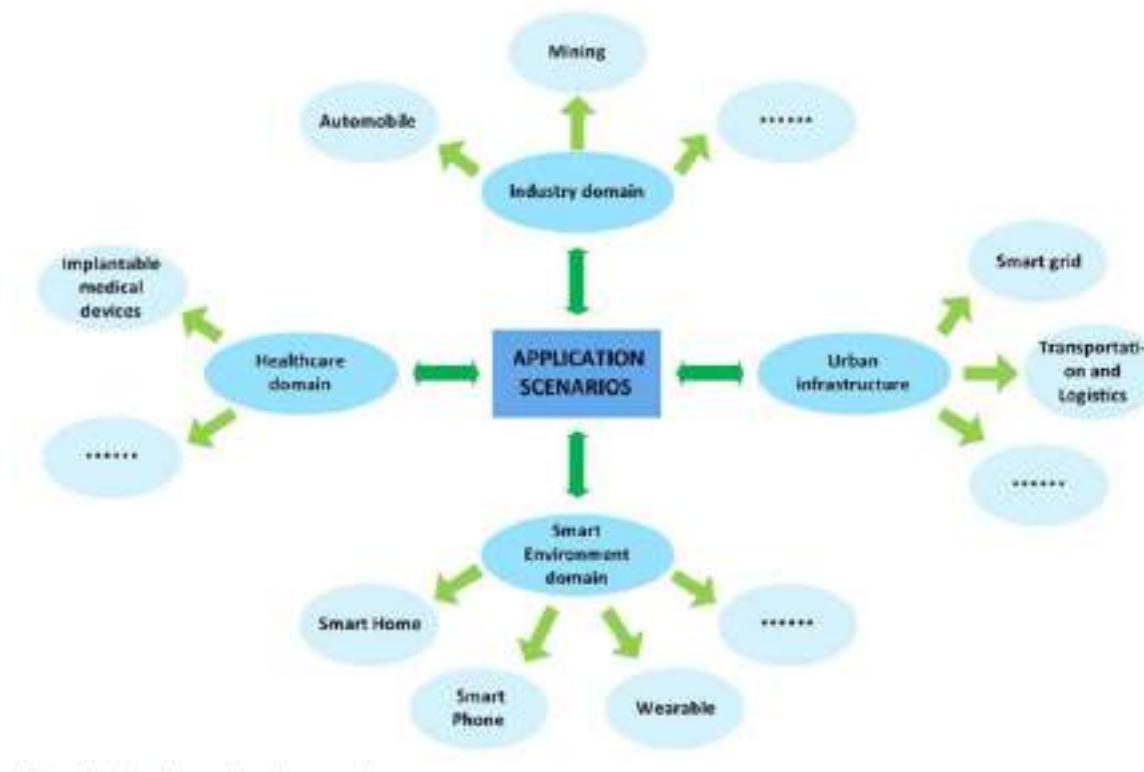
Ref: <https://www.intellectsoft.net/blog/biggest-iot-security-issues/>

## Types of Attacks based Architecture

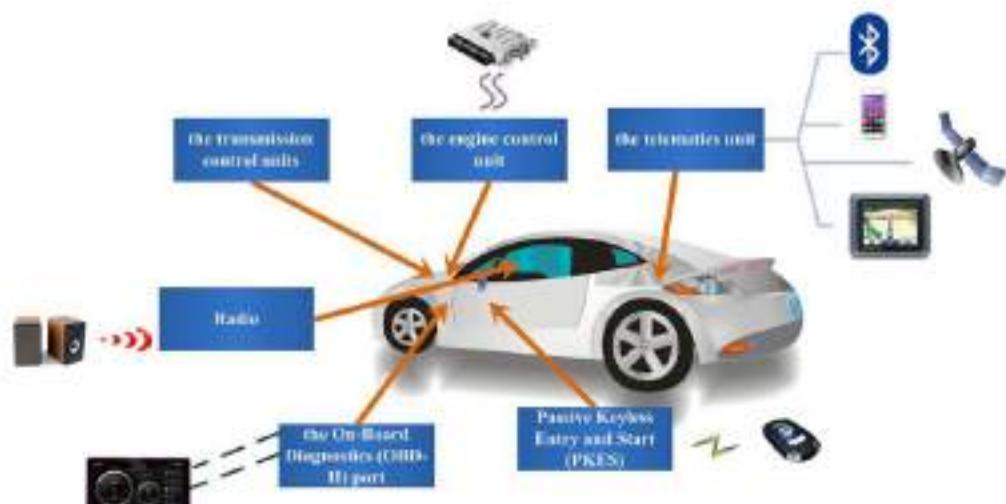




## Types of Attacks based Application Scenarios



Reverse engineering can also be used to "crack" software and media to remove their copy protection, or to create a (possibly improved) copy or even a knockoff; this is usually the goal of a competitor

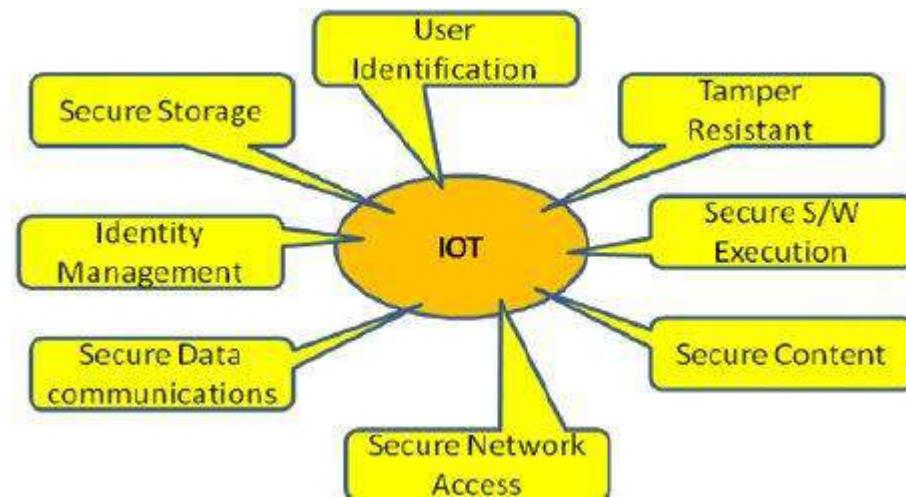
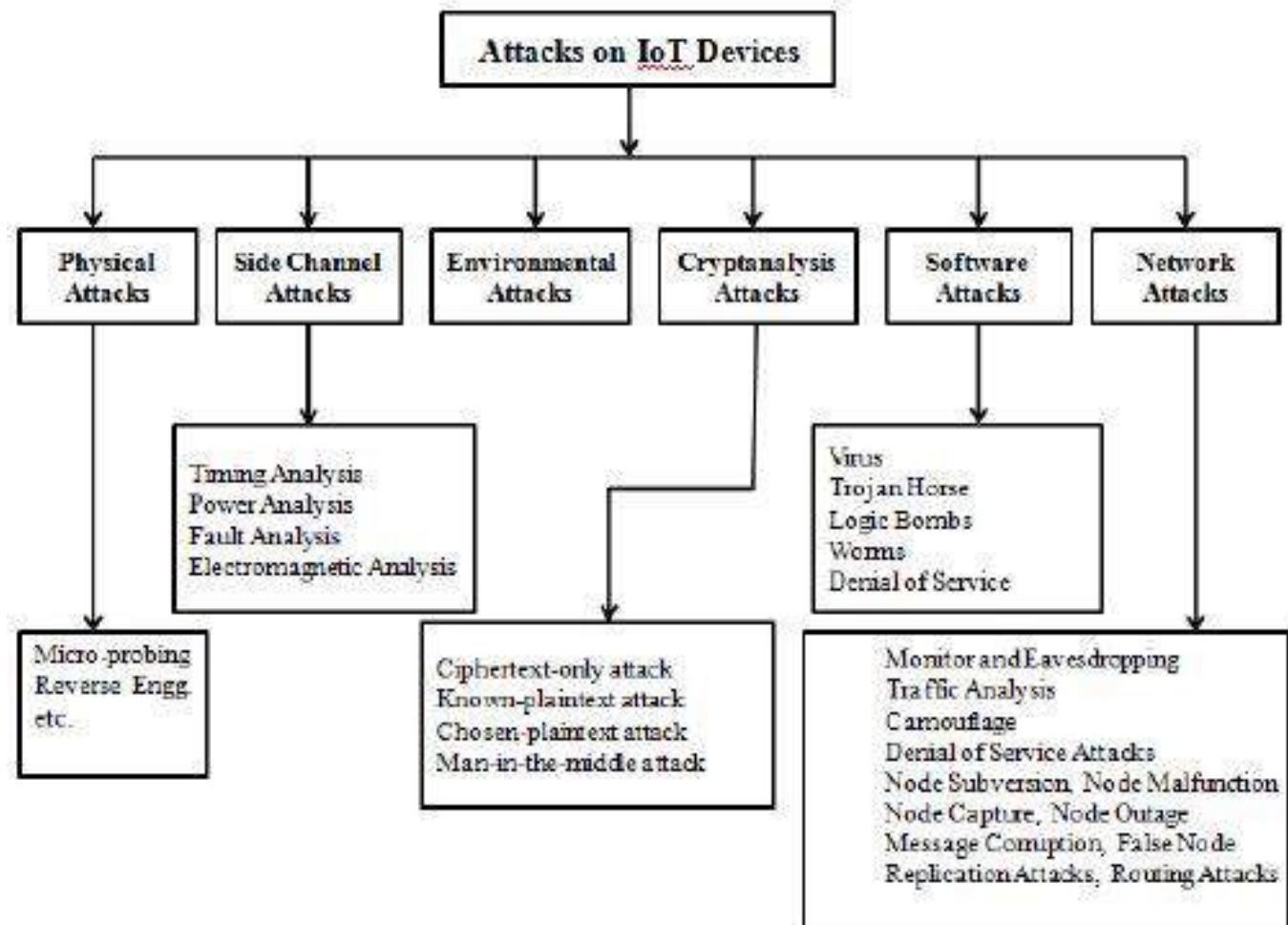




## Types of Attacks based on Sensor Vulnerabilities

| Sensor Type           | Sensor                     | Vulnerabilities      |
|-----------------------|----------------------------|----------------------|
| Motion Sensors        | Accelerometer              | Task Inference       |
|                       | Gyroscope                  | False Data Injection |
|                       | Linear Acceleration Sensor | Malware Transmission |
| Environmental Sensors | Light Sensor               | Eavesdropping        |
|                       | Proximity Sensor           | Task Inference       |
|                       | Air Pressure Sensor        | Smudge Attack        |
|                       | Audio Sensor               | False Data Injection |
|                       | Temperature Sensor         | Transferring Malware |
|                       | Soil Moisture sensor       | DoS                  |
|                       | Noise Sensor               | Information Leakage  |
| Position Sensors      | GPS                        | Location Inference   |
|                       | Magnetic Sensor            | Eavesdropping        |
|                       |                            | False Data Injection |

## Attack on IoT Devices



# IoT Attack based on IoT Reference Model



There is no single standard and unified IoT layer model.

Levels:

Identify management software

7 Collaboration & Processes  
(Involving People & Business Processes)

Authentication/authorization software

6 Application  
(Reporting, Analytics, Control)

Secure storage

5 Data Abstraction  
(Annotation & Access)

Tamper resistant software

4 Data Accumulation  
(Storage)

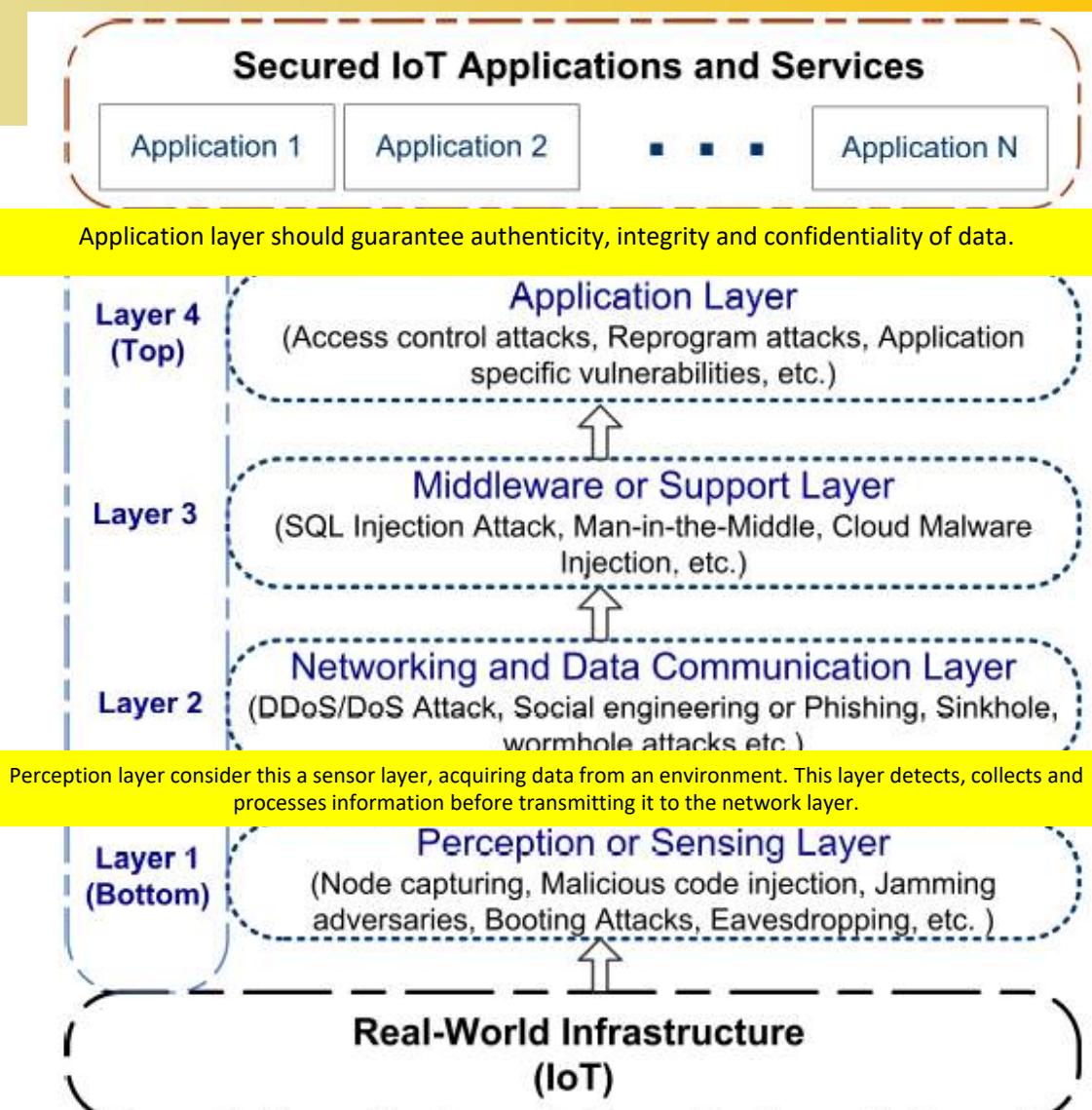
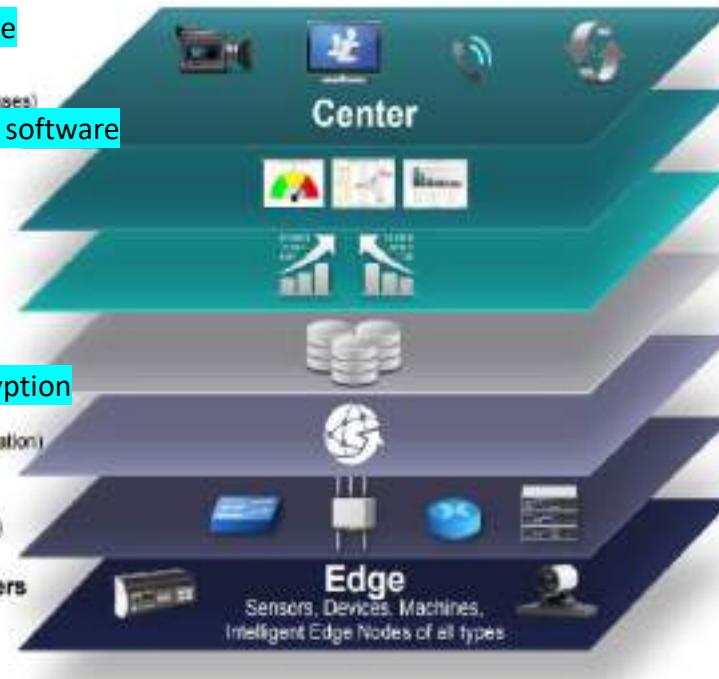
Secure communication/encryption

3 Edge (Fog) Computing  
(Data Element Analysis & Transformation)

Secure network access

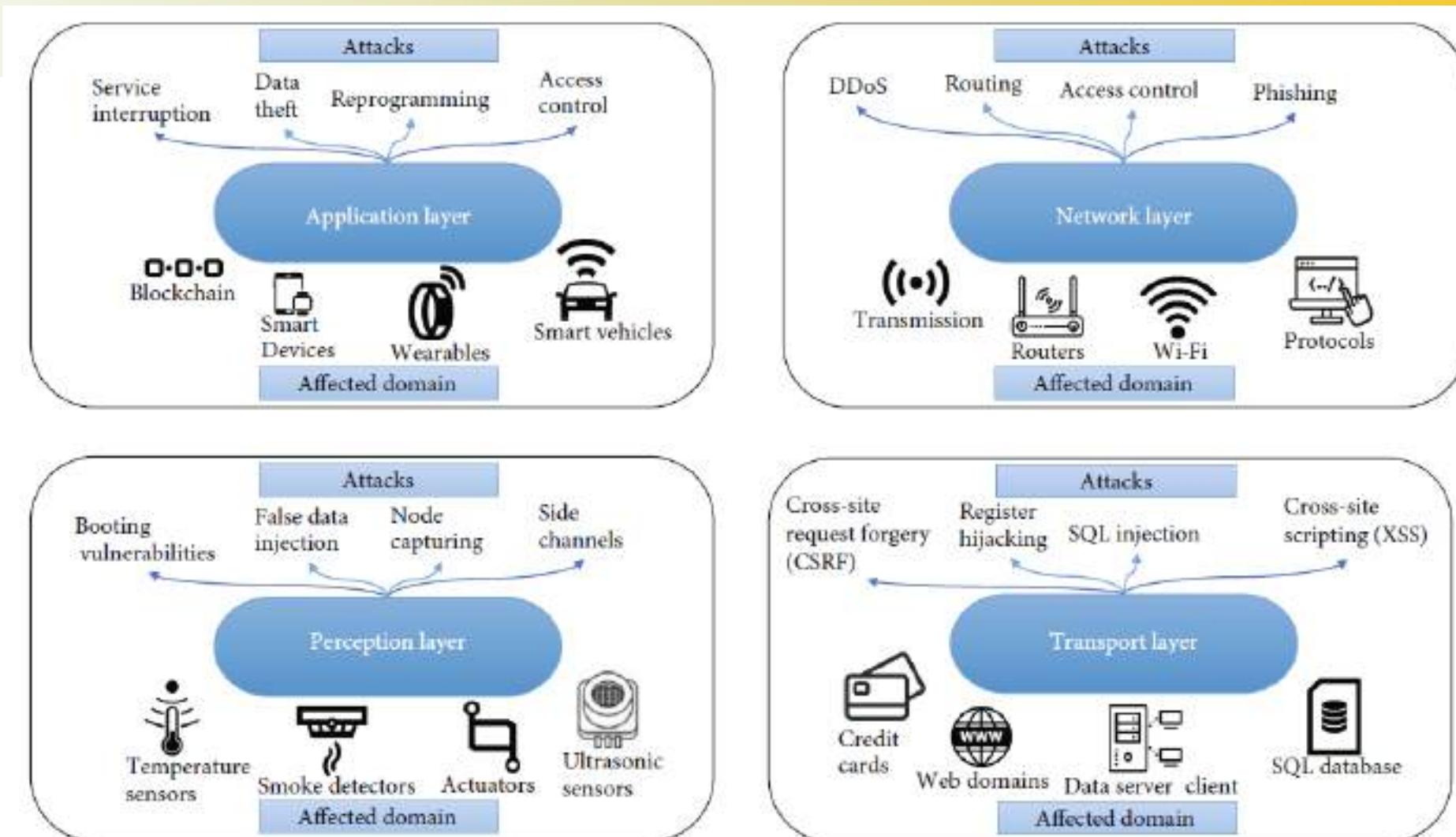
2 Connectivity  
(Communication & Processing Units)

1 Physical Devices & Controllers  
(The 'Things' in IoT)



S. Millar, "IoT Security Challenges and Mitigations: An Introduction", 2021.

# IoT Attack based on IoT Reference Model



R., S. Kataria, et.al, "Threats and Corrective Measures for IoT Security with Observance of Cybercrime: A Survey", 2021.

- **Perception Layer:** Eavesdropping, Battery drainage, Hardware failure, Malicious data injection, Sybil threat, Disclosure of critical information, Device comprise, Node cloning
- **Abstraction Layer:** Node replication, Illegal access, Device compromise, MITM, Eavesdropping, Spoofing, Insertion of rogue devices, Information theft, Threats to communication protocols
- **Network Layer:** Illegal access, MITM, Eavesdropping, Spoofing, Fragmentation threat, Hello flood, Network intrusion, Device compromise, Node replication, Insertion of rogue devices
- **Transport Layer:** Jamming, Eavesdropping, False data injection, Unfair access, Congestion, Hello flood
- **Computing Layer:** Malicious attack, SQL injection, Data integrity, Virtualization, Software modification, Illegal access, Identity theft
- **Operation Layer:** Fake information, Badmouthing, Unauthorized access, User privacy compromise, Stealing user critical information
- **Application Layer:** Malicious code, Software modification, Data tampering, SQL injection, Disclosure of critical information, Cross site script, Identity theft

- **Perception Layer:** Node capture attack, Side channel attack, Tag cloning, RF jamming, node injection attack, Node tampering, Physical damage, Exhaustion attack, Node outage, Battery drainage attack
- **Abstraction Layer:** Data manipulation attack, Device tampering attack, Tag cloning, MITM attack, Dos attack, DDoS attack, Side channel attack, Traffic analysis, Sleep deprivation attack
- **Network Layer:** Eavesdropping attack, Hello flood attack, Sinkhole attack, Sybil attack, Clone ID attack, Selective forwarding attack, Black hole attack, Wormhole attack, Traffic attack, RPL exploit
- **Transport Layer:** DoS attack, DDoS attack, Side channel attack, Desynchronization, MQTT exploit, Session hijacking, SYN-flooding attack, Timing attack, Unfairness attack
- **Computing Layer:** Flooding attack in cloud/edge, Malware injection attack, Access attack, False data injection, Path based DoS attack, Hole attack, Exhaustion attack, Edge/Cloud outage, Signature wrapping, Storage attack
- **Operation Layer:** Man-in-the-middle attack, Secure on-boarding attack, Firmware attack, Software attack, Illegal intervention, End-to-end encryption attack, Interrogation attack, DoS attack
- **Application Layer:** Virus attack, Malware attack, Spyware attack, Flooding, Spoofing, Code injection, Intersection, Message forging DDoS attack, Brute force attack.

# IoT Device/System/Software Attack Tools



- Kali Linux
- Tools (all available in Kali distro):
  - wireshark: [www.wireshark.org](http://www.wireshark.org)
  - nmap / zenmap: <https://nmap.org/>, <https://nmap.org/zenmap/>
- aircrack-ng / fern
  - <https://www.aircrack-ng.org/>
  - <https://tuxdiary.com/2015/08/30/fern-wifi-cracker/>
- metasploit / armitage
  - <https://www.metasploit.com/>
  - <http://www.fastandeasyhacking.com/>
- OWASP Zed
  - [https://www.owasp.org/index.php/OWASP\\_Zed\\_Attack\\_Proxy\\_Project](https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project)
- Nikto
  - <http://sectools.org/tool/nikto/>
- Sqlmap
  - <http://sqlmap.org/>
- Social Engineer Toolkit
  - <https://www.trustedsec.com/social-engineer-toolkit/>
- Maltego
  - <https://www.paterva.com/web7/>
- Shodan (the search engine website scans the Internet for publicly accessible devices.)
  - shodan.io

Red teams are offensive security professionals who are experts in attacking systems and breaking into defenses.

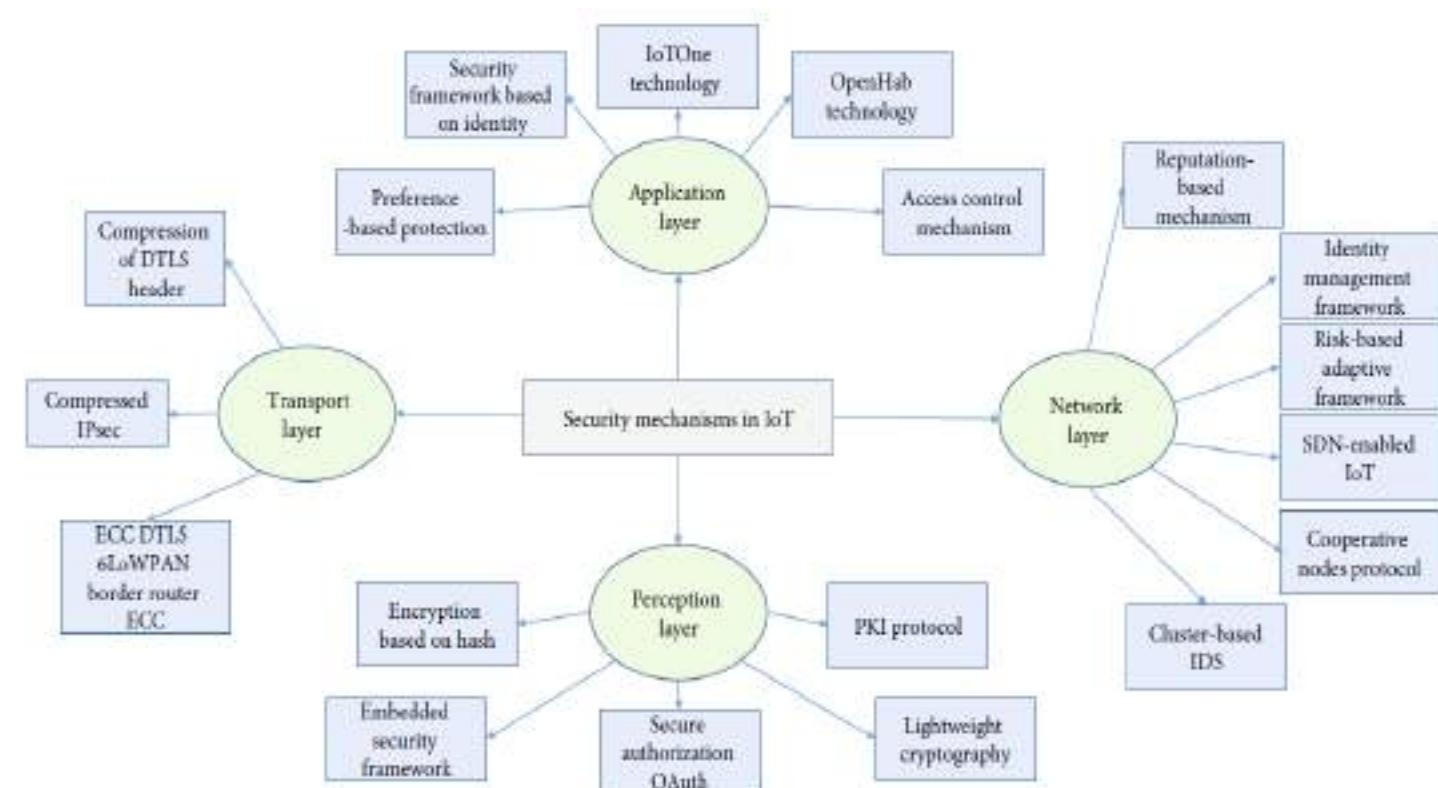
Blue teams are defensive security professionals responsible for maintaining internal network defenses against all cyber attacks and threats

# IoT Device Attack Tools



| Software                                   | OS/Support                                      | Features                                                                                                                                            | Sources                                                                                                                     |
|--------------------------------------------|-------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| E3 Universal                               | Window, Linux, macOS, iOS                       | IoT analysis, cloud data imaging, and analysis, registry analysis, email investigation, JTAG, and chip dump processing                              | <a href="https://paraben.com/digital-forensic-tools-6/">https://paraben.com/digital-forensic-tools-6/</a>                   |
| WireShark                                  | Windows, Linux, macOS, Solaris                  | VoIP, GUI, offline analysis, WAN/LAN analyzer                                                                                                       | <a href="https://www.wireshark.org/">https://www.wireshark.org/</a>                                                         |
| Autopsy                                    | Windows, Linux, macOS Android                   | Registry analysis, LNK file analysis, timeline analysis, file type detection, email analysis                                                        | <a href="https://www.sleuthkit.org/autopsy/">https://www.sleuthkit.org/autopsy/</a>                                         |
| Paladin                                    | Linux                                           | Device cloning support for many forensic image formats: E01, Ex01, RAW, VHD, AFF, disk manager, and automatic logging                               | <a href="https://sumuri.com/software/paladin/">https://sumuri.com/software/paladin/</a>                                     |
| Dumpzilla                                  | Unix, Windows                                   | Forensic information extraction from Firefox, SeaMonkey browsers including cookies, bookmarks, web forms, SSL certificates, browser-saved passwords | <a href="https://tools.kali.org/forensics/dumpzilla">https://tools.kali.org/forensics/dumpzilla</a>                         |
| SIFT (SANS investigative forensic toolkit) | Linux                                           | File system support, different evidence image format support, rapid scripting, and analysis                                                         | <a href="https://digital-forensics.sans.org/community/downloads">https://digital-forensics.sans.org/community/downloads</a> |
| Toolsley                                   | Web based                                       | File repairing, text encoding, file identification, file signature verification, binary inspection, CRC tool                                        | <a href="https://www.toolsley.com/">https://www.toolsley.com/</a>                                                           |
| NetworkMiner                               | Windows, Linux, macOS X, FreeBSD                | Live sniffing, OS fingerprinting, Geo IP localization, DNS whitelisting, audio extraction and playback of VoIP calls, PCAP and PcapNG file parsing  | <a href="https://sectools.org/tool/networkminer/">https://sectools.org/tool/networkminer/</a>                               |
| Elcomsoft                                  | Windows, macOS, iOS                             | Password recovery, cloud explorer, disk decryption, wireless security auditor                                                                       | <a href="https://www.elcomsoft.co.uk/">https://www.elcomsoft.co.uk/</a>                                                     |
| Belkasoft X                                | Windows, macOS, Linux, iOS, Android, Blackberry | E01/DD imaging, Hash set analysis, registry viewer, plist viewer, artifacts viewer, SQLite viewer                                                   | <a href="https://belkasoft.com/">https://belkasoft.com/</a>                                                                 |

# Existing Security Mechanisms



| Network layer       | Attacks           | Defenses                                                                          |
|---------------------|-------------------|-----------------------------------------------------------------------------------|
| Physical            | Jamming           | Spread-spectrum, priority messages, lower duty cycle, region mapping, mode change |
|                     | Tampering         | Tampering-proofing, hiding                                                        |
| Link                | Collision         | Error-correcting code                                                             |
|                     | Exhaustion        | Rate limitation                                                                   |
| Network and routing | Unfairness        | Small frames                                                                      |
|                     | Neglect and greed | Redundancy, probing                                                               |
| Transport           | Homing            | Encryption                                                                        |
|                     | Misdirection      | Egress filtering, authorization, monitoring                                       |
| Network and routing | Black holes       | Authorization, monitoring, redundancy                                             |
|                     | Flooding          | Client puzzles                                                                    |
| Transport           | Desynchronization | Authentication                                                                    |



# Challenges for the IoT Security

## According to IoT Architecture and Components

- Diversity of devices, capabilities, standards, programming environment
- Lots of these devices store personal data – e.g. fitness/health, home security, etc.
- Application driven field – innovation comes from non-tech people
  - First to market may mean no/little security
- Potential threats/attacks a guess work.



# Challenges for the IoT Security

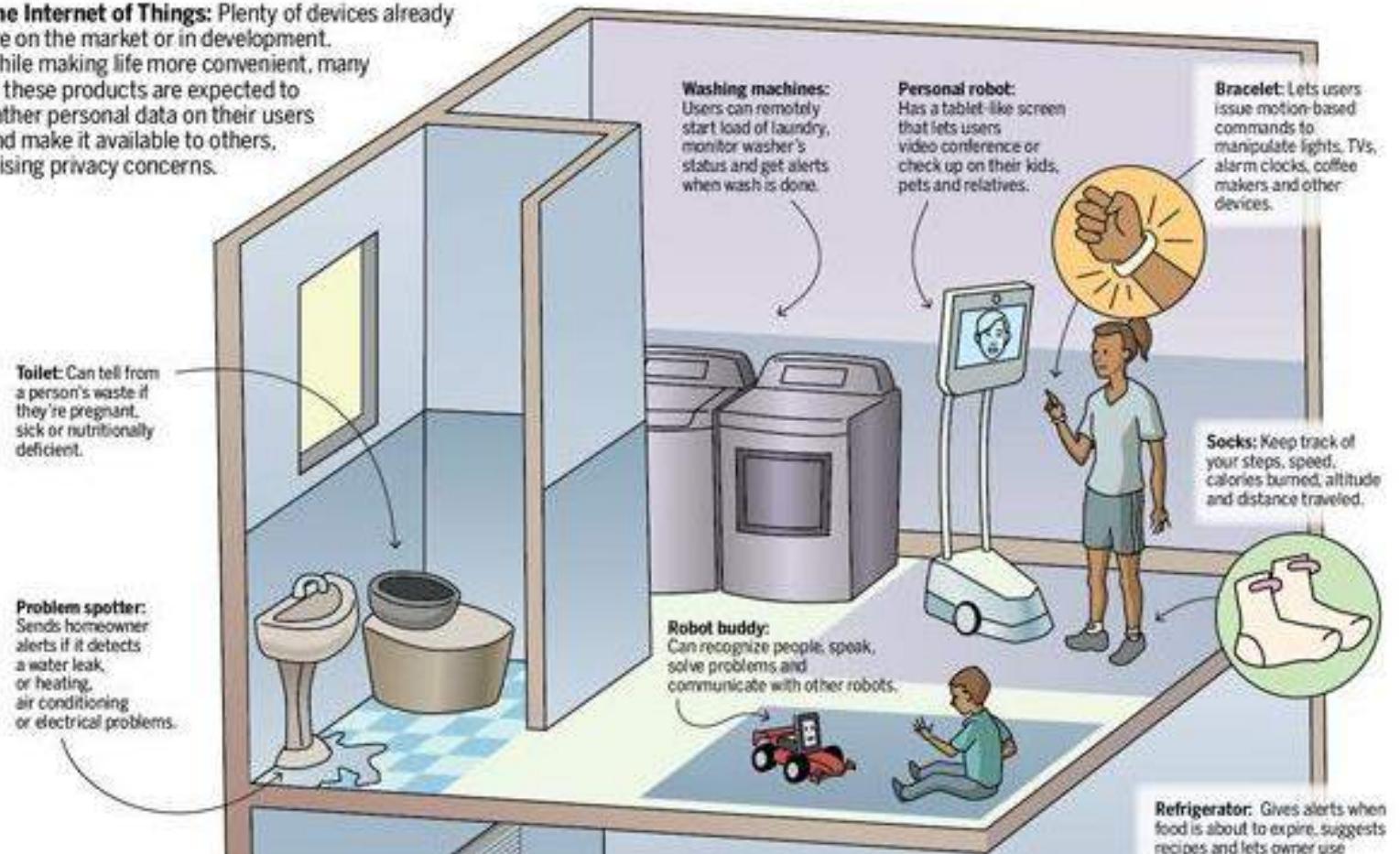
## Unique Challenges

- IoT relies on **microcontrollers** with **limited memory** and **computational power**
  - This often makes it impractical to implement approaches designed for powerful computers
  - This in turn requires constrained IoT devices to be hidden behind secure gateways
- Threats based upon gaining physical access to IoT devices
- How to bootstrap trust and security, and ways that this can unravel
- Evolving technology
  - More powerful Systems on a Chip (SOC) embedding hardware security support
  - Elliptic Curve Cryptography with reduced computational demands
- Anything that is exposed to the Internet must be securely software upgradable
- User experience must be good enough to avoid becoming a weak link in the chain
- The necessity of keeping up to date with security best practices

## How Secure are All These Devices? (1/2)

### A world filled with smart gadgets

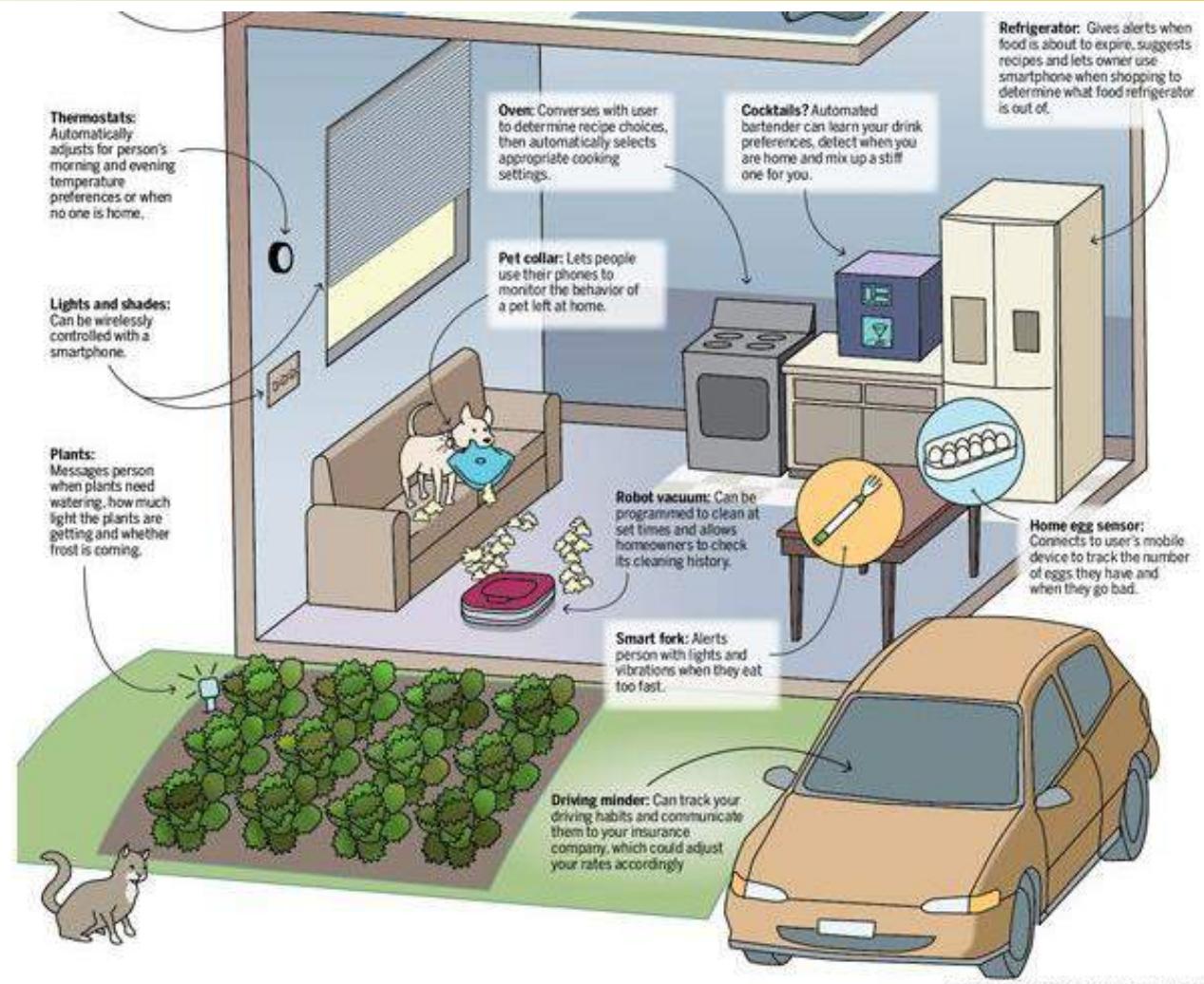
**The Internet of Things:** Plenty of devices already are on the market or in development. While making life more convenient, many of these products are expected to gather personal data on their users and make it available to others, raising privacy concerns.



# Challenges for the IoT Security



## How Secure are All These Devices? (2/2)



STEVE JOHNSON, JEFF DURHAM/BAY AREA NEWS GROUP



# Challenges for the IoT Security

## IoT Devices (Weakness)

- Devices are not reachable
  - Most of the time a device is not connected
- Devices can be lost and stolen
  - Makes security difficult when the device is not connected
- Devices are not crypto-engines
  - Strong security difficult without processing power
- Devices have finite life
  - Credentials need to be tied to lifetime
- Devices are transportable
  - Will cross borders
- Devices need to be recognised by many readers
  - What data is released to what reader?



# Challenges for IoT the Security

## IoT and Big Data

- Lots of sensors will generate a vast amount of data
  - API Research estimated 200 exabytes in 2014 and 1.6 zettabytes in 2020
  - 90% is currently processed locally, although this varies by domain
- This creates a greater volume of sensitive data, creating a greater risk of
  - Data Confidentiality
  - Data and identity theft
  - Device manipulation
  - Data falsification
  - Data ownership
  - Data lifecycle management
  - Data Publication
  - IP theft, server/network manipulation, etc.
- Privacy Preserving Data Correlation
  - Personal and population privacy
  - Privacy enhancing techniques
  - Data service monetization
- Impact of introduction of data consolidation and analytics at network edge
  - Cisco, HPE and others
  - App platforms in the cloud or at the network edge will be targets for attacks



## IoTs and Big Data

**CNNMoney** A Service of CNN, Fortune & Money

FORTUNE Money

Home Video Business News Markets My Portfolio Investing Economy Tech Personal Finance

All Latest Stories Companies World The Buzz Fortune 500 Interactive Video

**Target: 40 million credit cards compromised**

By CNNMoney Staff @CNNMoney December 18, 2013 4:41 PM ET

Recommend 52K

Share

Print

Get e-mail alerts

ON Money

A service of the California HealthCare Foundation

**iHealthBeat**  
Reporting Technology's Impact on Health Care

August 02, 2012 - Topic: Privacy and Security

### Nearly 21M Affected by Large Health Data Breaches Since 2009

Nearly 21 million individuals have been affected by large health data breaches since HHS' Office for Civil Rights began publicly reporting such incidents in September 2009, Modern Healthcare reports (Conn. Modern Healthcare, 8/1).

### THE WALL STREET JOURNAL

wsj.com

BUSINESS | June 9, 2011

### Hacking At Citi Is Latest Data Scare

By VICTORIA MCGRANE And RANDALL SMITH

Citigroup Inc. plans to send replacement credit cards to about 100,000 North American customers after its systems were breached by a hacking attack affecting about 200,000 accounts.



Citi said on Thursday that the hacked accounts amounted to about 1% of its 21 million North American card customers and that it has referred the incident to law enforcement. The bank said it is contacting affected customers and has implemented procedures to prevent a recurrence.

### Reports: 77 Million PlayStation Network Accounts Compromised

By Nick Mediati, PCWorld

To say that Sony has a mess on its hands may be the understatement of the year. On Tuesday, Sony posted a blog entry that provides some more information on the recent PlayStation Network hack. According to Sony, hackers obtained users' names, addresses, e-mail addresses, birthdates, and account login and password, and may have also taken users' security questions and answers. If you set up a sub-account for your child, that information may also be in hackers' hands.

- February 2016 IEEE Experts in Technology & Policy meeting identified a number of areas where we can start:
  - Education & Ethics
  - Data localization
  - Identity management
  - Technology policy development process
  - Autonomy
  - Accountability
  - Tradeoff adjudication
  - Solutions roadmap creation
  - End-to-end security/privacy by design

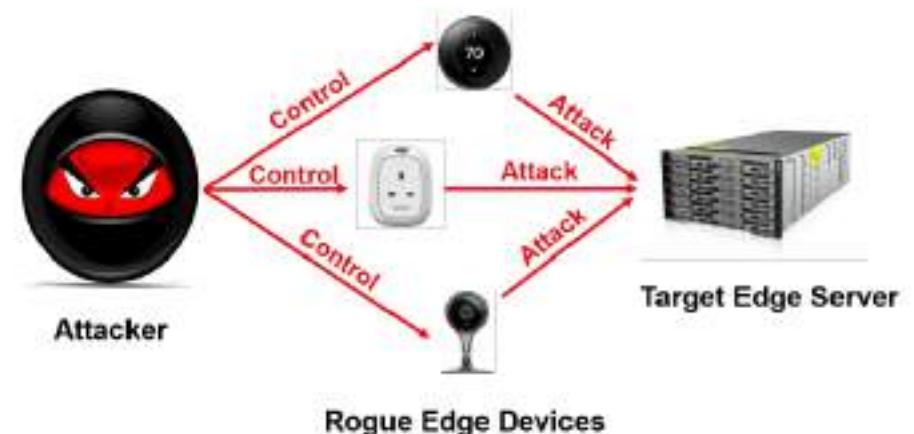
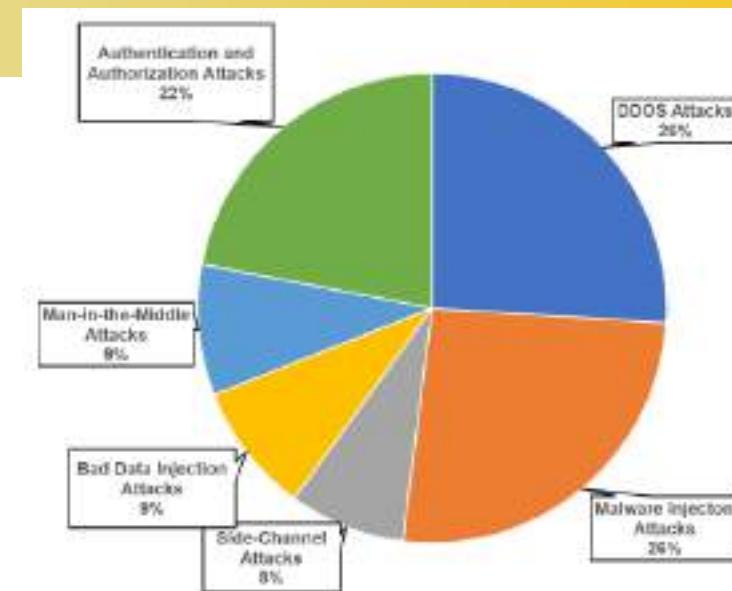
IoT Security Foundation

<https://www.iotsecurityfoundation.org/iot-security-resources/>

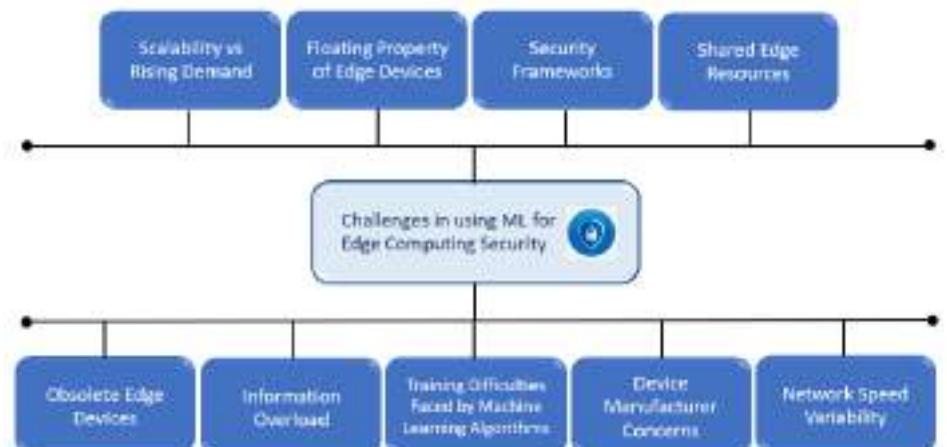


## Best Practices

- Use access control and surveillance to enhance the physical security at the edge.
- Control edge configuration and operation from central IT operations.
- Establish audit procedures to control data and application hosting changes at the edge.
- Apply the highest level of network security possible between devices/users and edge facilities.
- Treat the edge as a part of the public cloud portion of your IT operation.
- Monitor and log all edge activity, particularly activity relating to operations and configuration.



# ML Assisted Security and Privacy Provisioning for Edge Computing



| Attack        | Reference  | Security Breach Technique    | ML Tool                                                                                   | Dataset                                                                                                                  | Performance Measures                                                                                       |
|---------------|------------|------------------------------|-------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|
| DDoS          | [56]       | Packet traffic               | Neural network                                                                            | Vanderbilt Prowler SD                                                                                                    | Mean Square Error                                                                                          |
|               | [57]       | Access control               | Multivariate correlation analysis                                                         | KDD Cup 99                                                                                                               | Accuracy, FPR, TNR, Detection Rate                                                                         |
|               | [58]       | Data traffic                 | SVM                                                                                       | SD using the probabilistic wireless network simulator (Prowler)                                                          | Accuracy, Time taken                                                                                       |
|               | [59]       | Data traffic                 | Neural network                                                                            | NSL-KDD                                                                                                                  | Accuracy, Precision, Recall, F1 score                                                                      |
|               | [60]       | Packet flooding              | Random Forest                                                                             | SD of TCP and UDP flood attacks                                                                                          | Precision, F1 score, Accuracy, Error rate, Mameva correlation coefficient                                  |
|               | [61]       | Malicious traffic and volume | SVM                                                                                       | KDD99                                                                                                                    | Detection Rate Mean Threshold                                                                              |
| Eavesdropping | [62]       | Malicious data traffic       | Extreme Learning Machine (ELM) using neural network                                       | CTU dataset                                                                                                              | Accuracy, Precision, Recall, F1 score                                                                      |
|               | [63]       | Authentication               | Non-parametric Bayesian                                                                   | RSSI data                                                                                                                | Proximity passing rate, number of breaches correctly identified by the algorithm proposed                  |
| Malware       | [64]       | Access control               | KNN                                                                                       | Public (MalGromo) and Private (Self-collected)                                                                           | Precision, Recall, Fmeasure, TPR, FPR                                                                      |
|               | [65]       | Resource information         | SVM                                                                                       | Real time data from normal and malicious applications                                                                    | Accuracy, Precision, FPR                                                                                   |
|               | [66]       | Access control               | Ensemble learning                                                                         | Android malware samples using Java based APK analysis tool                                                               | Accuracy, FPR, FNR, TPR, TNR, Precision                                                                    |
| Intrusion     | [67]–[69]  | Access control               | SVM                                                                                       | TCPDUMP and BSM SD [67], DARPA dataset from the Lincoln Laboratory of MIT [68], Real time dataset obtained from ISP [69] | [Attack Detection Rate, FPR [67]], [TPR, FPR, Precision, Recall, Accuracy [68]], [Accuracy, FPR, TNR [69]] |
|               | [70]–[75]  | Traffic anomaly              | K-NN [70]–[72], [K-NN + Particle Swarm Optimization [73]], [Genetic Algorithm + KNN [74]] | 1998 DARPA BSM audit data [70], KDD Cup 1999 dataset [71], KDD-CLUP 1999 dataset [72]–[74]                               | [FPR, Attack Detection Rate [70]], Accuracy [71], Accuracy [72] [73], [TPR, TNR, FPR, FNR, Accuracy [74]]  |
|               | [76], [77] | Traffic anomaly              | Decision Tree [76], [Hybrid model with Decision Tree + SVM [77]]                          | KDDCap99 dataset [76], NSL-KDD data set [77]                                                                             | [Accuracy, Precision, FPR, FNR, TPR, TNR [76]], [FPR, detection rate [77]]                                 |
|               | [78]–[79]  | Anomalous packets            | Naive Bayes                                                                               | KDD99 [78], NSL-KDD dataset [79]                                                                                         | [Detection rate, FPR [78]], [Accuracy, Root Mean Squared Error (RMSE), TPR [79]]                           |

S. Singh, et. Al. "Machine Learning Assisted Security and Privacy Provisioning for Edge Computing: A Survey", 2021.

# Working Groups on the IoT Security



- OWASP IoT Security Guidance (<https://owasp.org/www-project-internet-of-things/>)
  - IAB Privacy & Security studies
  - RFC 7452 – Architectural Considerations in Smart Object Networking
  - RFC 7456 – Cryptographic algorithm agility
- EU Article 29 Data Protection Working party
  - Anonymization, privacy and the IoT
- NIST (National Institute of Standards and Technology) - Cyber Security Framework
- Track the emerging standards, e.g.
  - W3C Security Activity
  - IETF ACE & JOSE

# OWASP IoT Top 10 - 2018

[https://wiki.owasp.org/index.php/OWASP\\_Internet\\_of\\_Things\\_Project#tab=IoT%20Top%2010](https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT%20Top%2010)

## OWASP IoT Top 10 2014

I1 Insecure Web Interface

I2 Insufficient Authentication/Authorization

I3 Insecure Network Services

I4 Lack of Transport Encryption/Integrity Verification

I5 Privacy Concerns

I6 Insecure Cloud Interface

I7 Insecure Mobile Interface

I8 Insufficient Security Configurability

I9 Insecure Software/Firmware

I10 Poor Physical Security

## OWASP IoT Top 10 2018 Mapping

I3 Insecure Ecosystem Interfaces

I1 Weak, Guessable, or Hardcoded Passwords

I3 Insecure Ecosystem Interfaces

I9 Insecure Default Settings

I2 Insecure Network Services

I7 Insecure Data Transfer and Storage

I6 Insufficient Privacy Protection

I3 Insecure Ecosystem Interfaces

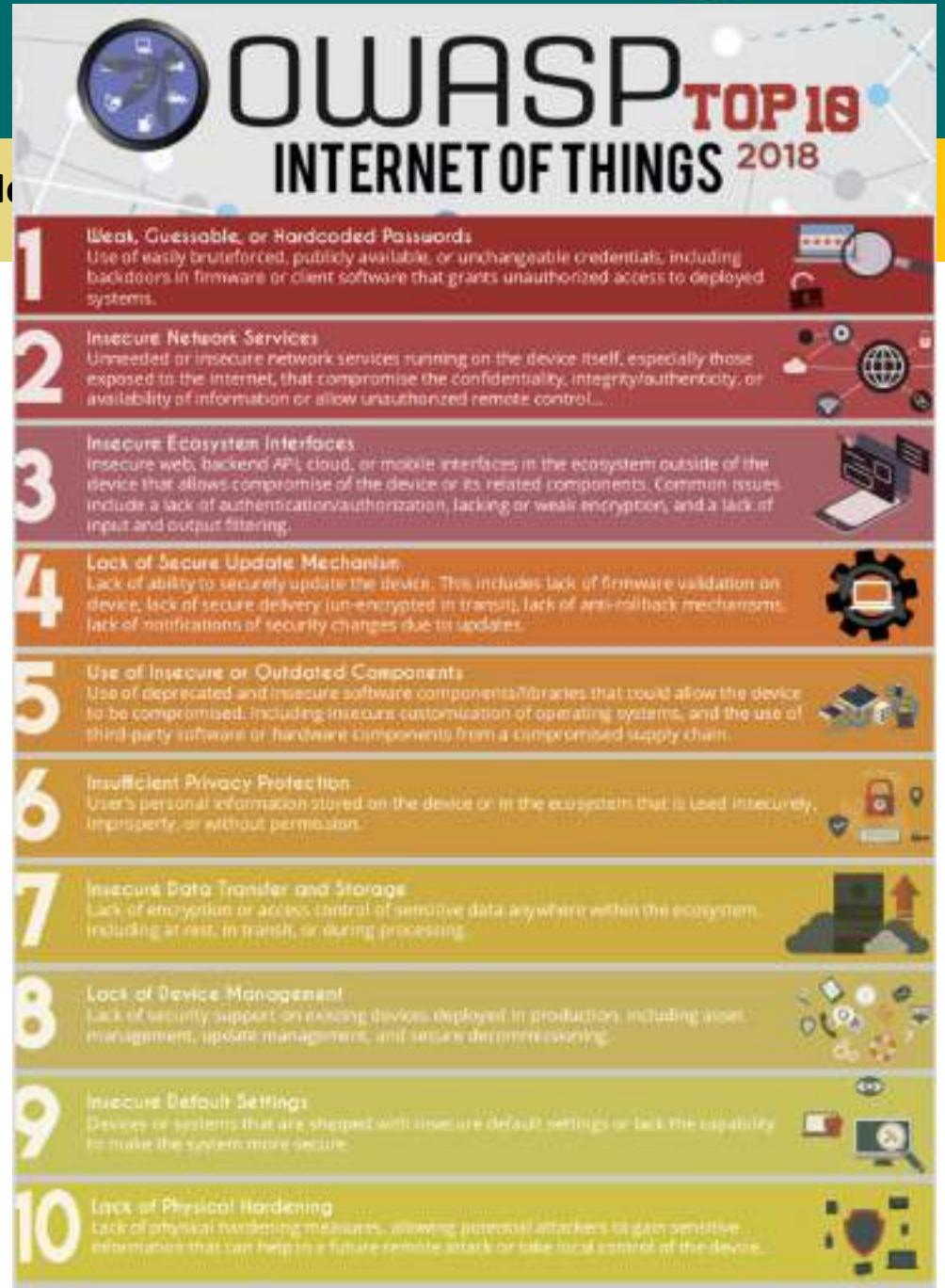
I3 Insecure Ecosystem Interfaces

I9 Insecure Default Settings

I4 Lack of Secure Update Mechanism

I5 Use of Insecure or Outdated Components

I10 Lack of Physical Hardening



Ref: <https://scriptingxss.gitbook.io/owasp-iot-top-10-mapping-project/>



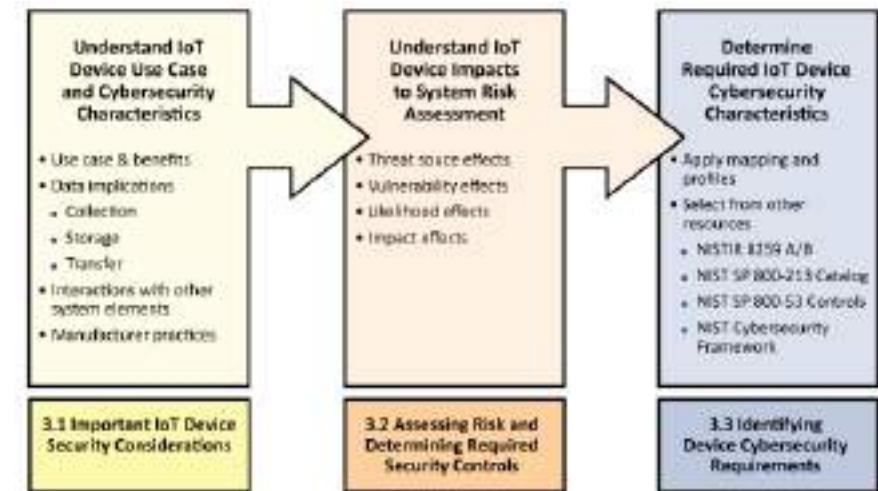
# OWASP IoT and Wrap Up



# NIST: IoT Cybersecurity Framework



- NIST Cyber Security Framework (CSF): Identify-Protect-Detect-Respond-Recover
- NIST Computer Security Resource Center: <https://csrc.nist.gov/>
  - NIST Cybersecurity for IoT Program: <https://www.nist.gov/itl/applied-cybersecurity/nist-cybersecurity-iot-program>
  - Securing the Industrial Internet of Things: Cybersecurity for Distributed Energy Resources: <https://www.nccoe.nist.gov/sites/default/files/2021-09/es-iiot-nist-sp1800-32-draft.pdf>, September 2021.
  - Methodology for Characterizing Network Behavior of Internet of Things Devices: <https://nvlpubs.nist.gov/nistpubs/ir/2022/NIST.IR.8349-draft.pdf>
  - NIST Special Publication (SP) 800-213: **IoT Device Cybersecurity Guidance** for the Federal Government: Establishing IoT Device Cybersecurity Requirements;  
<https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-213.pdf>, November 2021.



NIST SP.800.213

# The IoT Security Recommendations



- Accommodate IoT with existing practices:
  - Policies, Procedures, & Standards
  - Awareness Training
  - Risk Management
  - Vulnerability Management
  - Forensics
- Plan for IoT growth:
  - Additional types of logging, log storage: Can you find the needle in the haystack?
  - Increased network traffic: will your firewall / IDS / IPS be compatible and keep up?
  - Increased demand for IP addresses both IPv4 and IPv6
  - Increased network complexity – should these devices be isolated or segmented?

# References



- H. F. Atlam and G. B. Wills, “IoT Security, Privacy, Safety, and Ethics”, in Digital Twin Technologies and Smart Cities, Springer, 2020.
- Presentation Slides: “CSG Business Meeting: End-to-End Trust & Security Open Architecture for IoT” by Scot Ransbottom, Virginia Tech and Florence Hudson, Internet2, April 28, 2016.
- Tackling Data Security and Privacy Challenges for the Internet of Things, Dave Raggett, June 14, 2016.
- Brian Russell and Drew Van Duren, Practical Internet of Things Security, Packt Publishing, 2017
- Shancang Li and Li Da Xu, Securing the Internet of Things, Packt Publishing, 2017
- Maurice Dawson, Mohamed Eltayeb and Marwan Omar, Security Solutions for Hyperconnectivity and the Internet of Things, IGI Global, 2017.
- N. Jeyanthi and R. Thandeeswaran, Security Breaches and Threat Prevention in the Internet of Things, IGI Global, 2017.
- [https://www.owasp.org/index.php/OWASP\\_Internet\\_of\\_Things\\_Project](https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project)
- <https://www.experfy.com/training/courses/securing-enterprise-internet-of-things-implementations>
- <https://www.iotsecurityfoundation.org/>
- <https://users.ece.cmu.edu/~vsekar/Teaching/Fall21/18739//lectures/>
- <https://www.coursera.org/lecture/iot/lecture-3-2-risks-privacy-and-security-GnJON>
- [https://www.iot-now.com/2021/11/24/115797-arrow-electronics-psa-certified-development-kit-accelerates-time-to-market-for-iot-devices/?fbclid=IwAR1ir\\_fRs-D4gvbWujhT4\\_04PCOjdIrfrz8mhSmbSg4qm-WkB-ZWnJTNFhA](https://www.iot-now.com/2021/11/24/115797-arrow-electronics-psa-certified-development-kit-accelerates-time-to-market-for-iot-devices/?fbclid=IwAR1ir_fRs-D4gvbWujhT4_04PCOjdIrfrz8mhSmbSg4qm-WkB-ZWnJTNFhA)



# Part 3

# Battery Life & Power Management



## Example 1



- Battery Life (Battery Last) – (hr)=  
Battery Capacity in mAh / Load(Draw,consumption) Current in mA
- If we are pessimistic and assume that the Pi will use 800mA (with WiFi + a couple of peripherals+CPU processing programs),  
then we can expect about  $1000 / 800 = 1.25$  hours of operation from a 1000mA battery.

Ref: <https://electronics977.rssing.com/chan-8359630/latest.php>



## Power Modes

# Deep sleep?

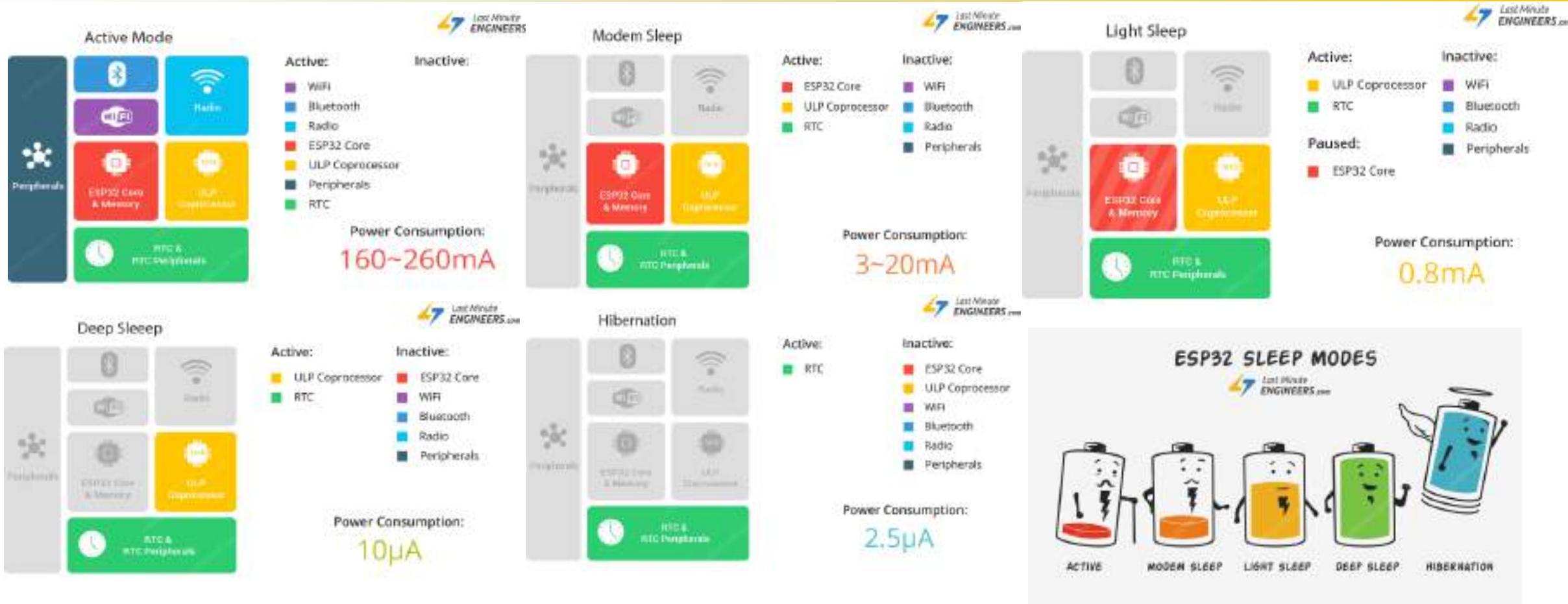
## Overview of power modes



# ESP32 Power Management



## ESP32



<https://lastminuteengineers.com/esp32-sleep-modes-power-consumption/>

[https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/sleep\\_modes.html](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/sleep_modes.html)

[https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/power\\_management.html](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/power_management.html)



## Example 2

Based on ESP32 Ultra-Low-Power (ULP) Management

Suppose, for the deep-sleep operation,  $175 \mu\text{A} * 24 \text{ h} = 4.2 \text{ mAh}$  per day,

For the transmission bursts  $140 \text{ mA} * 1 \text{ s} / 3600 \text{ s/h} * 24 = 0.93 \text{ mAh}$ ,

Therefore, the total battery consumption 5.13 mAh per day.

Based on the battery capacity (1350mA),

this gives a calculated service life of 263.16 days.

<http://www.of-things.de/battery-life-calculator.php>

<https://www.macnica.eu/>

<https://diyi0t.com/reduce-the-esp32-power-consumption/>

<https://community.hiveeyes.org/t/low-power-esp32-hardware-and-software/538>