Chapter 1

Introduction to IoT

Outline

- IoT definition
- Characteristics of IoT
- Physical Design of IoT
- Logical Design of IoT
- IoT Protocols
- IoT Levels & Deployment Templates

Definition of IoT

A dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual "things" have identities, physical attributes, and virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network, often communicate data associated with users and their environments.

Characteristics of IoT

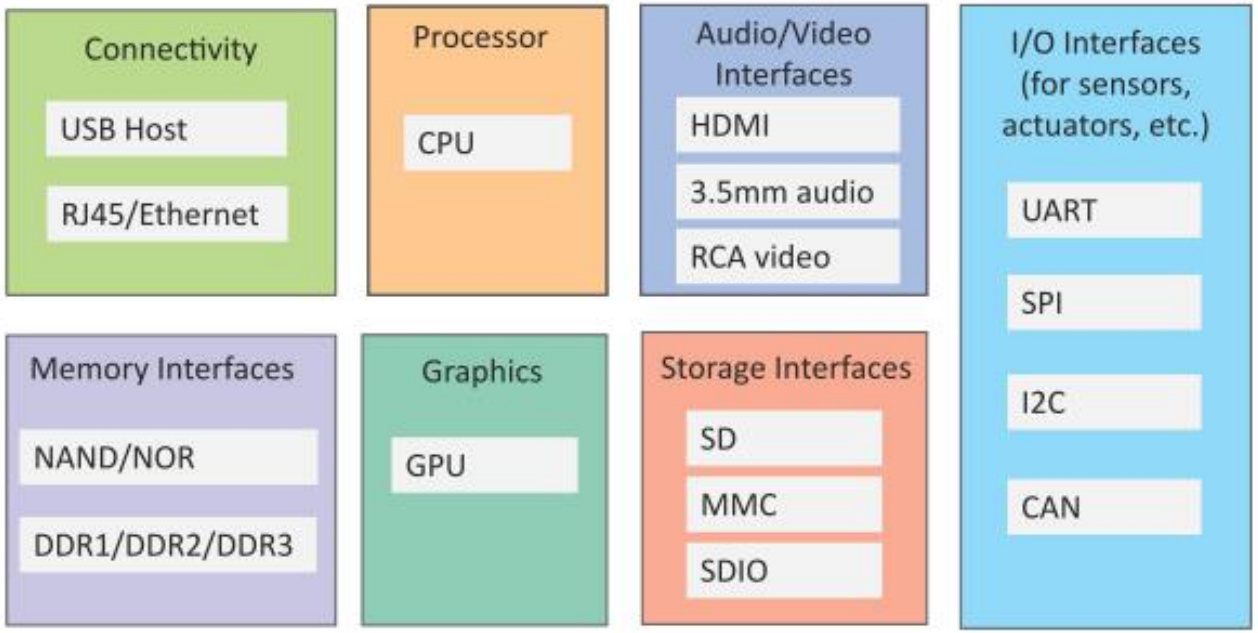
- Dynamic & Self-Adapting
- Self-Configuring
- Interoperable Communication Protocols
- Unique Identity
- Integrated into Information Network

Physical Design of IoT

- The "Things" in IoT usually refers to IoT devices which have unique identities and can perform remote sensing, actuating and monitoring capabilities.
- IoT devices can:
  - Exchange data with other connected devices and applications (directly or indirectly), or
  - Collect data from other devices and process the data locally or
  - Send the data to centralized servers or cloud-based application back-ends for processing the data, or
  - Perform some tasks locally and other tasks within the IoT infrastructure, based on temporal and space constraints

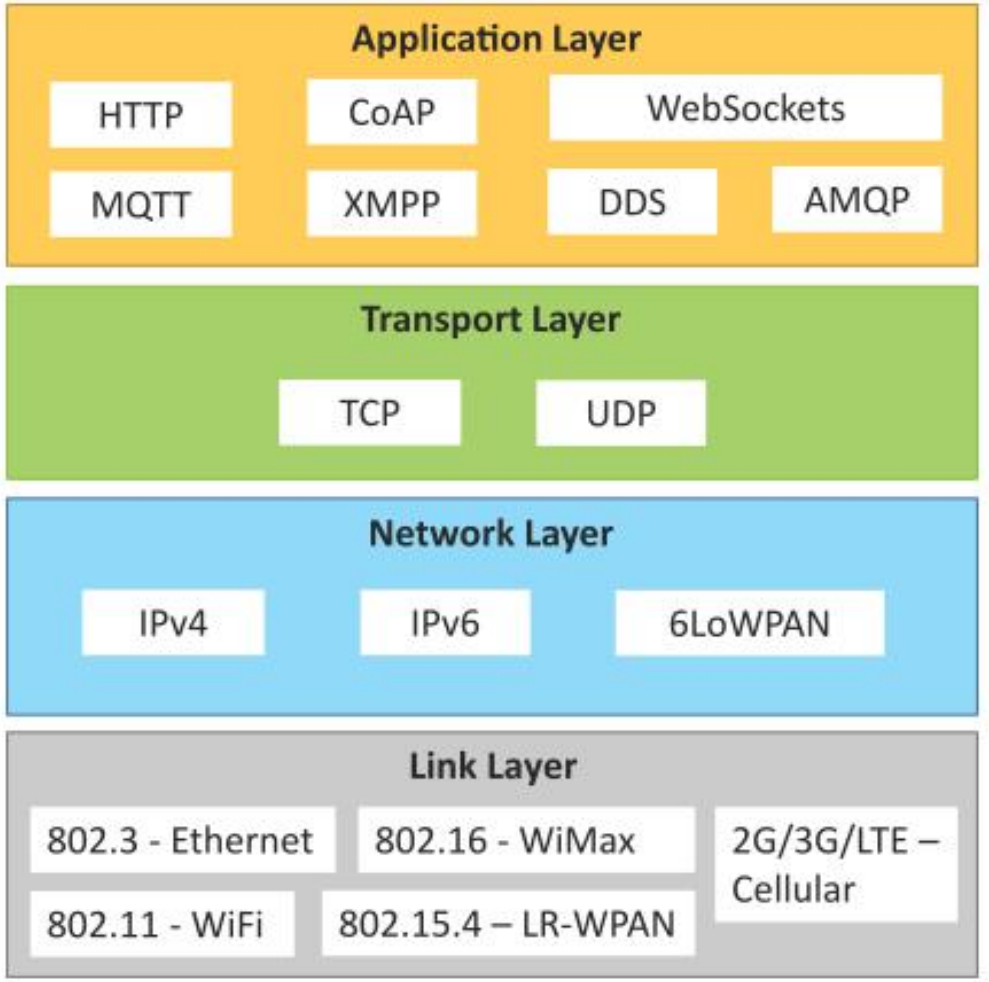
Generic block diagram of an IoT Device

- An IoT device may consist of several interfaces for connections to other devices, both wired and wireless.
  - I/O interfaces for sensors
  - Interfaces for Internet connectivity
  - Memory and storage interfaces
  - Audio/video interfaces.



IoT Protocols

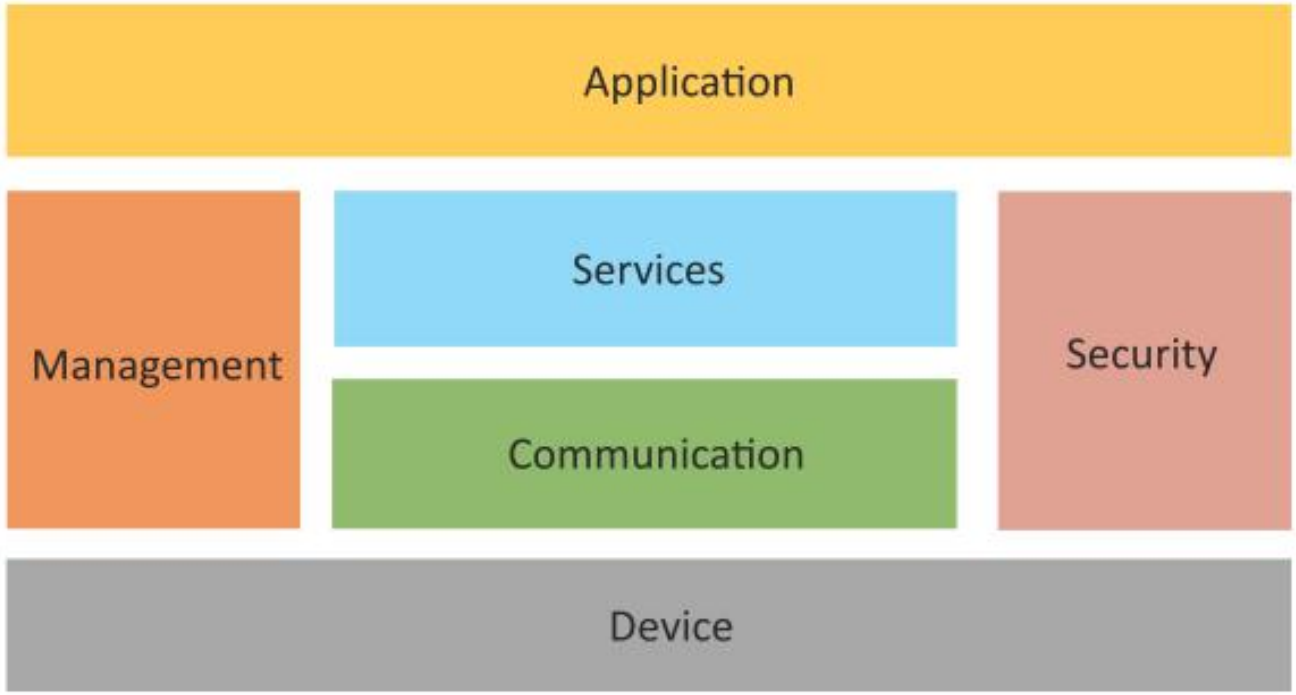
- Link Layer
  - 802.3 – Ethernet
  - 802.11 – WiFi
  - 802.16 – WiMax
  - 802.15.4 – LR-WPAN
  - 2G/3G/4G
- Network/Internet Layer
  - IPv4
  - IPv6
  - 6LoWPAN
- Transport Layer
  - TCP
  - UDP
- Application Layer
  - HTTP
  - CoAP
  - WebSocket
  - MQTT
  - XMPP
  - DDS
  - AMQP



Logical Design of IoT

- Logical design of an IoT system refers to an abstract representation of the entities and processes without going into the low-level specifics of the implementation.

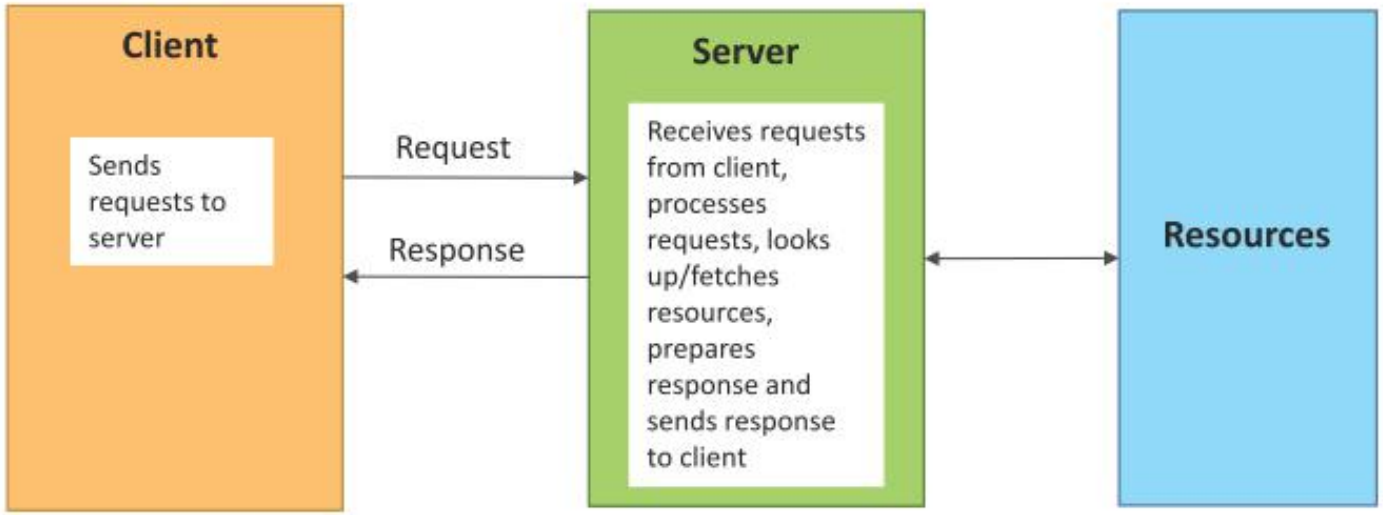
- An IoT system comprises of a number of functional blocks that provide the system the capabilities for identification, sensing, actuation, communication, and management.



Request-Response communication model

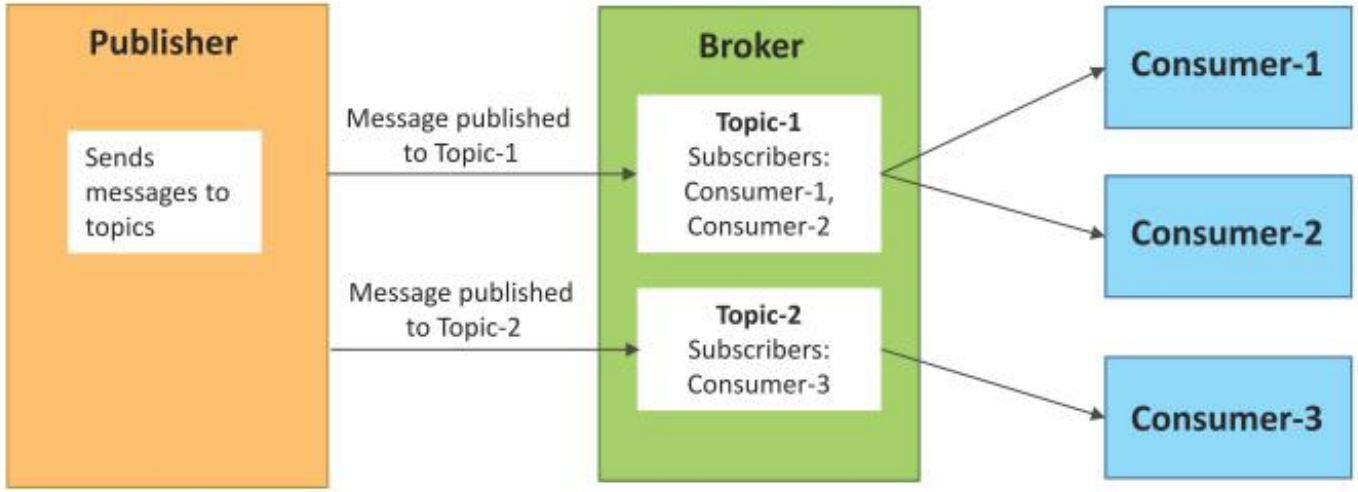
- Request-Response is a communication model in which the client sends requests to the server and the server responds to the requests.

- When the server receives a request, it decides how to respond, fetches the data, retrieves resource representations, prepares the response, and then sends the response to the client.



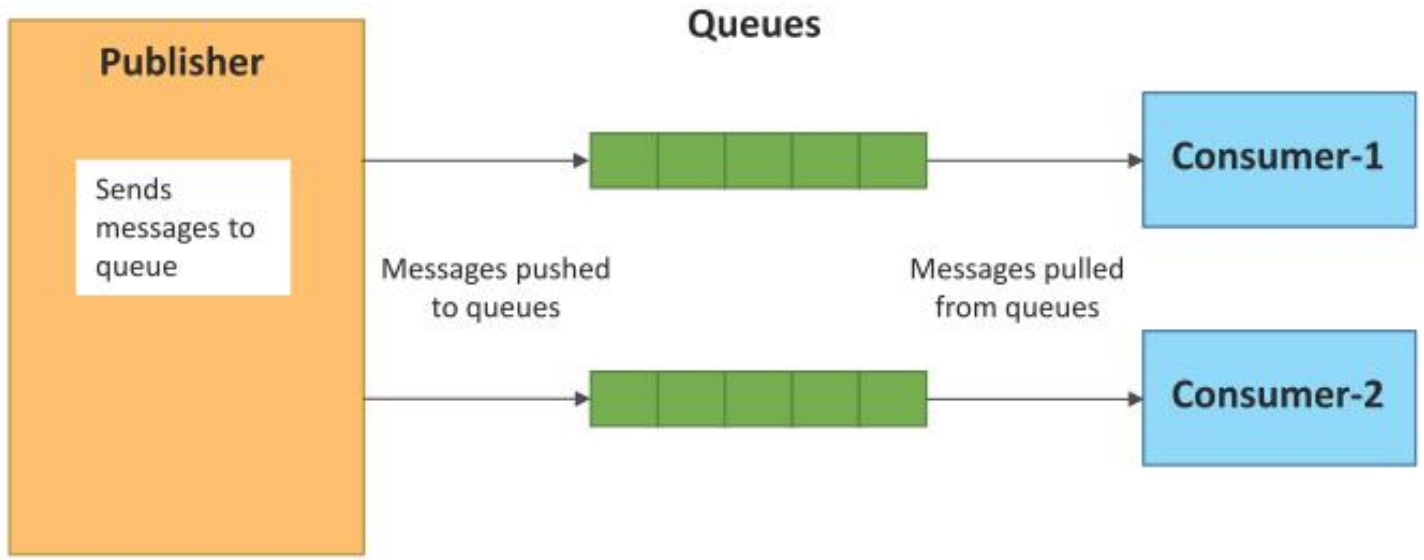
Publish-Subscribe communication model

- Publish-Subscribe is a communication model that involves publishers, brokers and consumers.
- Publishers are the source of data. Publishers send the data to the topics which are managed by the broker. Publishers are not aware of the consumers.
- Consumers subscribe to the topics which are managed by the broker.
- When the broker receives data for a topic from the publisher, it sends the data to all the subscribed consumers.



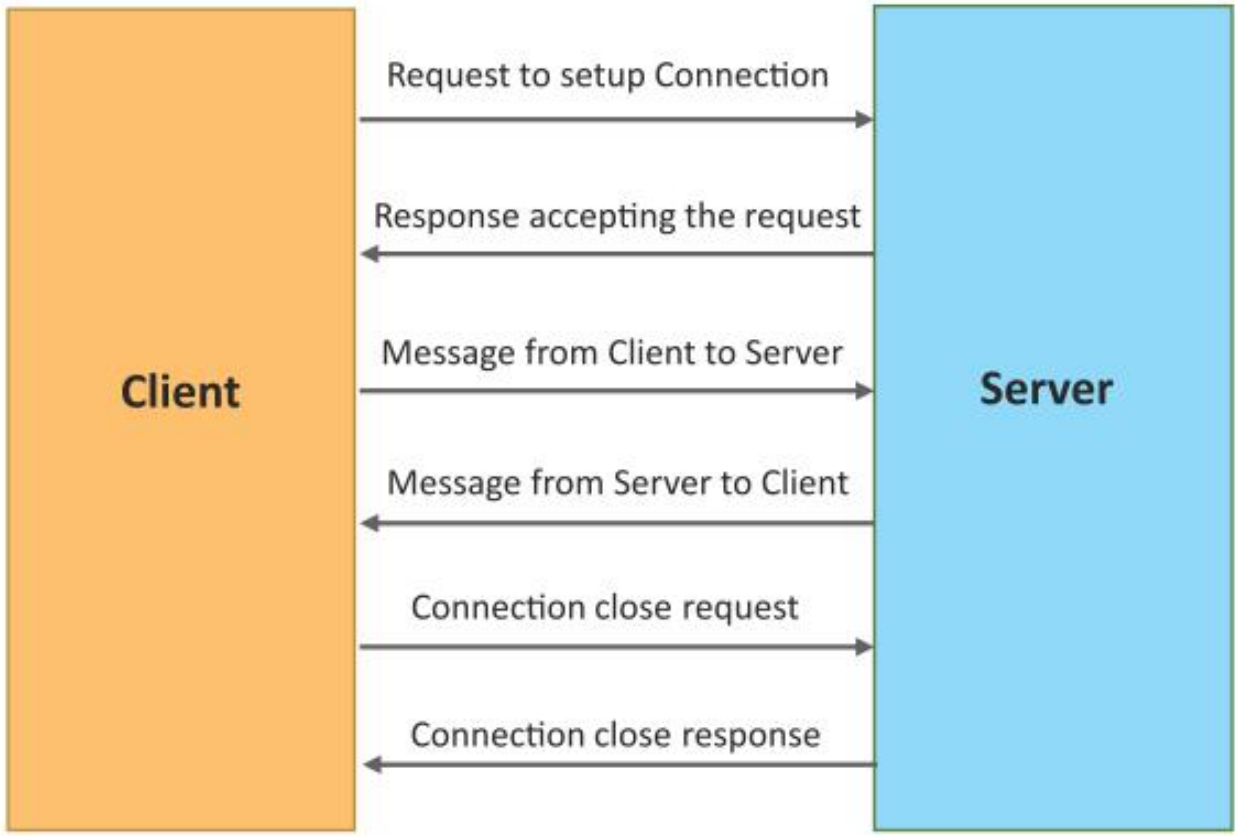
Push-Pull communication model

- Push-Pull is a communication model in which the data producers push the data to queues and the consumers pull the data from the queues. Producers do not need to be aware of the consumers.
- Queues help in decoupling the messaging between the producers and consumers.
- Queues also act as a buffer which helps in situations when there is a mismatch between the rate at which the producers push data and the rate rate at which the consumers pull data.



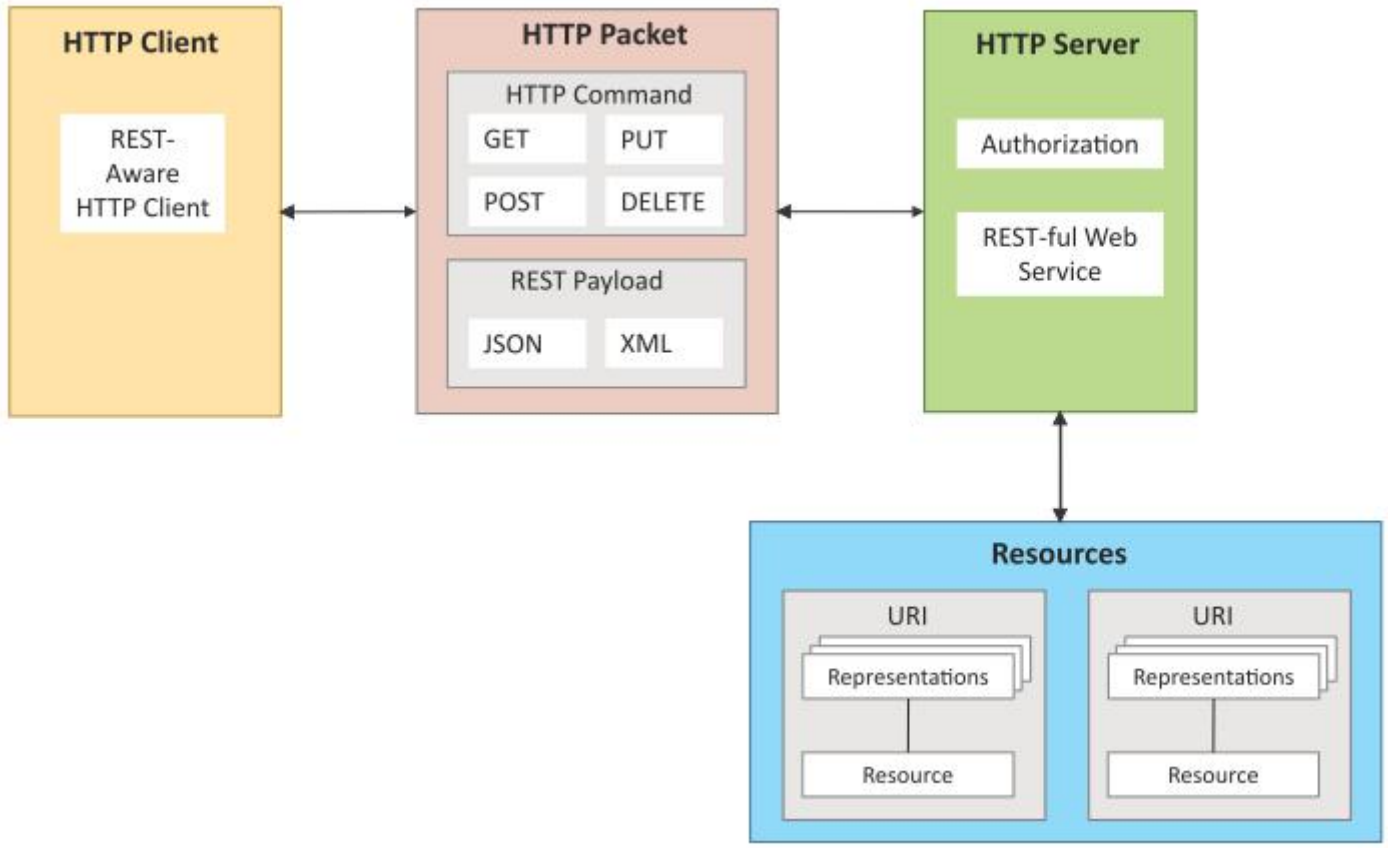
Exclusive Pair communication model

- Exclusive Pair is a bidirectional, fully duplex communication model that uses a persistent connection between the client and server.
- Once the connection is setup it remains open until the client sends a request to close the connection.
- Client and server can send messages to each other after connection setup.



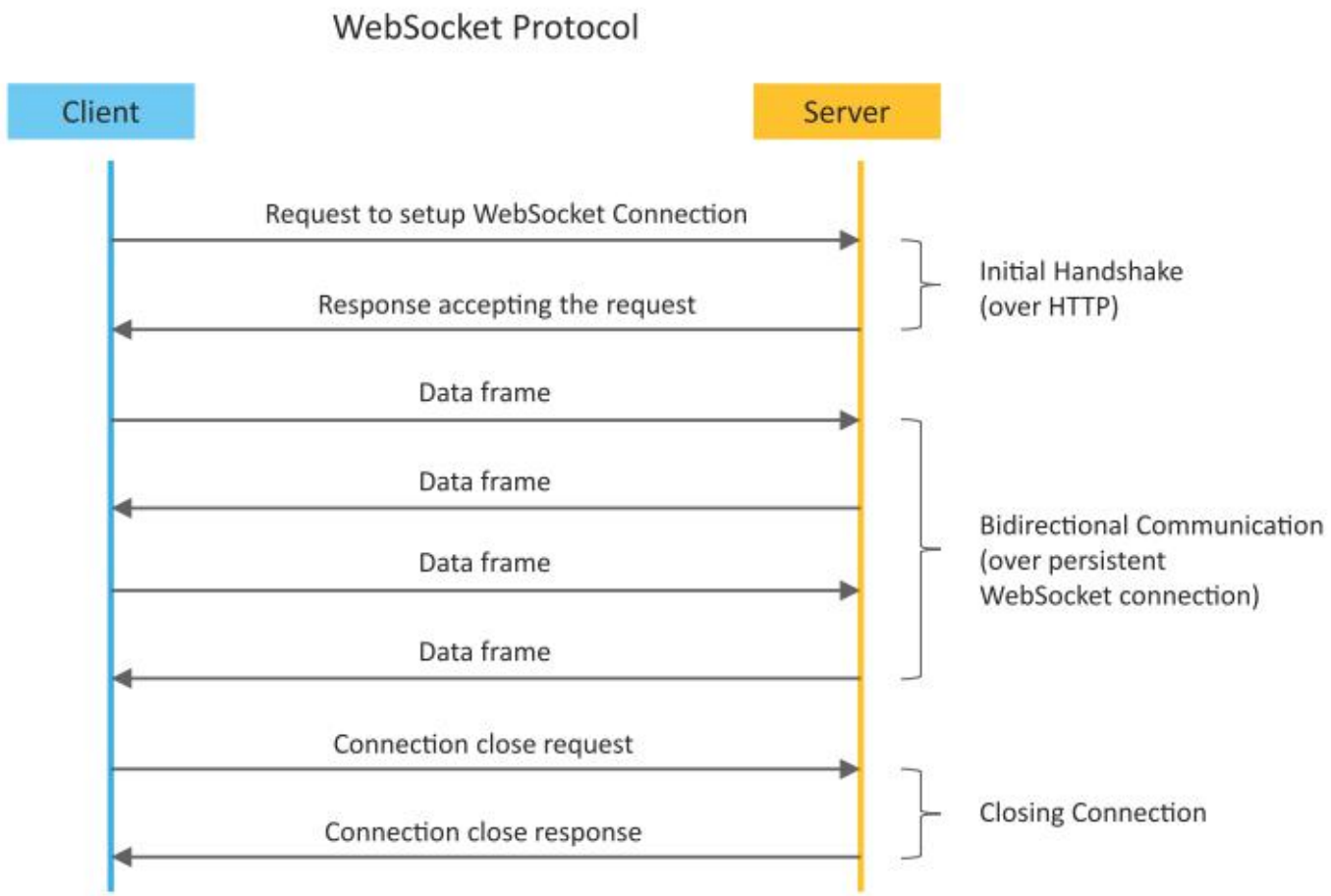
REST-based Communication APIs

- Representational State Transfer (REST) is a set of architectural principles by which you can design web services and web APIs that focus on a system's resources and how resource states are addressed and transferred.
- REST APIs follow the request-response communication model.
- The REST architectural constraints apply to the components, connectors, and data elements, within a distributed hypermedia system.



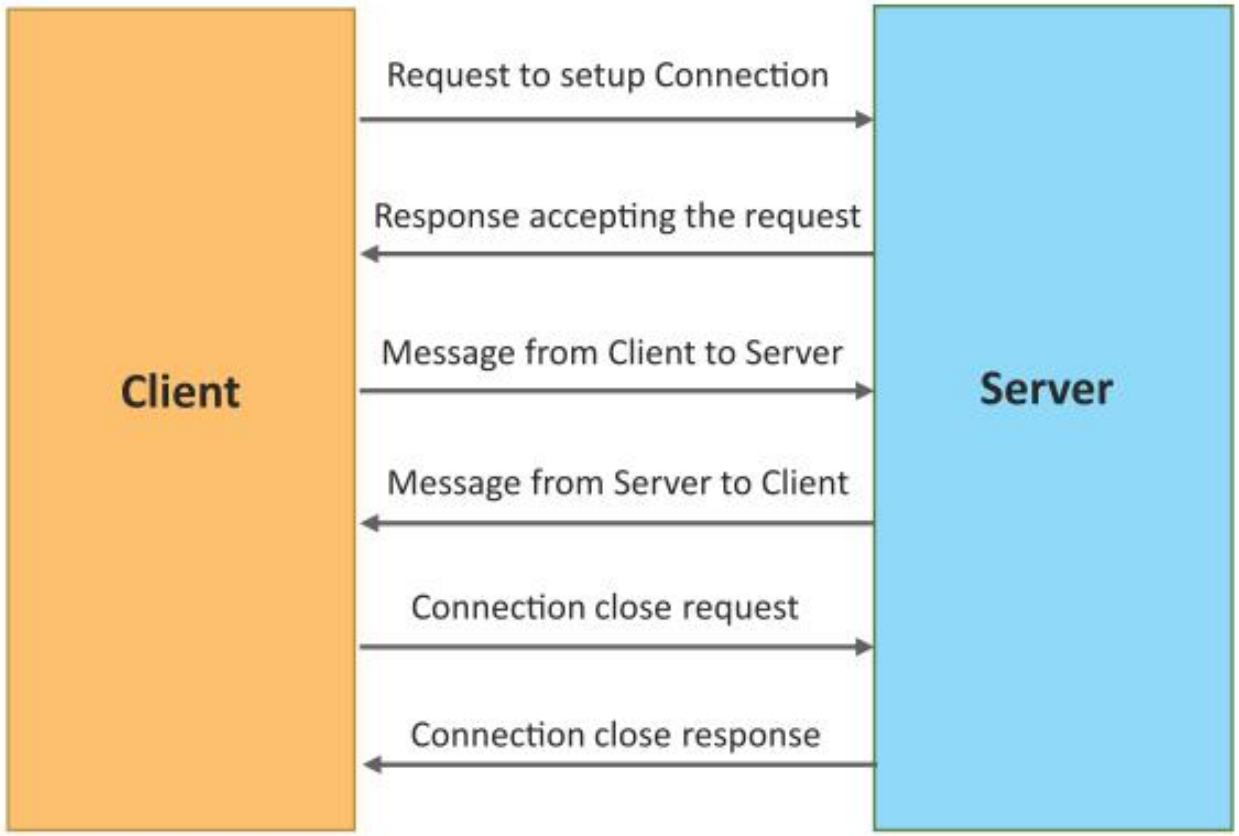
WebSocket-based Communication APIs

- WebSocket APIs allow bi-directional, full duplex communication between clients and servers.
- WebSocket APIs follow the exclusive pair communication model



Exclusive Pair communication model

- Exclusive Pair is a bidirectional, fully duplex communication model that uses a persistent connection between the client and server.
- Once the connection is setup it remains open until the client sends a request to close the connection.
- Client and server can send messages to each other after connection setup.



IoT Levels & Deployment Templates

An IoT system comprises of the following components:

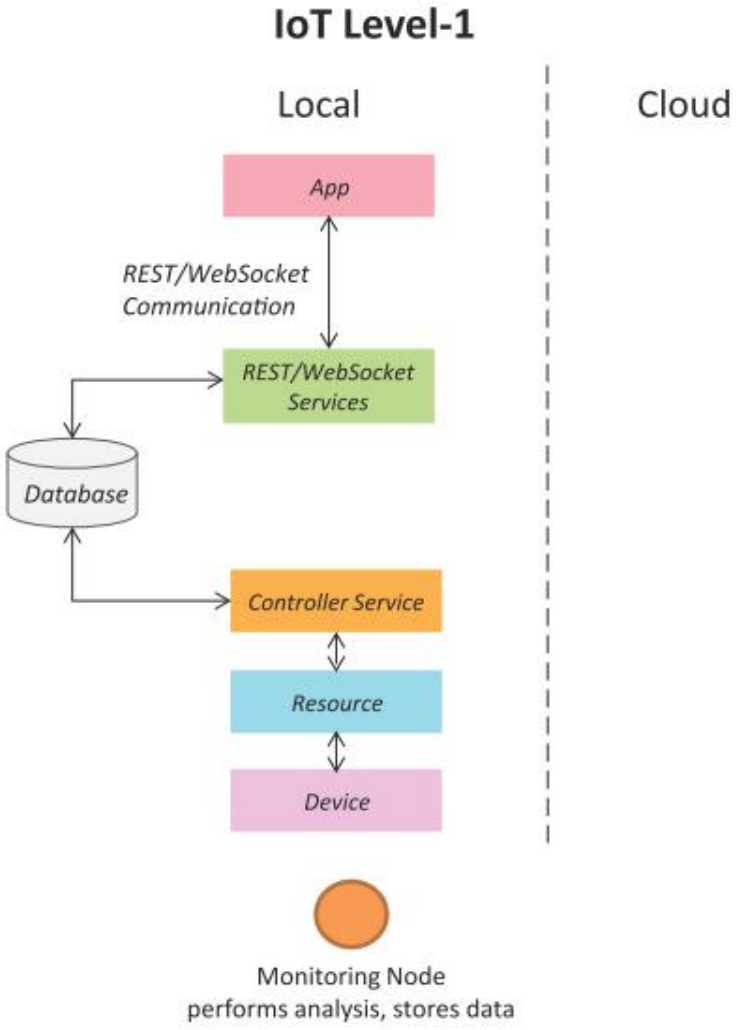
- **Device**: An IoT device allows identification, remote sensing, actuating and remote monitoring capabilities. You learned about various examples of IoT devices in section
- **Resource**: Resources are software components on the IoT device for accessing, processing, and storing sensor information, or controlling actuators connected to the device. Resources also include the software components that enable network access for the device.
- **Controller Service**: Controller service is a native service that runs on the device and interacts with the web services. Controller service sends data from the device to the web service and receives commands from the application (via web services) for controlling the device.

IoT Levels & Deployment Templates

- **Database**: Database can be either local or in the cloud and stores the data generated by the IoT device.
- **Web Service**: Web services serve as a link between the IoT device, application, database and analysis components. Web service can be either implemented using HTTP and REST principles (REST service) or using WebSocket protocol (WebSocket service).
- **Analysis Component**: The Analysis Component is responsible for analyzing the IoT data and generate results in a form which are easy for the user to understand.
- **Application**: IoT applications provide an interface that the users can use to control and monitor various aspects of the IoT system. Applications also allow users to view the system status and view the processed data.

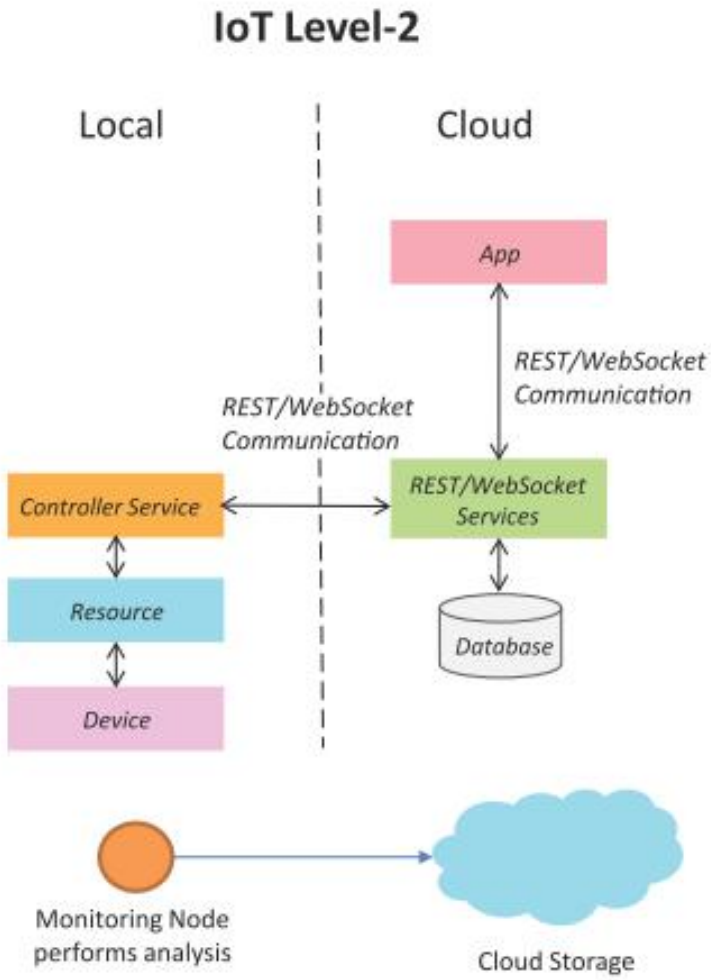
IoT Level-1

- A level-1 IoT system has a single node/device that performs sensing and/or actuation, stores data, performs analysis and hosts the application
- Level-1 IoT systems are suitable for modeling low-cost and low-complexity solutions where the data involved is not big and the analysis requirements are not computationally intensive.



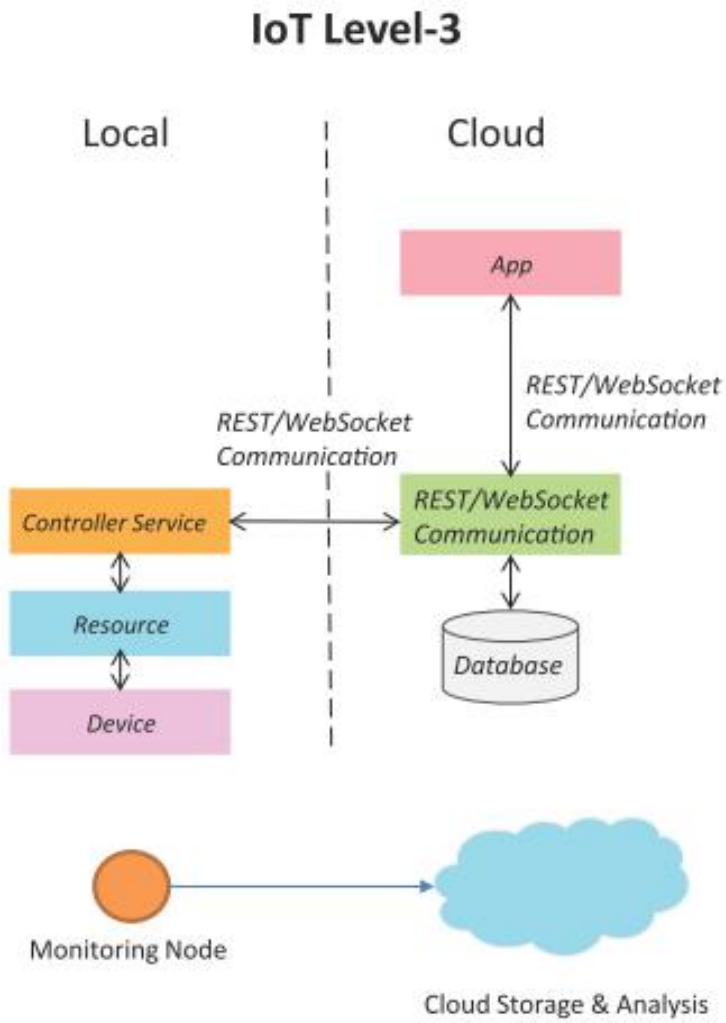
IoT Level-2

- A level-2 IoT system has a single node that performs sensing and/or actuation and local analysis.
- Data is stored in the cloud and application is usually cloud-based.
- Level-2 IoT systems are suitable for solutions where the data involved is big, however, the primary analysis requirement is not computationally intensive and can be done locally itself.



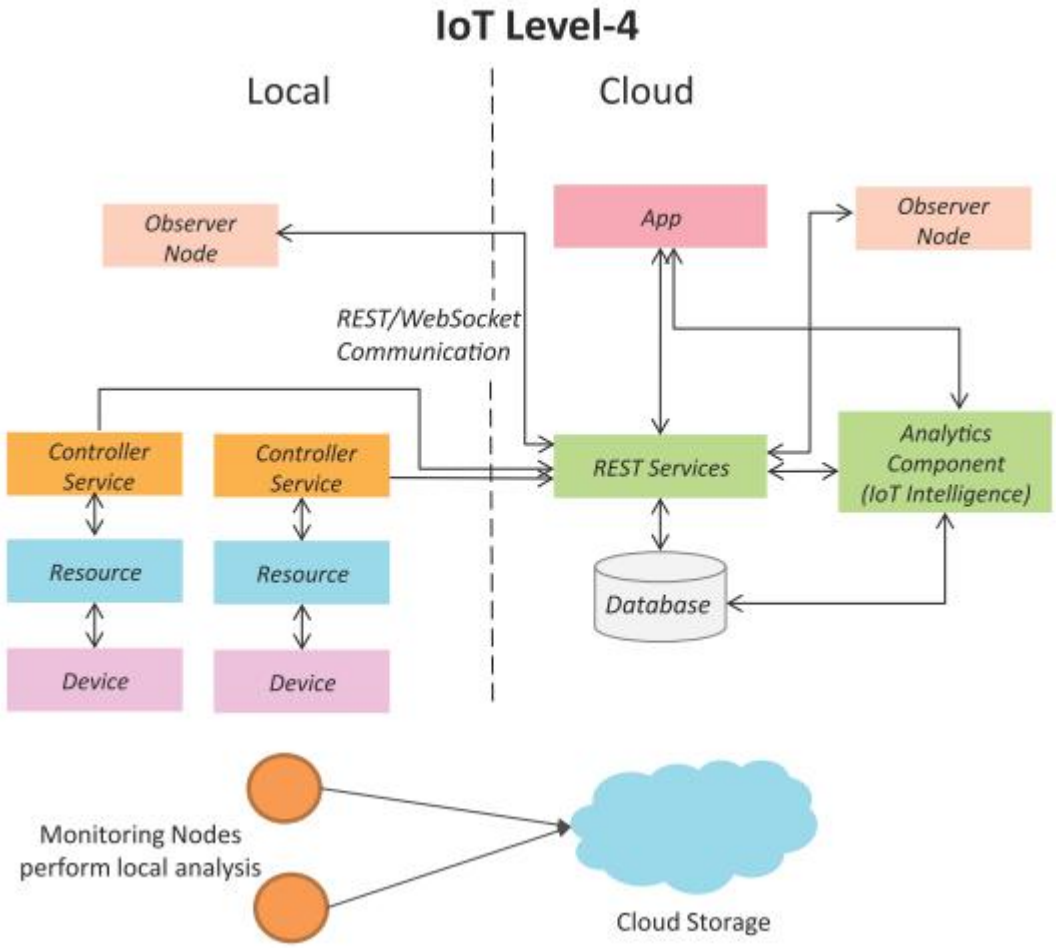
IoT Level-3

- A level-3 IoT system has a single node. Data is stored and analyzed in the cloud and application is cloud-based.
- Level-3 IoT systems are suitable for solutions where the data involved is big and the analysis requirements are computationally intensive.



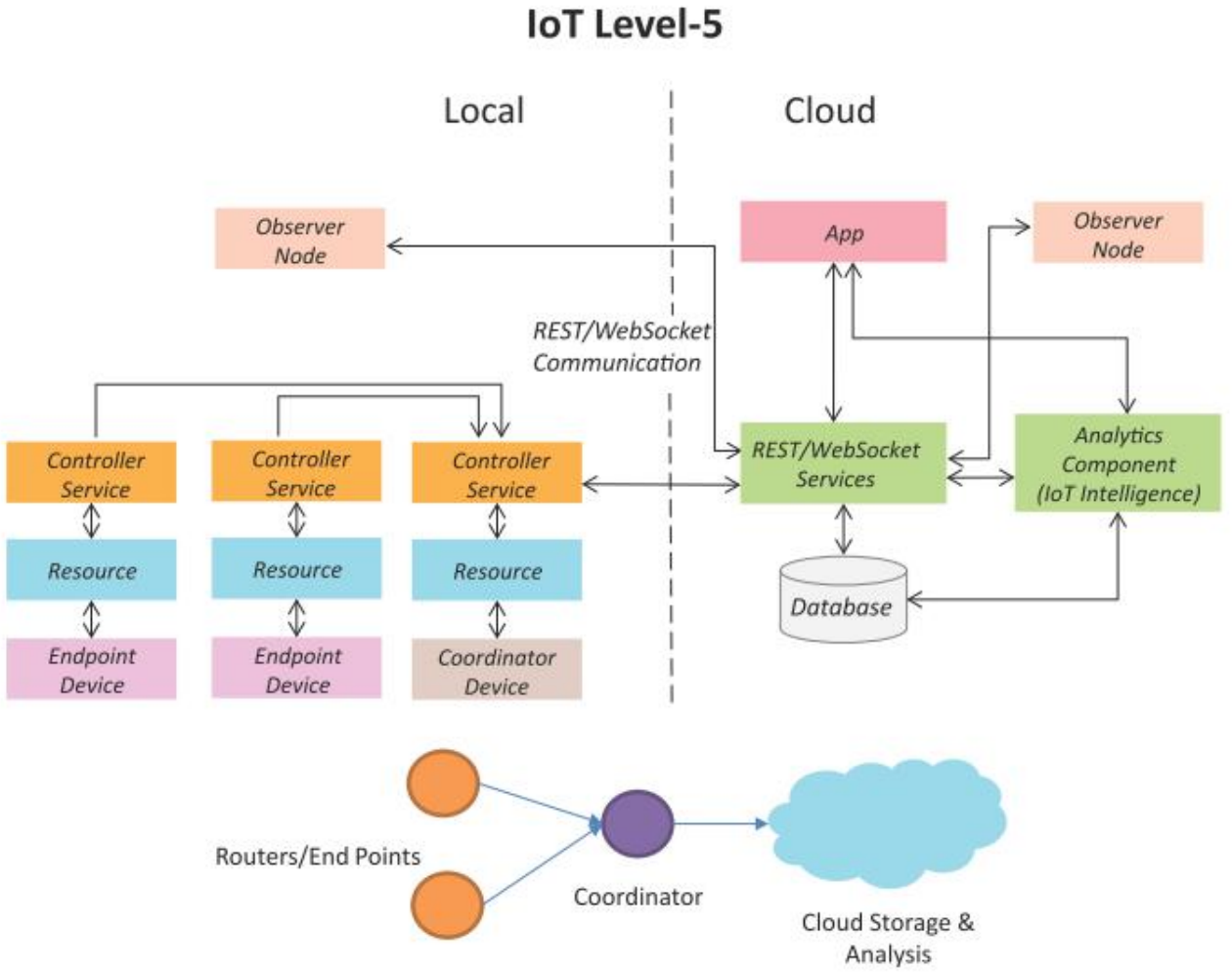
IoT Level-4

- A level-4 IoT system has multiple nodes that perform local analysis. Data is stored in the cloud and application is cloud-based.
- Level-4 contains local and cloud-based observer nodes which can subscribe to and receive information collected in the cloud from IoT devices.
- Level-4 IoT systems are suitable for solutions where multiple nodes are required, the data involved is big and the analysis requirements are computationally intensive.



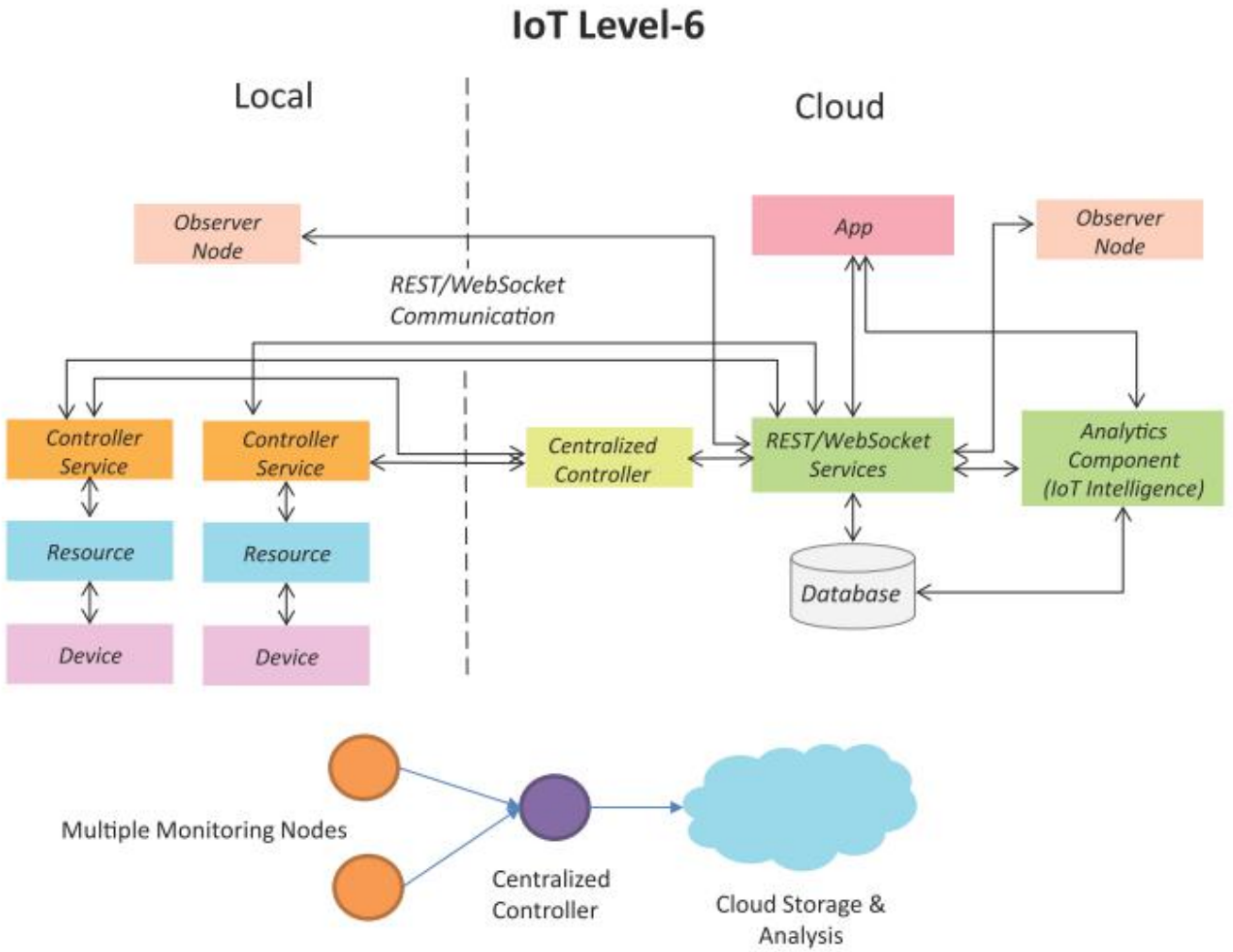
IoT Level-5

- A level-5 IoT system has multiple end nodes and one coordinator node.
- The end nodes that perform sensing and/or actuation.
- Coordinator node collects data from the end nodes and sends to the cloud.
- Data is stored and analyzed in the cloud and application is cloud-based.
- Level-5 IoT systems are suitable for solutions based on wireless sensor networks, in which the data involved is big and the analysis requirements are computationally intensive.



IoT Level-6

- A level-6 IoT system has multiple independent end nodes that perform sensing and/or actuation and send data to the cloud.
- Data is stored in the cloud and application is cloud-based.
- The analytics component analyzes the data and stores the results in the cloud database.
- The results are visualized with the cloud-based application.
- The centralized controller is aware of the status of all the end nodes and sends control commands to the nodes.



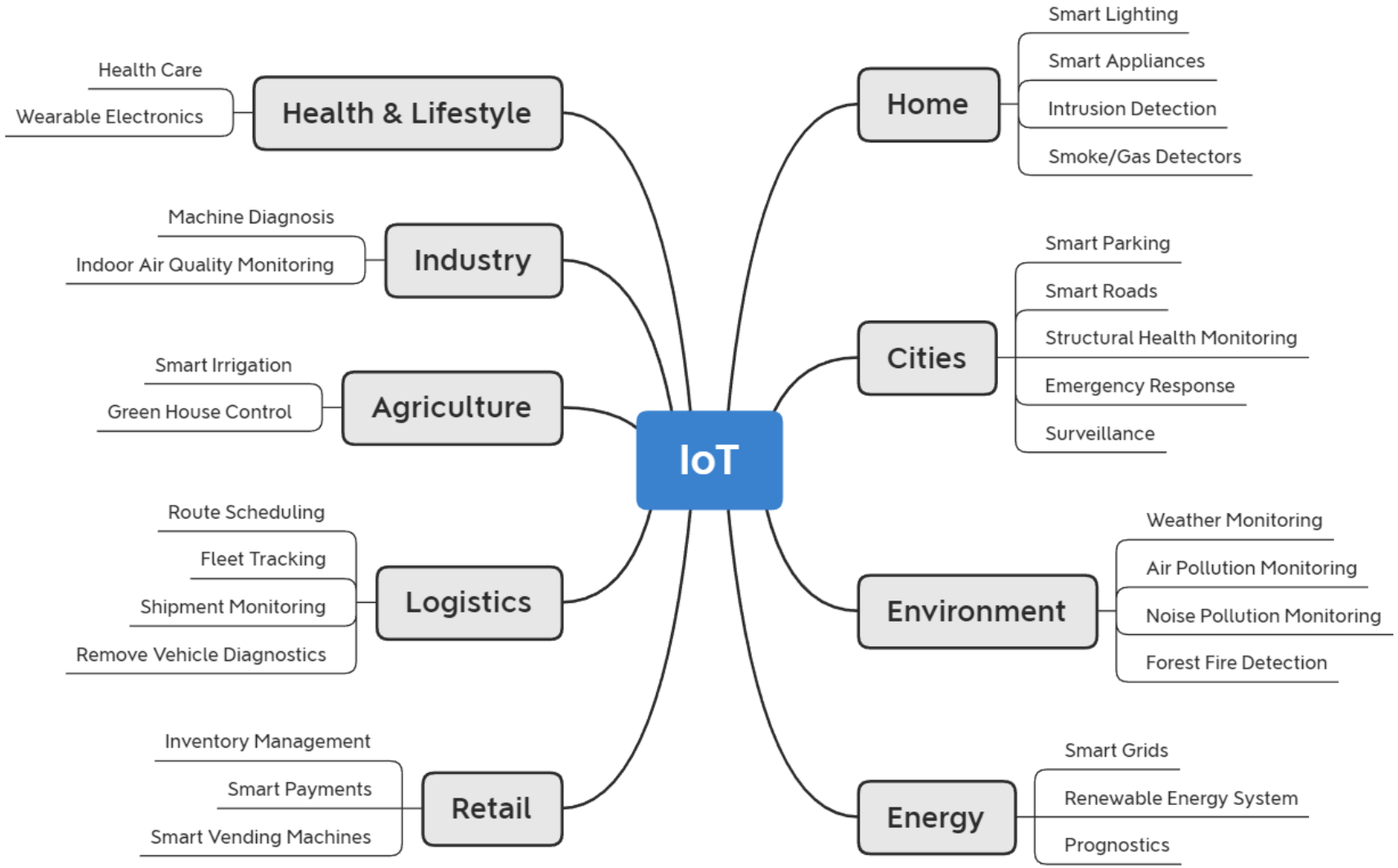
Chapter 2

Domain Specific IoTs

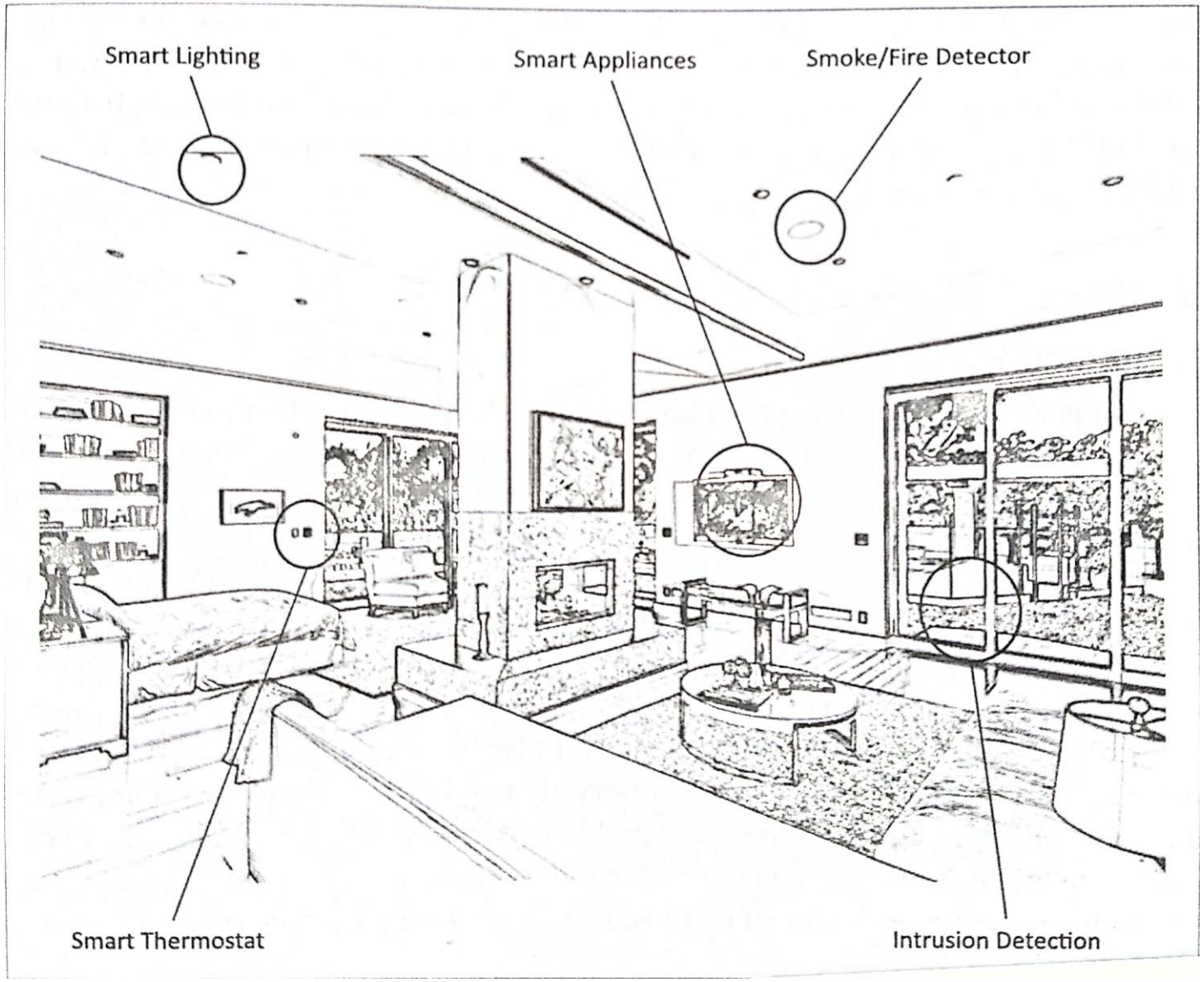
Outline

- Introduction
- Home Automation
- Cities
- Environment
- Energy
- Retail
- Logistics
- Agriculture
- Industry
- Health & Lifestyle

Introduction – Applications of IoT



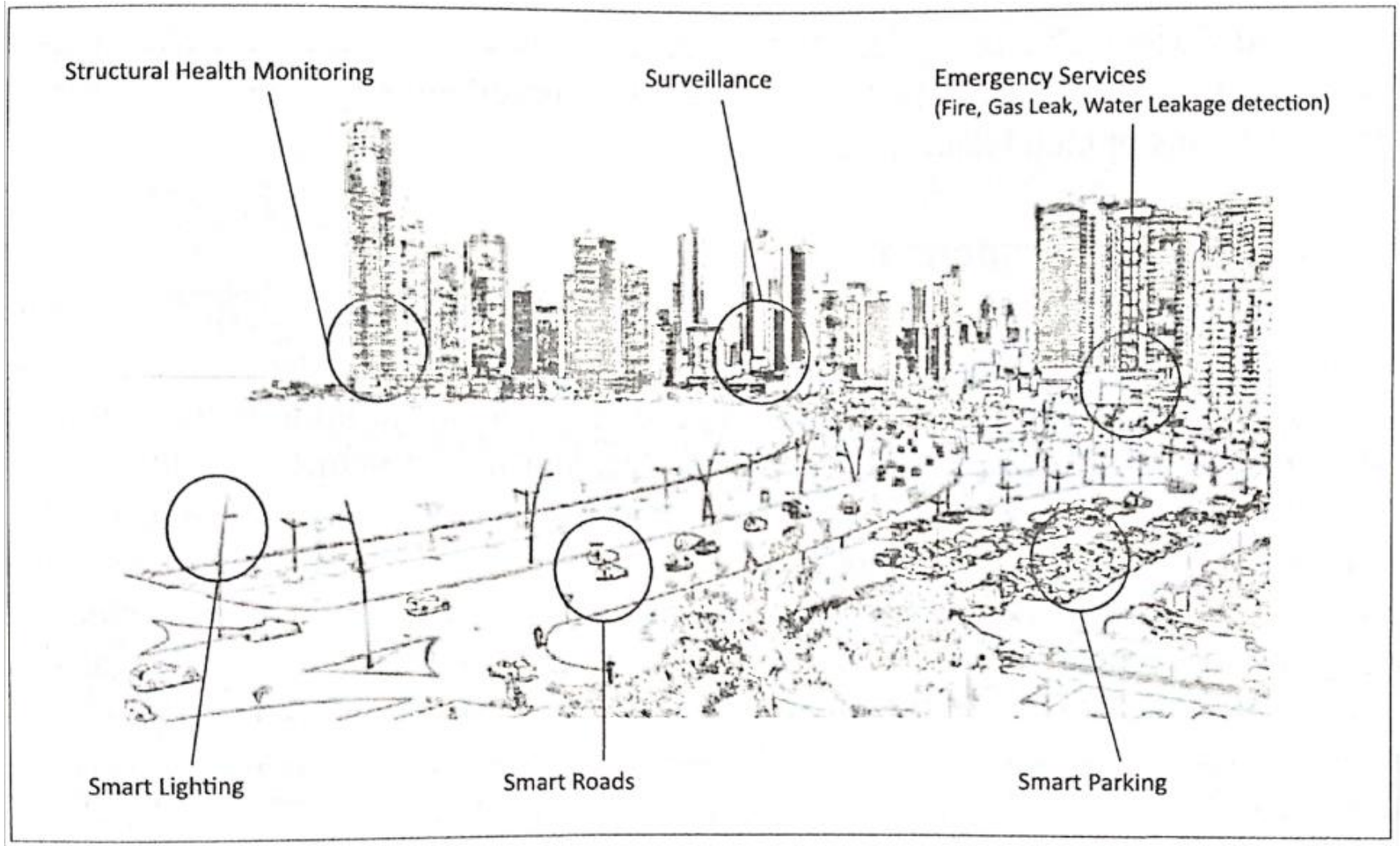
Home Automation



Home Automation (2/2)

- Smart Lighting
  - Control lighting by remotely (mobile or web applications)
- Smart Appliances
  - Provide status information to the users remotely
- Intrusion Detection
  - Use security cameras and sensors (PIR sensors and door sensors)
  - Detect intrusions and raise alerts
  - The alerts form: an SMS or an email sent to the user
- Smoke/Gas Detectors
  - Use optical detection, ionization, or air sampling techniques to detect the smoke
  - Gas detectors can detect harmful gases
    - Carbon monoxide (CO)
    - Liquid petroleum gas (LPG)
  - Raise alerts to the user or local fire safety department

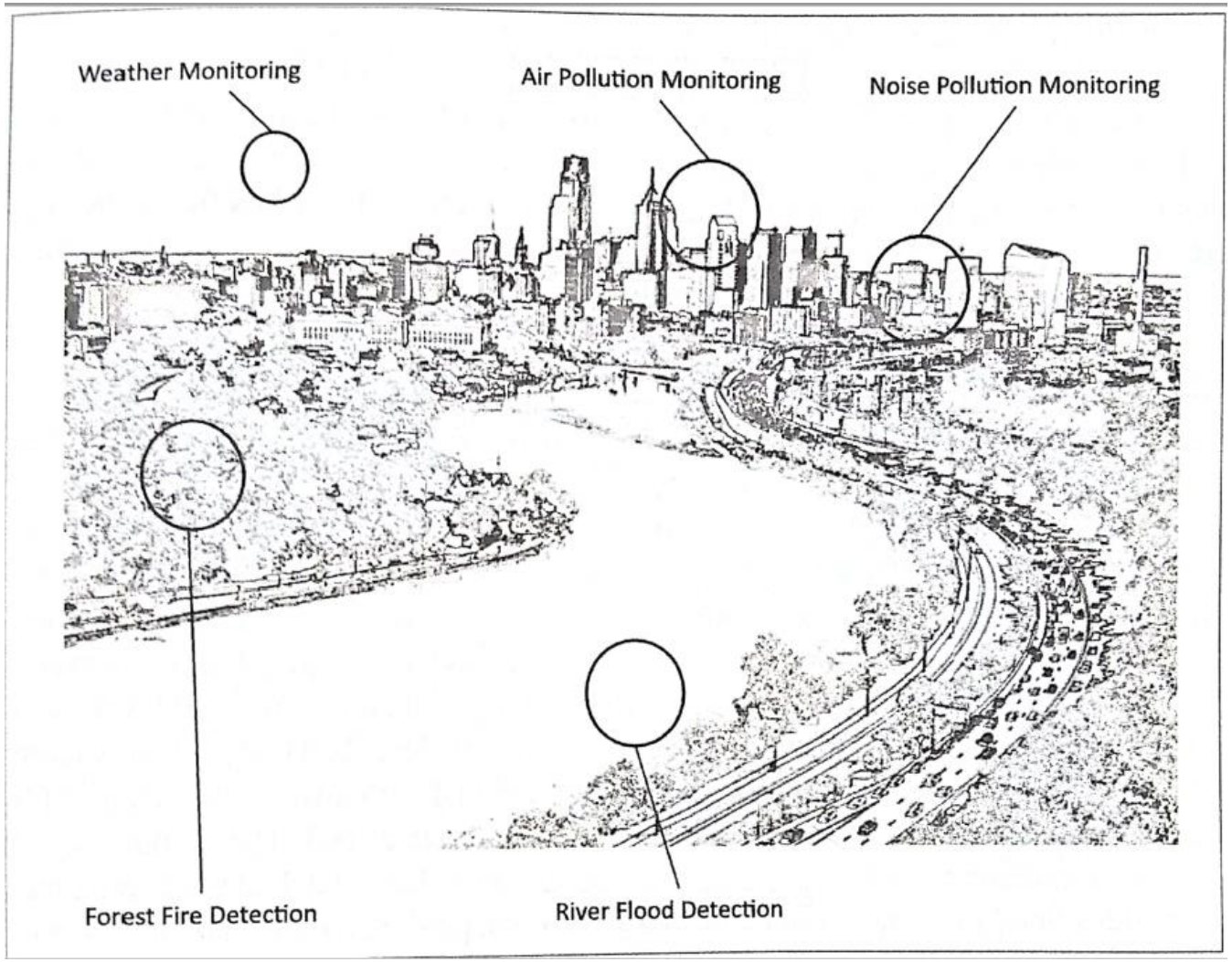
Cities (1/2)


Structural Health Monitoring
Surveillance
Emergency Services
(Fire, Gas Leak, Water Leakage detection)
Smart Lighting
Smart Roads
Smart Parking

Cities (2/2)

- Smart Parking
  - Detect the number of empty parking slots
  - Send the information over the internet and accessed by smartphones
- Smart Roads
  - Provide information on driving conditions, traffic congestions, accidents
  - Alert for poor driving conditions
- Structural Health Monitoring
  - Monitor the vibration levels in the structures (bridges and buildings)
  - Advance warning for imminent failure of the structure
- Surveillance
  - Use the large number of distributed and internet connected video surveillance cameras
  - Aggregate the video in cloud-based scalable storage solutions
- Emergency Response
  - Used for critical infrastructure monitoring
  - Detect adverse events

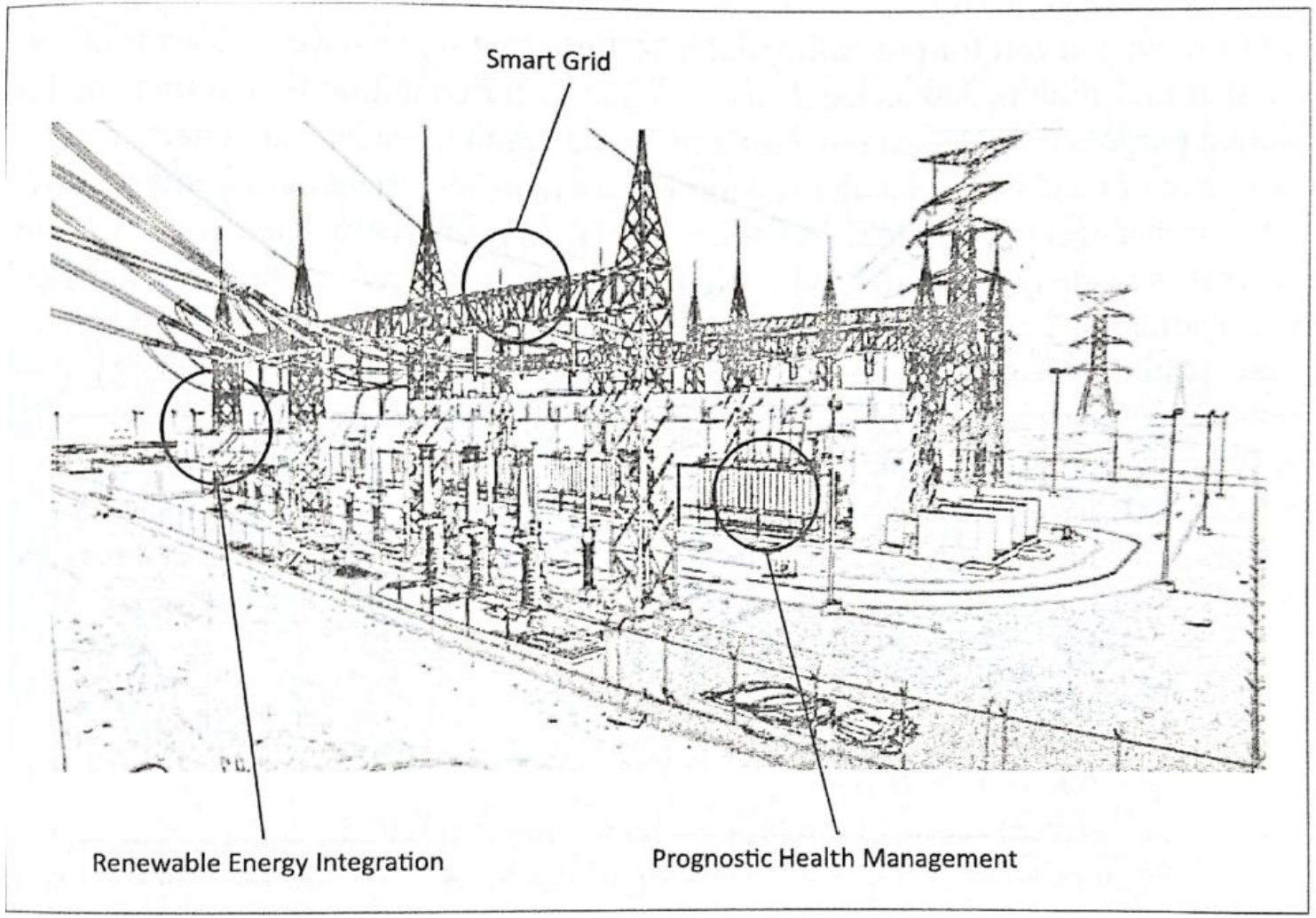
Environment (1/2)



Environment (2/2)

- Weather Monitoring
  - Collect data from several sensors (temperature, humidity, pressure, etc.)
  - Send the data to cloud-based applications and storage back-ends
- Air Pollution Monitoring
  - Monitor emission of harmful gases (CO_2, CO, NO, NO_2, etc.)
  - Factories and automobiles use gaseous and meteorological sensors
  - Integration with a single-chip microcontroller, several air pollution sensors, GPRS-modem, and a GPS module
- Noise Pollution Monitoring
  - Use a number of noise monitoring stations
  - Generate noise maps from data collected
- Forest Fire Detection
  - Use a number of monitoring nodes deployed at different locations in a forests
    - Use temperature, humidity, light levels, etc.
  - Provide early warning of potential forest fire
  - Estimates the scale and intensity
- River Floods Detection
  - Monitoring the water level (using ultrasonic sensors) and flow rate (using the flow velocity sensors)
  - Raise alerts when rapid increase in water level and flow rate is detected

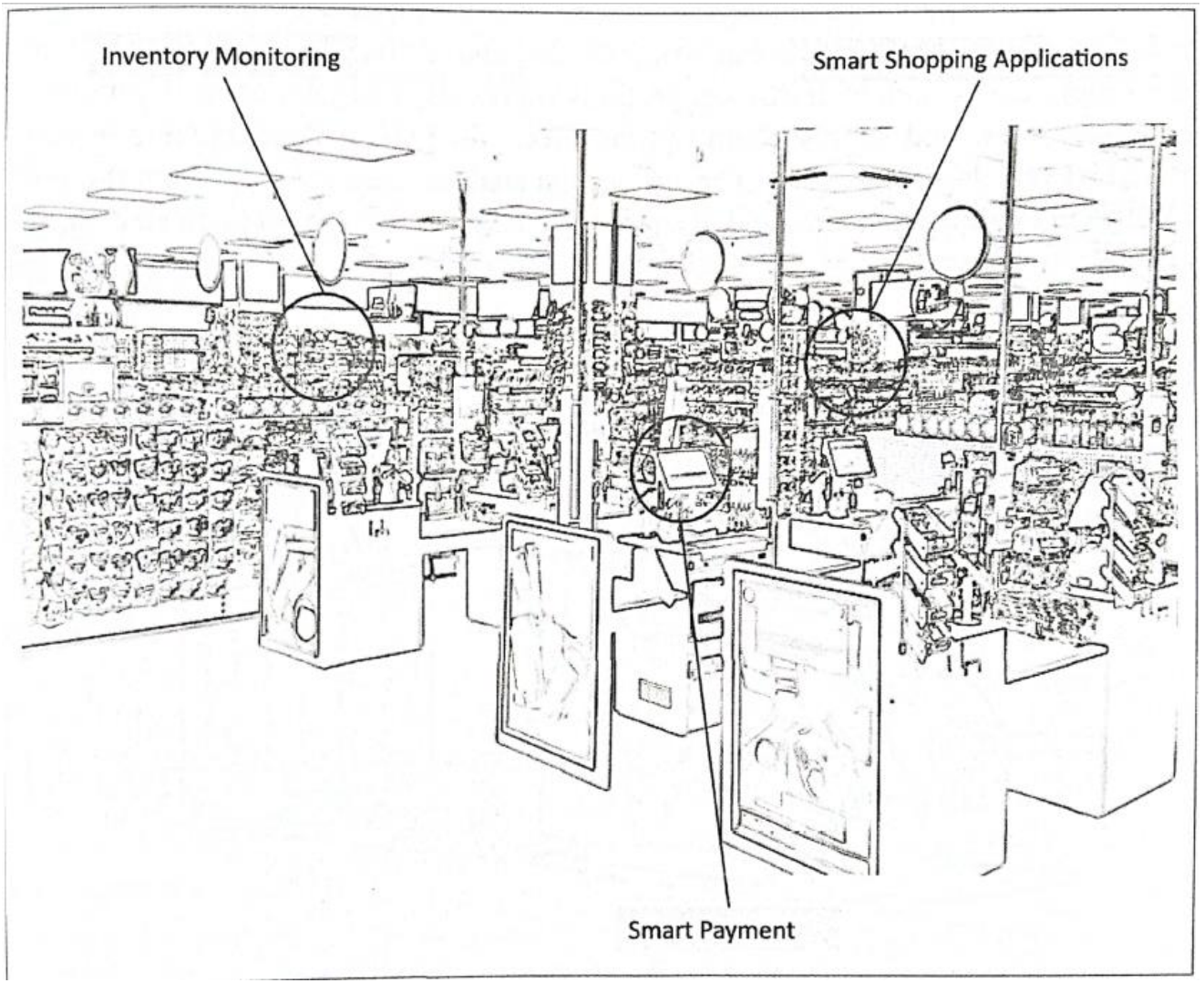
Energy (1/2)


Smart Grid
Renewable Energy Integration
Prognostic Health Management

Energy (2/2)

- Smart Grids
  - Collect data regarding electricity generation, consumption, storage (conversion of energy into other forms), distribution, equipment health data
  - Control the consumption of electricity
  - Remotely switch off supply
- Renewable Energy Systems
  - Measure the electrical variables
  - Measure how much the power is fed into the grid
- Prognostics
  - Predict performance of machines or energy systems
    - By collect and analyze the data from sensors

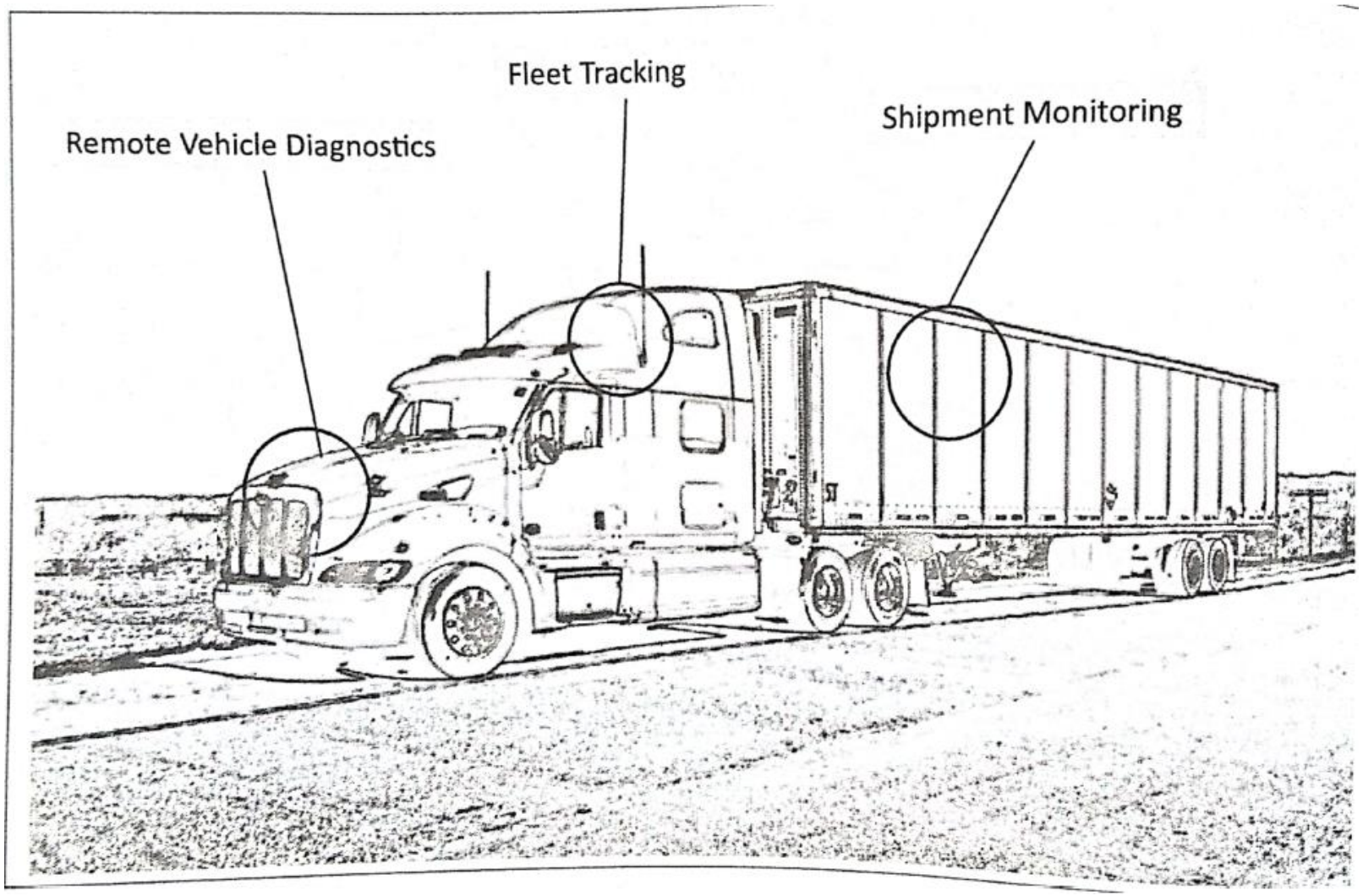
Retail (1/2)


Inventory Monitoring
Smart Shopping Applications
Smart Payment

Retail (2/2)

- Inventory Management
  - Monitoring the inventory by the RFID readers
  - Tracking the products
- Smart Payments
  - Use the NFC
    - Customers store the credit card information in their NFC-enabled
- Smart Vending Machines
  - Allow remote monitoring of inventory levels
  - Elastic pricing of products
  - Contact-less payment using NFC
  - Send the data to the cloud for predictive maintenance
    - The information of inventory levels
    - The information of the nearest machine in case a product goes out of stock in a machine

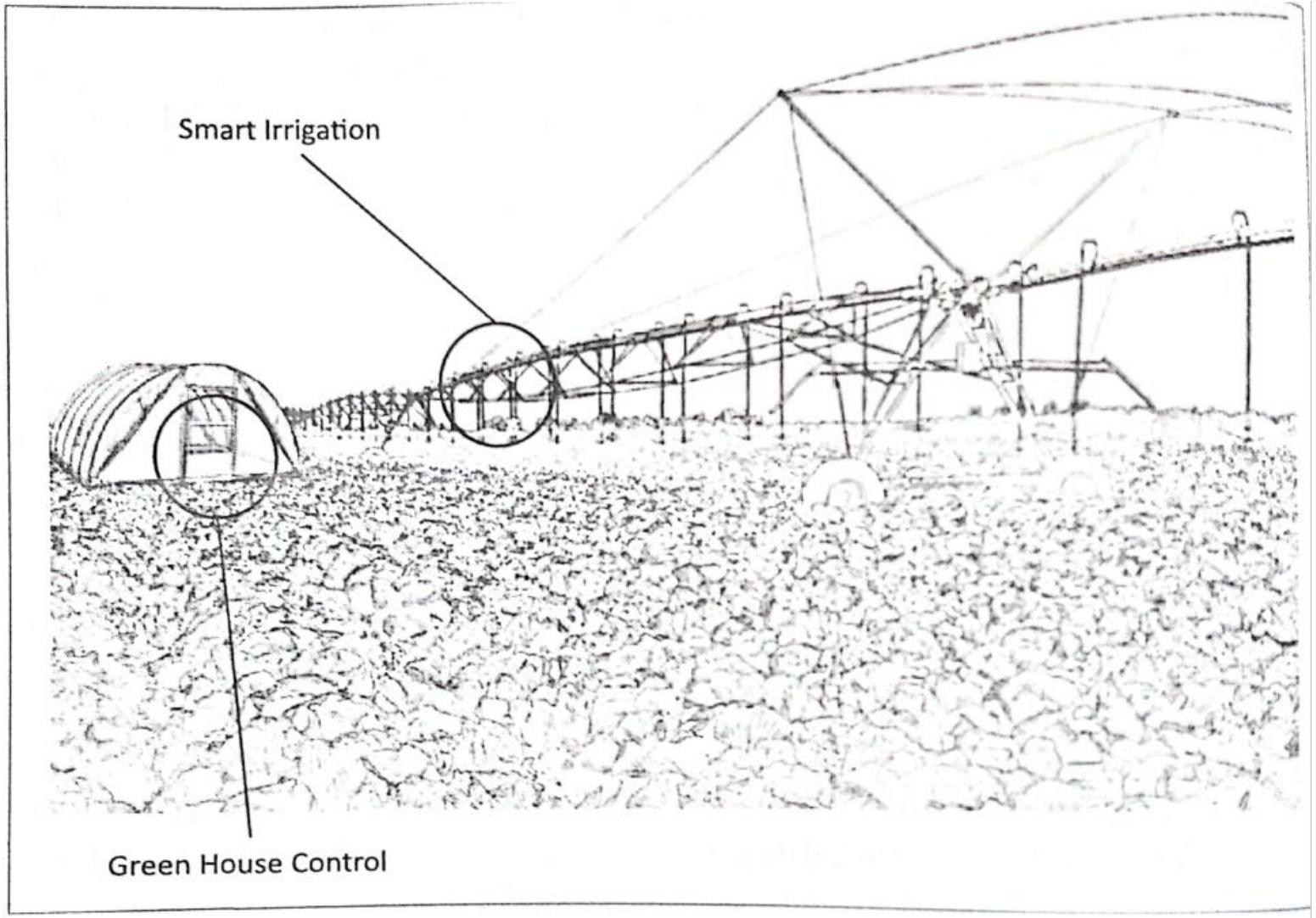
Logistics (1/2)



Logistics (2/2)

- Route Generation & Scheduling
  - Generate end-to-end routes using combination of route patterns
  - Provide route generation queries
  - Can be scale up to serve a large transportation network
- Fleet Tracking
  - Track the locations of the vehicles in real-time
  - Generate alerts for deviations in planned routes
- Shipment monitoring
  - Monitoring the conditions inside containers
  - Using sensors (temperature, pressure, humidity)
  - Detecting food spoilage
- Remote Vehicle Diagnostics
  - Detect faults in the vehicle
  - Warn of impending faults
  - IoT collects the data on vehicle (speed, engine RPM, coolant temperature)
  - Generate alerts and suggest remedial actions

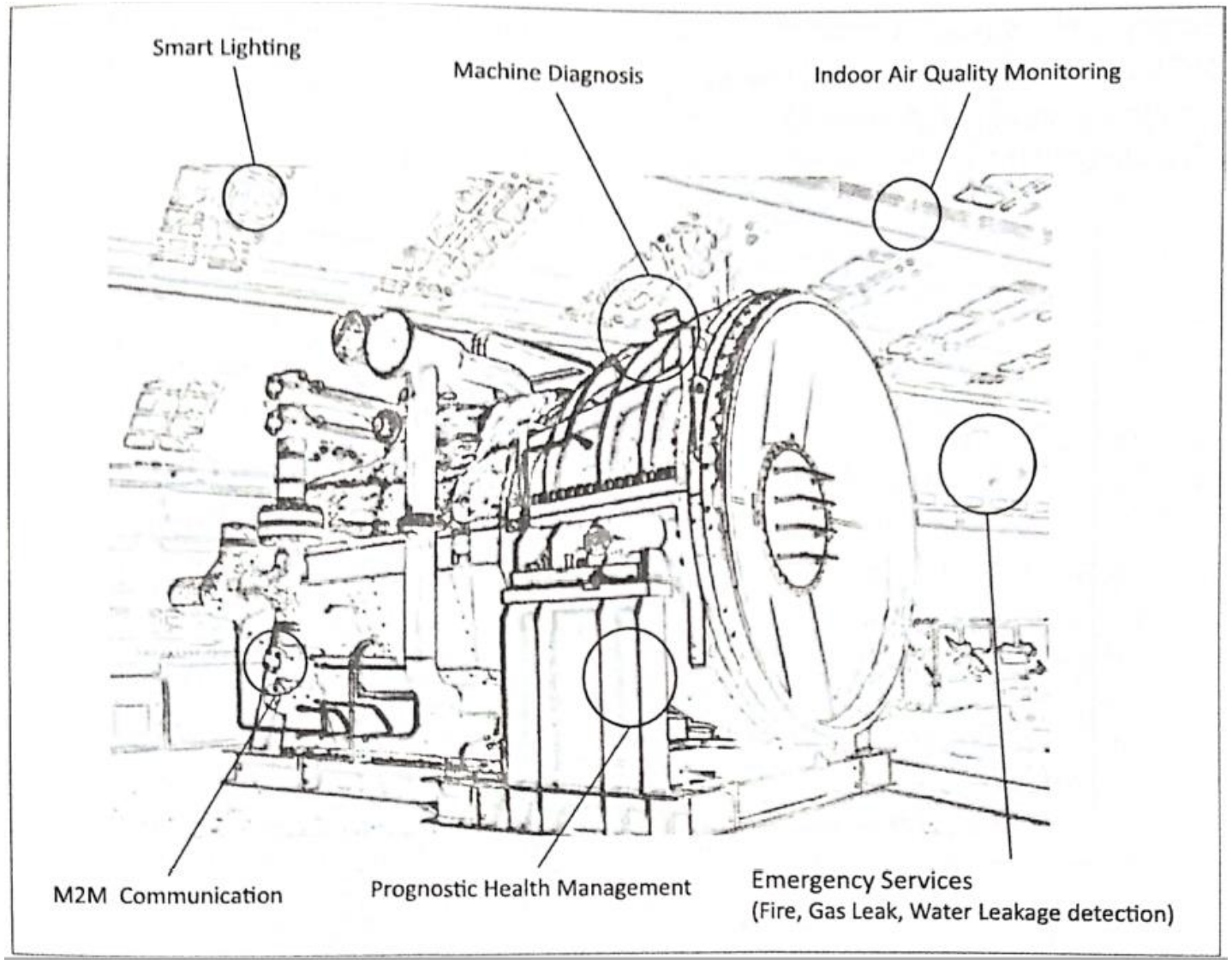
Agriculture (1/2)



Agriculture (2/2)

- Smart Irrigation
  - Use sensors to determine the amount of moisture in the soil
  - Release the flow of water
    - Using predefined moisture levels
  - Water Scheduling
- Green House Control
  - Automatically control the climatological conditions inside a green house
    - Using several sensors to monitor
    - Using actuation devices to control
      - Valves for releasing water and switches for controlling fans
  - Maintenance of agricultural production

Industry (1/2)


Smart Lighting
Machine Diagnosis
Indoor Air Quality Monitoring
M2M Communication
Prognostic Health Management
Emergency Services
(Fire, Gas Leak, Water Leakage detection)

Industry (2/2)

- Machine Diagnosis
  - Sensors in machine monitor the operating conditions
    - For example: temperature & vibration levels
  - Collecting and analyzing massive scale machine sensor data
    - For reliability analysis and fault prediction in machines
- Indoor Air Quality Monitoring
  - Use various gas sensors
    - To monitor the harmful and toxic gases (*CO*, *NO*, NO_2, etc.)
  - Measure the environmental parameters to determine the indoor air quality
    - Temperature, humidity, gaseous pollutants, aerosol

Health & Lifestyle

- Health & Fitness Monitoring
  - Collect the health-care data
    - Using some sensors: body temperature, heart rate, movement (with accelerometers), etc.
  - Various forms : belts and wrist-bands
- Wearable electronic
  - Assists the daily activities
    - Smart watch
    - Smart shoes
    - Smart wristbands

Module 2 Content

- Embedded Systems – Introduction, the basics of sensors and actuators, need for ADC & DAC peripherals.

- Introduction to Arduino, The Arduino UNO development board – architecture and specifications, the Arduino development environment, setting up the IDE, programming the Arduino, basic examples.

- Introduction to RaspberryPi – the RaspberryPi architecture & Hardware overview, RaspberryPi as a gateway device.

Embedded Systems

- Software + Hardware + RTOS (Real Time OS)

Characteristics of ES-

- Single functioned
- Tightly Coupled (cost, size, performance, size)
- Real time operations
- Microprocessors/Microcontrollers
- s/w (features & flexibility)
- h/w (performance & security)

Sensor and its types

- Active & Passive (based on inputs)
- Analog & Digital (based on outputs)
- Scalar & Vector (based on type of data collected)

Actuators and its types

- Electrical: converts electrical to mechanical energy Ex. Relays
- Pneumatic : with less pressure perform actions Ex. Pistons and ignition chamber in gasoline engines
- Hydraulic: converts hydro power to mechanical Ex. Pump, pistons
- Mechanical: any motion to linear motion Ex. Car stirring

What Is Arduino?

- Arduino is an open source programmable circuit board that can be integrated into a wide variety of makerspace projects both simple and complex.

- This board contains a microcontroller which is able to be programmed to sense and control objects in the physical world. By responding to sensors and inputs, the Arduino is able to interact with a large array of outputs such as LEDs, motors and displays.

- Because of it's flexibility and low cost, Arduino has become a very popular choice for makers and makerspaces looking to create interactive hardware projects.

What is the Arduino

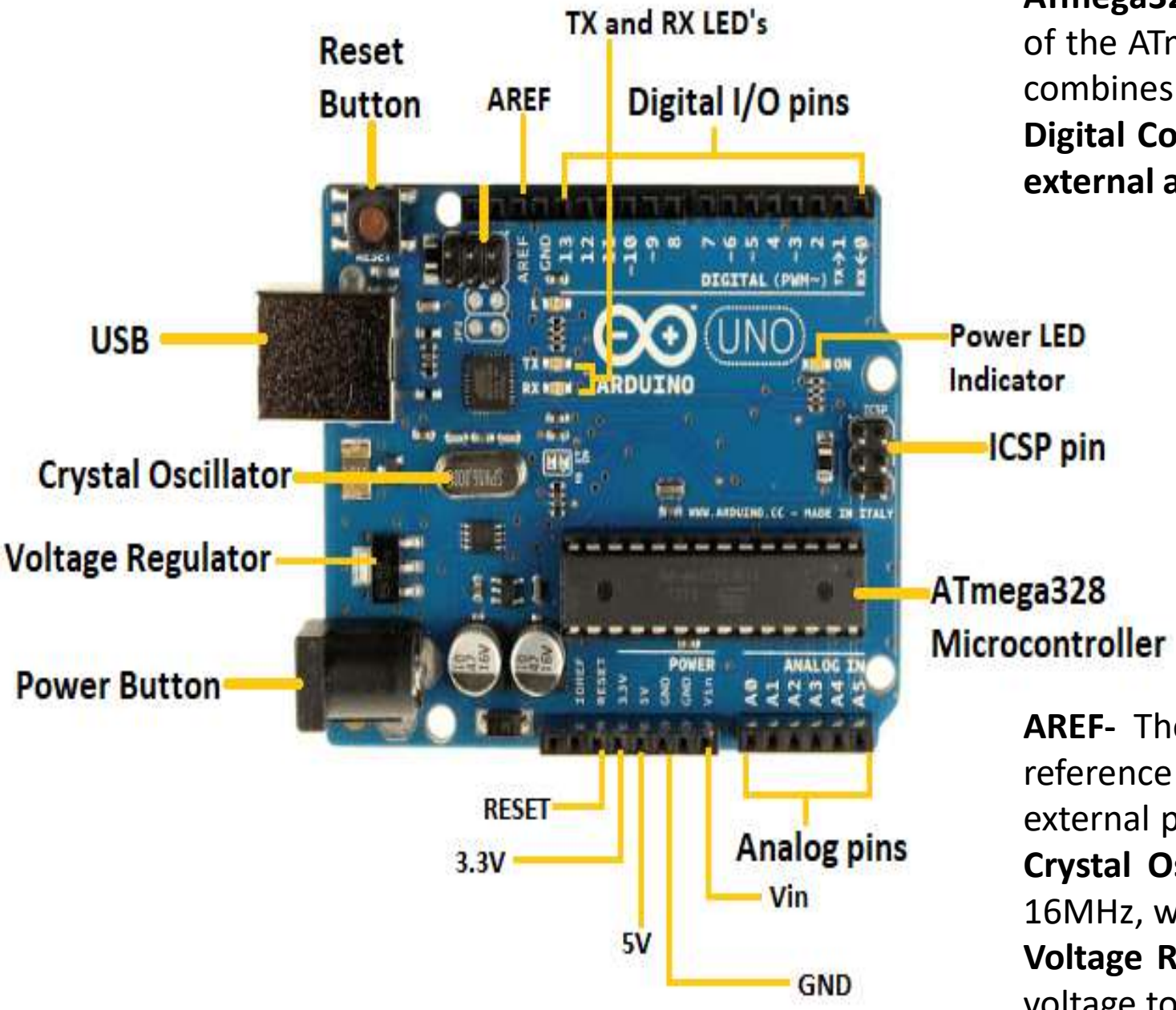
The word "Arduino" can mean 3 things

A physical piece of hardware

A programming environment

A community & philosophy

todbot.com/blog/bionicarduino



Arduino UNO

ATmega328 Microcontroller- is a single chip Microcontroller of the ATmel family. The processor code inside it is of 8-bit. It combines **Memory (SRAM, EEPROM, and Flash), Analog to Digital Converter, SPI serial ports, I/O lines, registers, timer, external and internal interrupts, and oscillator.**

ICSP pin - The In-Circuit Serial Programming pin allows the user to program using the firmware of the Arduino board.

Digital I/O pins- The digital pins D0 - D13 have the value HIGH or LOW.

Analog Pins- The pins A0 to A5 are analog pins.

AREF- The Analog Reference (AREF) pin is used to feed a reference voltage to the Arduino UNO board from the external power supply.

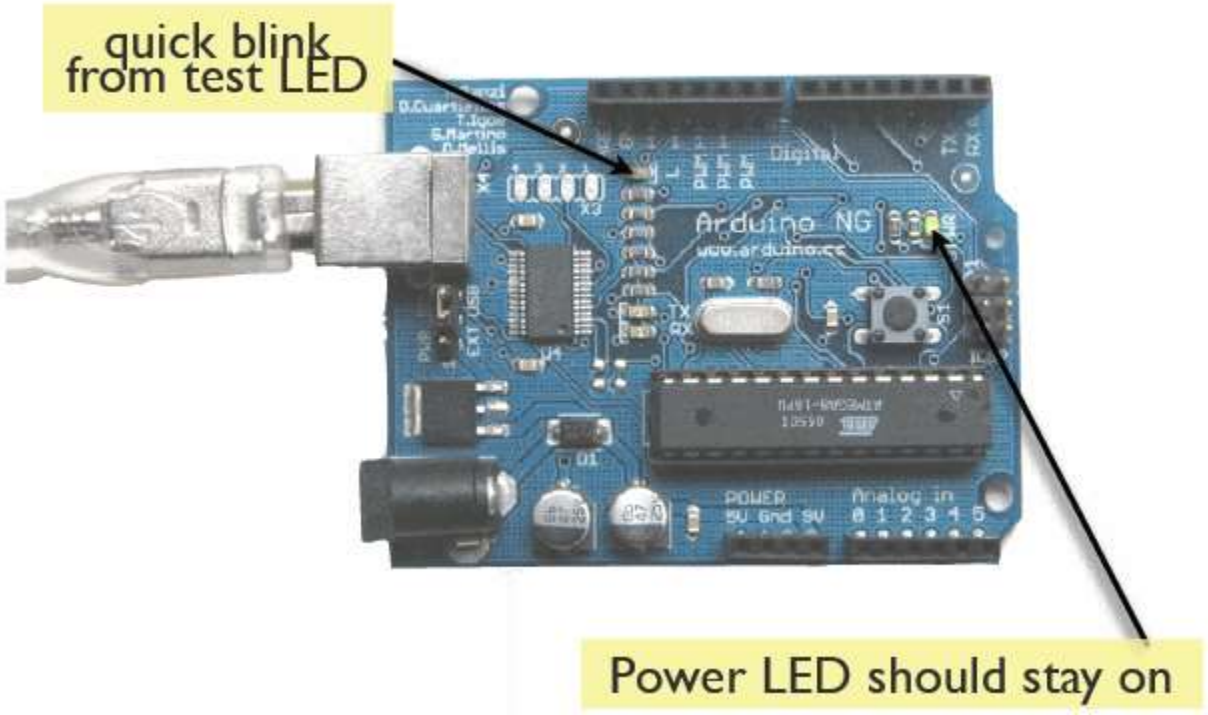
Crystal Oscillator- The Crystal oscillator has a frequency of 16MHz, which makes the Arduino UNO a powerful board.

Voltage Regulator- The voltage regulator converts the input voltage to 5V.

Getting Started

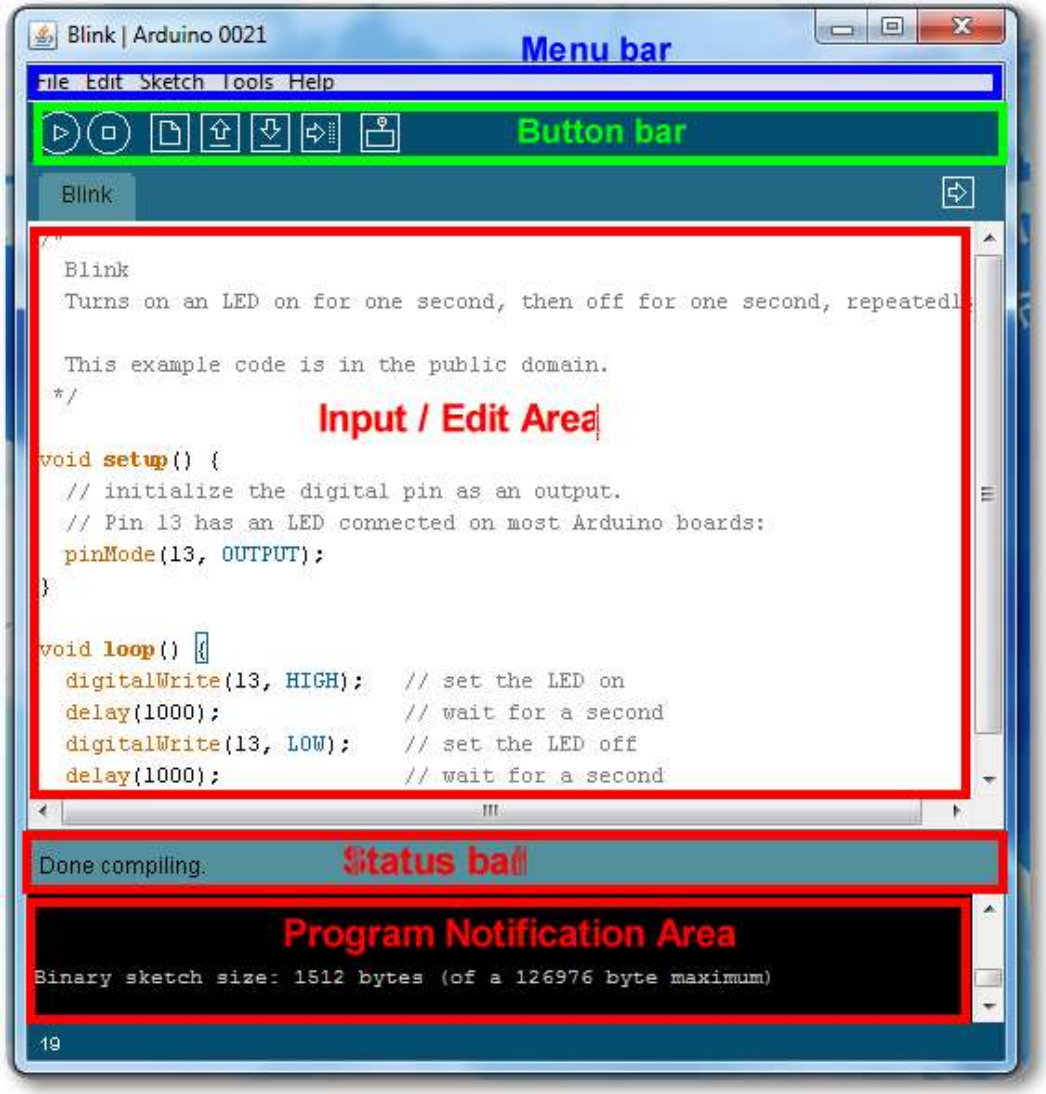
- Check out: http://arduino.cc/en/Guide/HomePage
  1. **Download & install the Arduino environment (IDE)**
  2. **Connect the board to your computer via the UBS cable**
  3. **If needed, install the drivers (not needed in lab)**
  4. **Launch the Arduino IDE**
  5. **Select your board**
  6. **Select your serial port**
  7. **Open the blink example**
  8. **Upload the program**

Try It: Connect the USB Cable



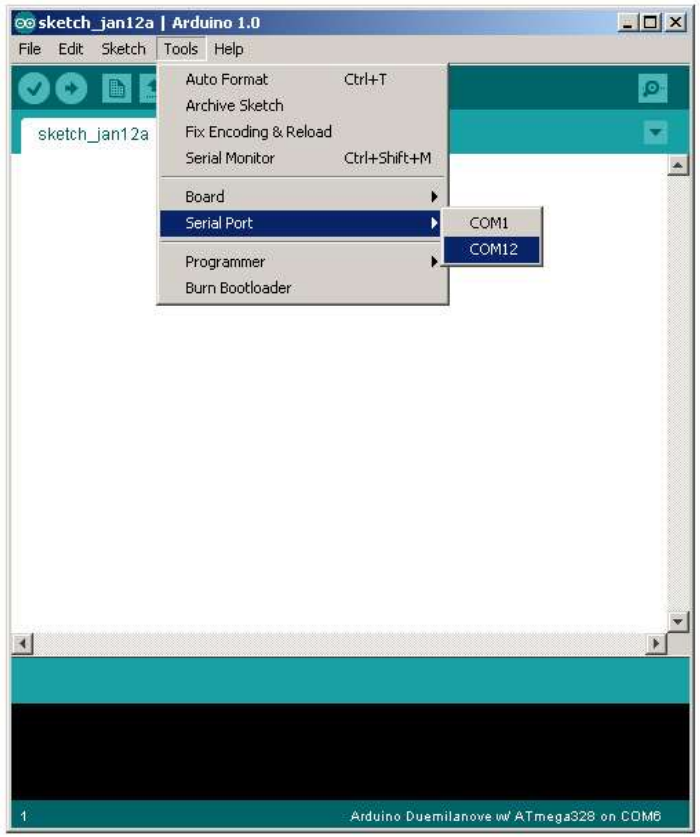
todbot.com/blog/bionicarduino

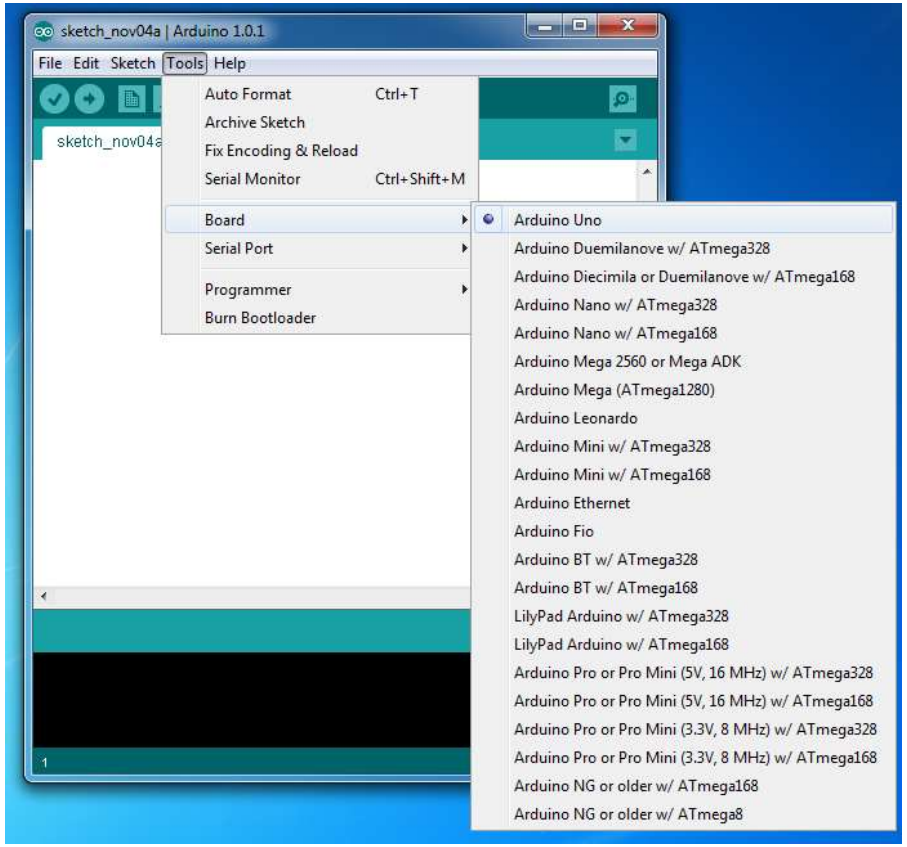
Arduino IDE



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

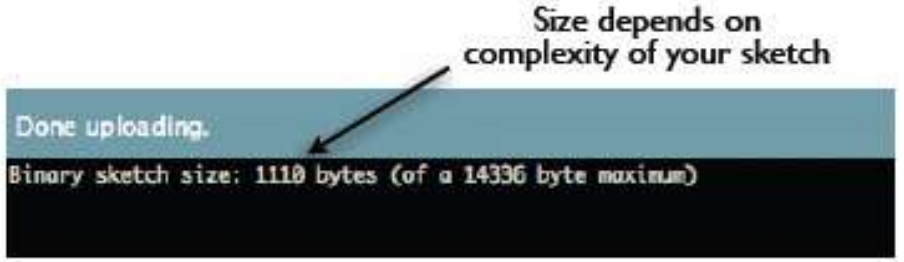
Select Serial Port and Board





Status Messages

Uploading worked



Wrong serial port selected

Wrong board selected

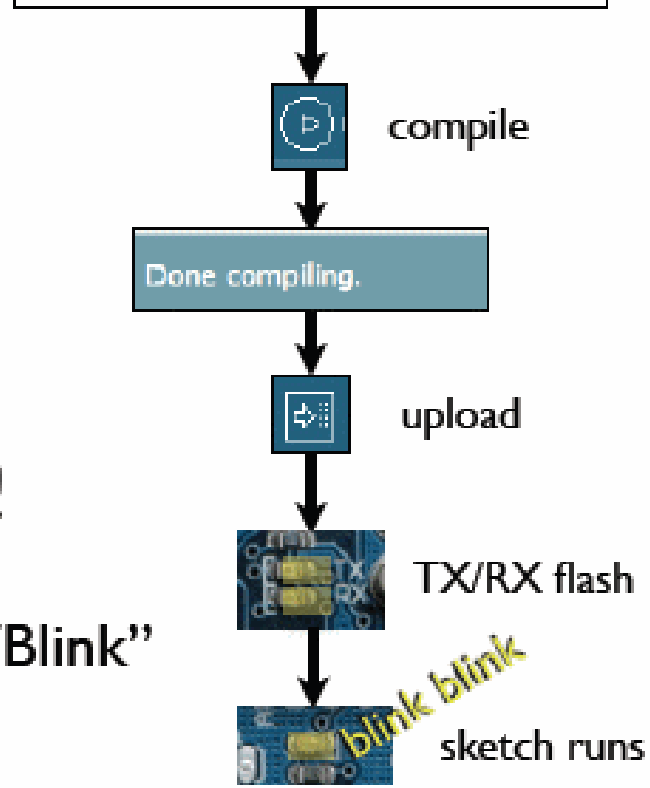
nerdy cryptic error messages

todbot.com/blog/bionicarduino

Using Arduino

- Write your sketch
- Press Compile button
  (to check for errors)
- Press Upload button to program
  Arduino board with your sketch

```
void setup() {
  pinMode(ledPin, OUTPUT);      // sets t
}
void loop() {
  digitalWrite(ledPin, HIGH);   // sets t
  delay(1000);                  // waits
  digitalWrite(ledPin, LOW);    // sets t
  delay(1000);                  // waits
}
```



Try it out with the "Blink" sketch!

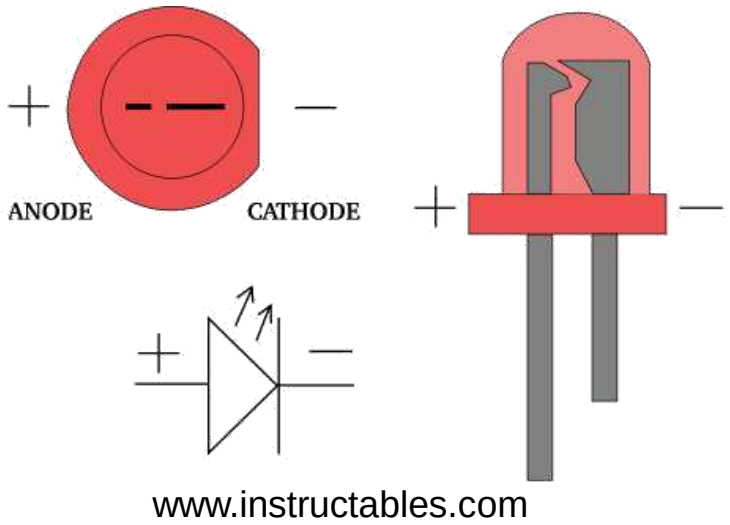
Load "File/Sketchbook/Examples/Digital/Blink"

A Little Bit About Programming

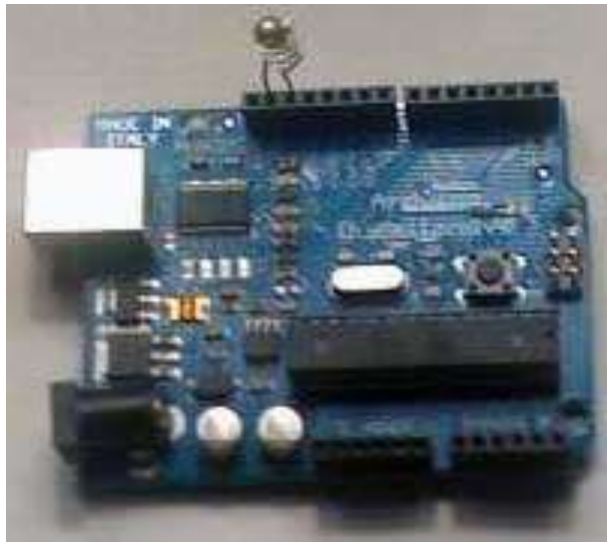
- Code is case sensitive
- Statements are commands and must end with a semi-colon
- Comments follow a // or begin with /* and end with */
- loop and setup

Add an External LED to pin 13

- **File > Examples > Digital > Blink**
- LED's have polarity
  - Negative indicated by flat side of the housing and a short leg



www.instructables.com



Terminology

"*sketch*" – a program you write to run on an Arduino board

"*pin*" – an input or output connected to something.

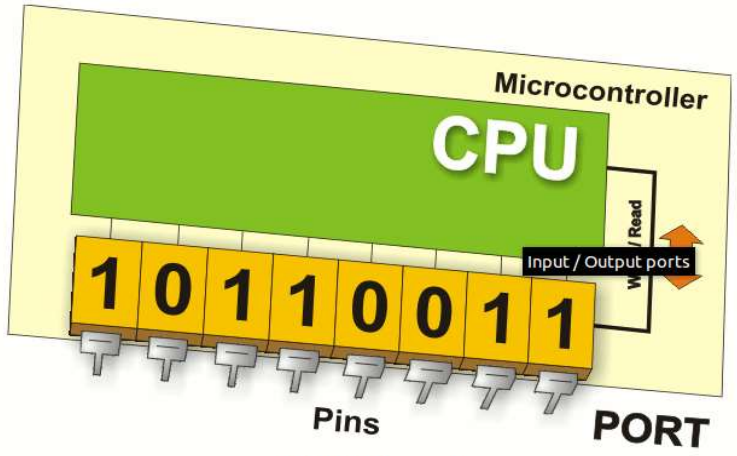
e.g. output to an LED, input from a knob.

"*digital*" – value is either HIGH or LOW.

(aka on/off, one/zero) e.g. switch state

"*analog*" – value ranges, usually from 0-255.

e.g. LED brightness, motor speed, etc.



www.mikroe.com/chapters/view/1

Digital I/O

`pinMode(`*`pin`*`, `*`mode`*`)`

Sets pin to either `INPUT` or `OUTPUT`

`digitalRead(`*`pin`*`)`

Reads `HIGH` or `LOW` from a pin

`digitalWrite(`*`pin`*`, `*`value`*`)`

Writes `HIGH` or `LOW` to a pin

Electronic stuff

Output pins can provide 40 mA of current

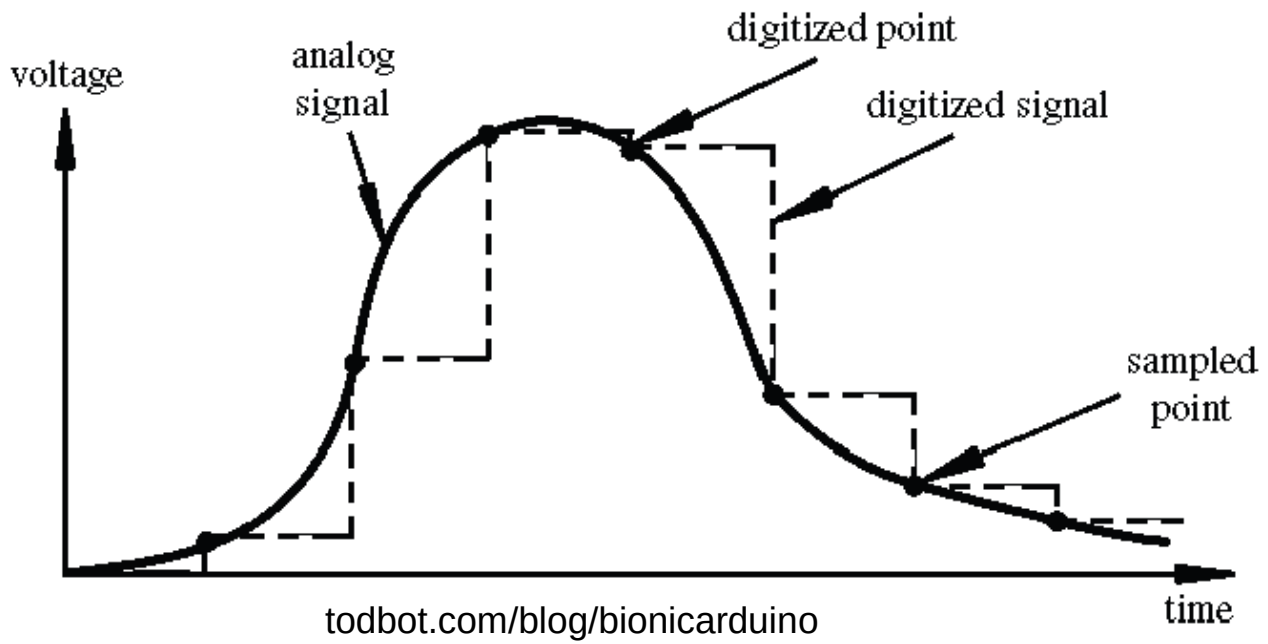
Writing `HIGH` to an input pin installs a 20KΩ pullup

Arduino Timing

- `delay(`*`ms`*`)`
  - Pauses for a few milliseconds
- `delayMicroseconds(`*`us`*`)`
  - Pauses for a few microseconds
- More commands: arduino.cc/en/Reference/HomePage

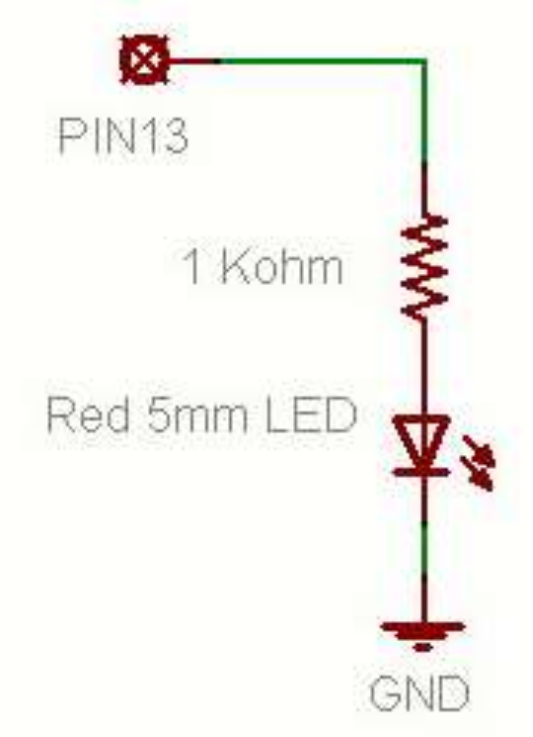
Digital? Analog?

- Digital has two values: **on** and **off**
- Analog has many (infinite) values
- Computers don't really do analog, they ***quantize***
- Remember the 6 analog input pins---here's how they work

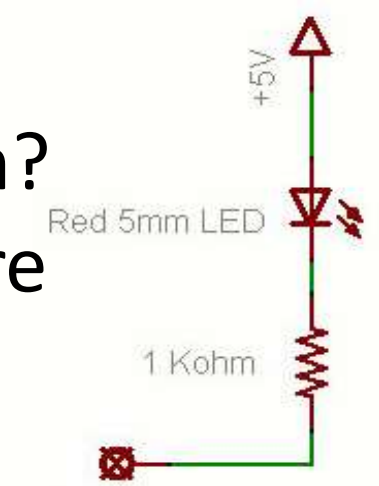


todbot.com/blog/bionicarduino

Putting It Together



- Complete the sketch (program) below.
- What output will be generated by this program?
- What if the schematic were changed? ➔




```
void loop()                     // run over and over again
{
  digitalWrite(ledPin, HIGH);   // sets the LED on
  delay(500);                  // waits for a second
  digitalWrite(ledPin, LOW);    // sets the LED off
  delay(500);                  // waits for a second
}
```

How To Program Arduino

Once the circuit has been created on the breadboard, you'll need to upload the program (known as a sketch) to the Arduino.

The sketch is a set of instructions that tells the board what functions it needs to perform.

An Arduino board can only hold and perform one sketch at a time.

The software used to create Arduino sketches is called the IDE which stands for Integrated Development Environment.

Arduino Programming Language

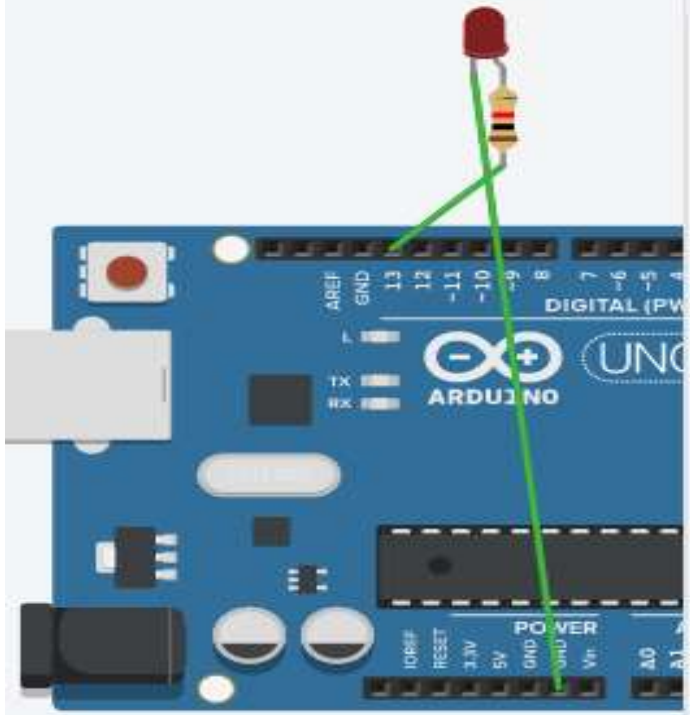
Arduino programming language can be divided in three main parts:

- FUNCTIONS - For controlling the Arduino board and performing computations.
- VARIABLES - Arduino data types and constants.
- STRUCTURE - The elements of Arduino (C++) code.

Every Arduino sketch has two main parts to the program:

void setup() – Sets things up that have to be done once and then don't happen again.

void loop() – Contains the instructions that get repeated over and over until the board is turned off.



Code | Start Simulation | Exp

Text | 1 (Ardu

```
1  void setup()
2  {
3    pinMode(13, OUTPUT);
4  }
5
6  void loop()
7  {
8    digitalWrite(13, HIGH);
9    delay(1000); // Wait for 1000 millisecond(s)
10   digitalWrite(13, LOW);
11   delay(1000); // Wait for 1000 millisecond(s)
12 }
```

Serial Monitor

Serial Library

Used for communication between the Arduino board and a computer or other devices. All Arduino boards have at least one serial port (also known as a UART or USART), and some have several.

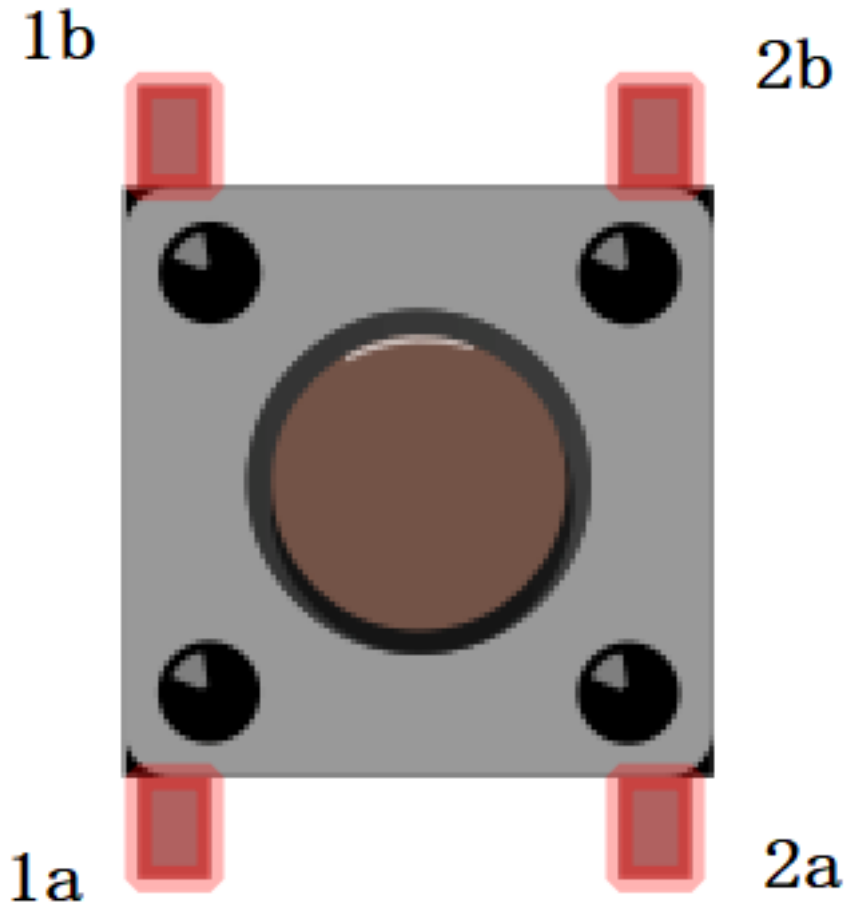
As the name implies serial communication means sending and receiving data bit by bit over a single line. Arduino uno board has one serial port at digital pins 0(RX) and 1(TX) to communicate with other external serial devices or with computer through USB cable.

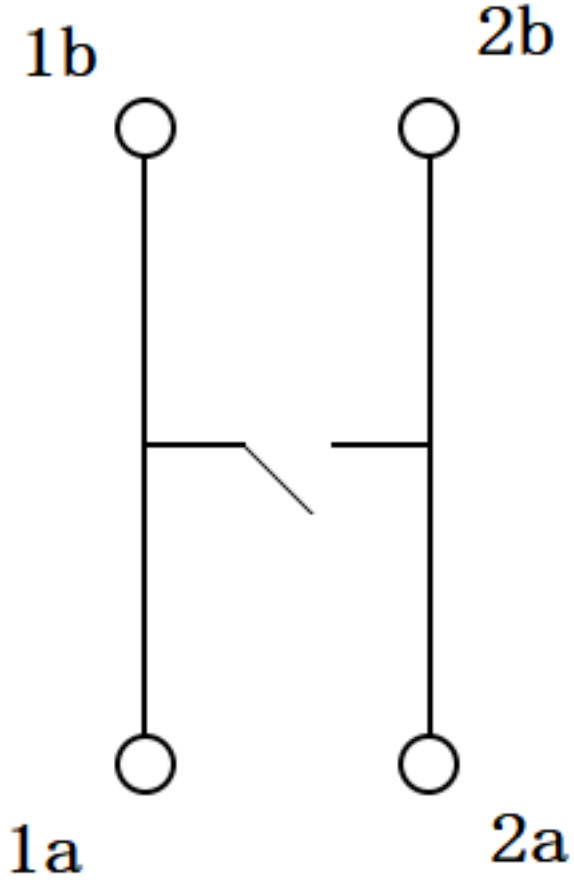
The process of sending and receiving data can be observed by flashing of TX and RX LED's on the arduino board.

The baud rate specifies how fast the data is sent over the serial line or in simple terms, the speed of serial communication. Some common rates for UART are 9600 baud, 11520 baud etc.

Serial.begin(9600) - opens the serial communication between the arduino and the computer at baud rate 9600bps.
Serial.available() - return the number of bytes that are currently present in the arduino serial buffer.
Serial.println() - this function prints data to the serial port to which arduino is connected.
Serial.read() – will read the serial data(ASCII value) from the key board

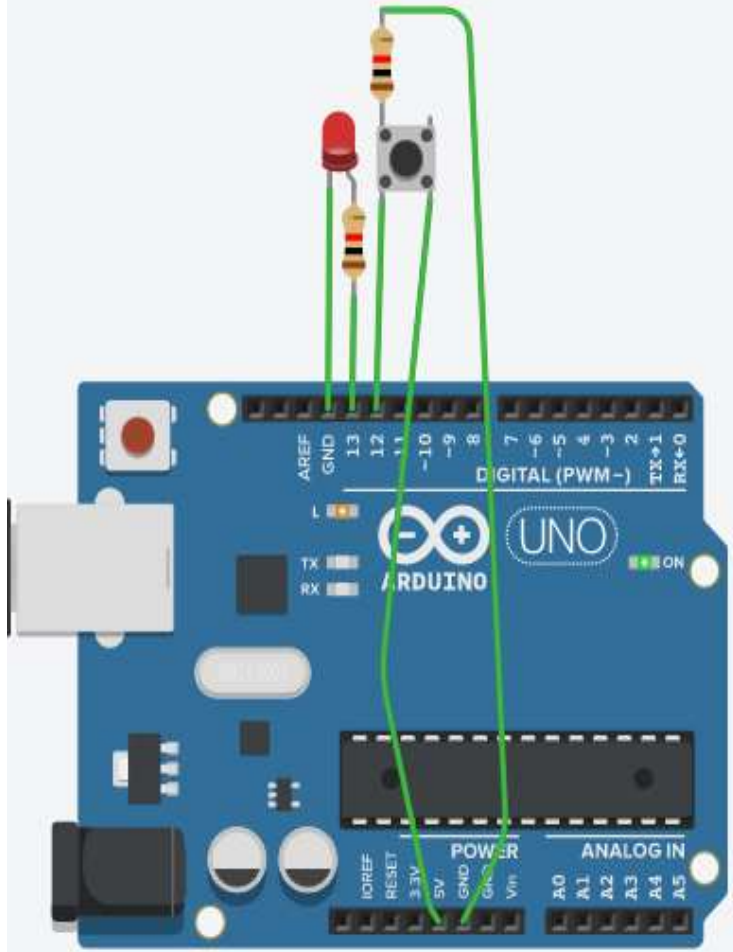
push button





Simulator time: 00:00:12

Code | Stop Simulation | Export | Share



Text

1 (Arduino Uno R3)

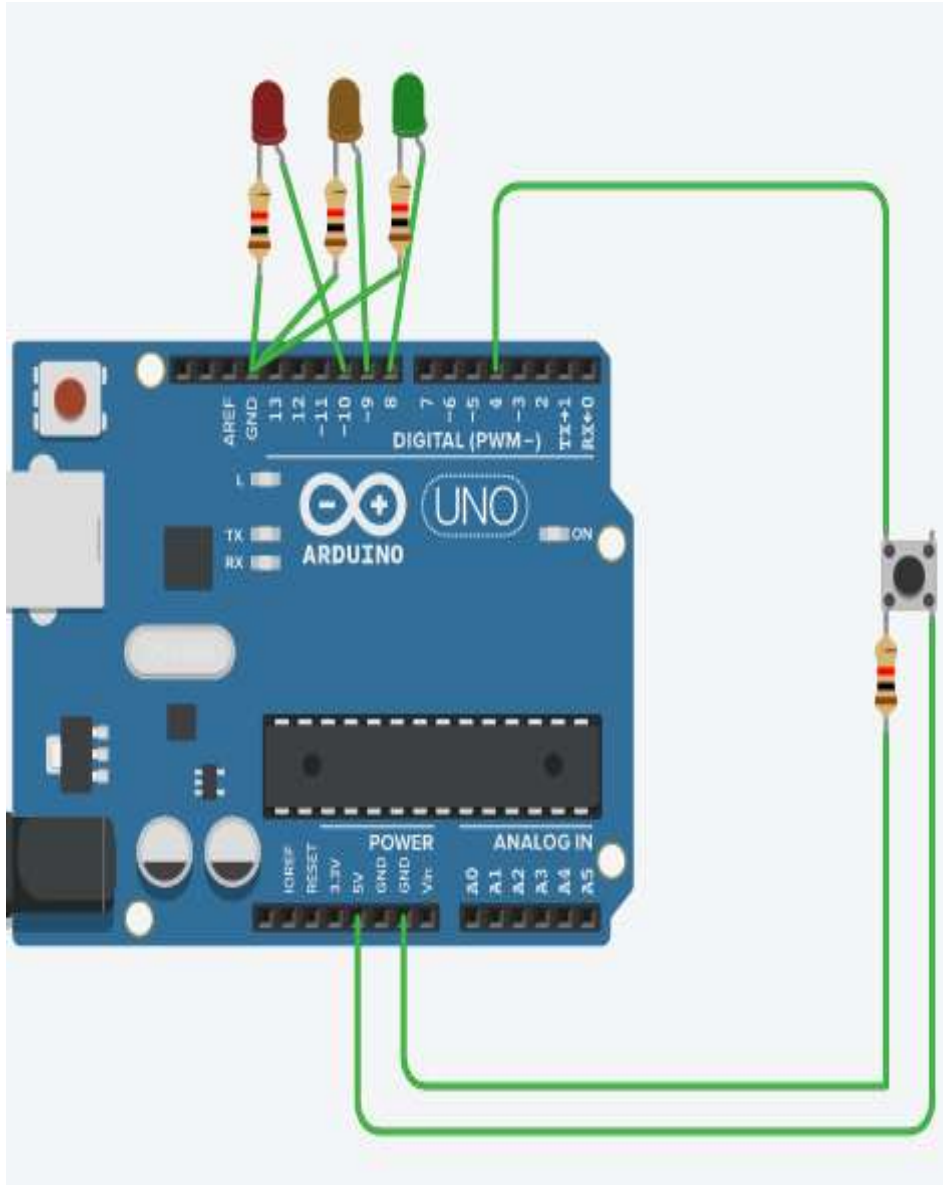
```
1  void setup()
2  {
3    pinMode(13, OUTPUT);
4
5    pinMode(12, INPUT);
6    Serial.begin(9600);
7  }
8
9  void loop()
10 {
11  if(digitalRead(12)==1){
12
13    digitalWrite(13, HIGH);
14     Serial.println("ON");
15    delay(1000); // Wait for 1000 millisecond(s)
16  }
17    digitalWrite(13, LOW);
18    delay(1000); // Wait for 1000 millisecond(s)
19    Serial.println("OFF");
20
21 }
```

Serial Monitor

```
OFF
ON
OFF
OFF
ON
```

Send | Clear

LED Rotation



Text

1 (Arduino Uno R3)

```
1  int count=1;
2  void setup()
3  {
4    pinMode(4, INPUT);      pinMode(8, OUTPUT);
5    pinMode(9, OUTPUT);     pinMode(10, OUTPUT);
6  }
7
8  void loop()
9  {
10    int d=digitalRead(4);
11
12   if(d==HIGH && count==1){
13     digitalWrite(10,HIGH);  count=2;
14   }else if(d==HIGH && count==2){
15     digitalWrite(10,LOW);   digitalWrite(9,HIGH);
16     count=3;
17   }else if(d==HIGH && count==3){
18     digitalWrite(9,LOW);    digitalWrite(8,HIGH);
19     count=4;
20   }else if(d==HIGH && count==4){
21      digitalWrite(10,LOW); digitalWrite(9,LOW); digitalWrite(8,LOW
22     count=1;
23   }
24   delay(200);
25
26 }
```

A **potentiometer** is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider.

A variable resistor with a third adjustable terminal. The potential at the third terminal can be adjusted to give any fraction of the potential across the ends of the resistor.

An **LDR (**Light Dependent Resistor**)** is a component that has a (variable) resistance that changes with the light intensity that falls upon it. This allows them to be used in light sensing circuits.

map()

Re-maps a number from one range to another.

Arduino has an analogRead range from 0 to 1023, and an analogWrite range only from 0 to 255, therefore the data from the potentiometer needs to be converted to fit into the smaller range before using it to dim the LED.

Example:

```
int var=analogRead(A0);
int val=map(var,0,1024,0,256);
```

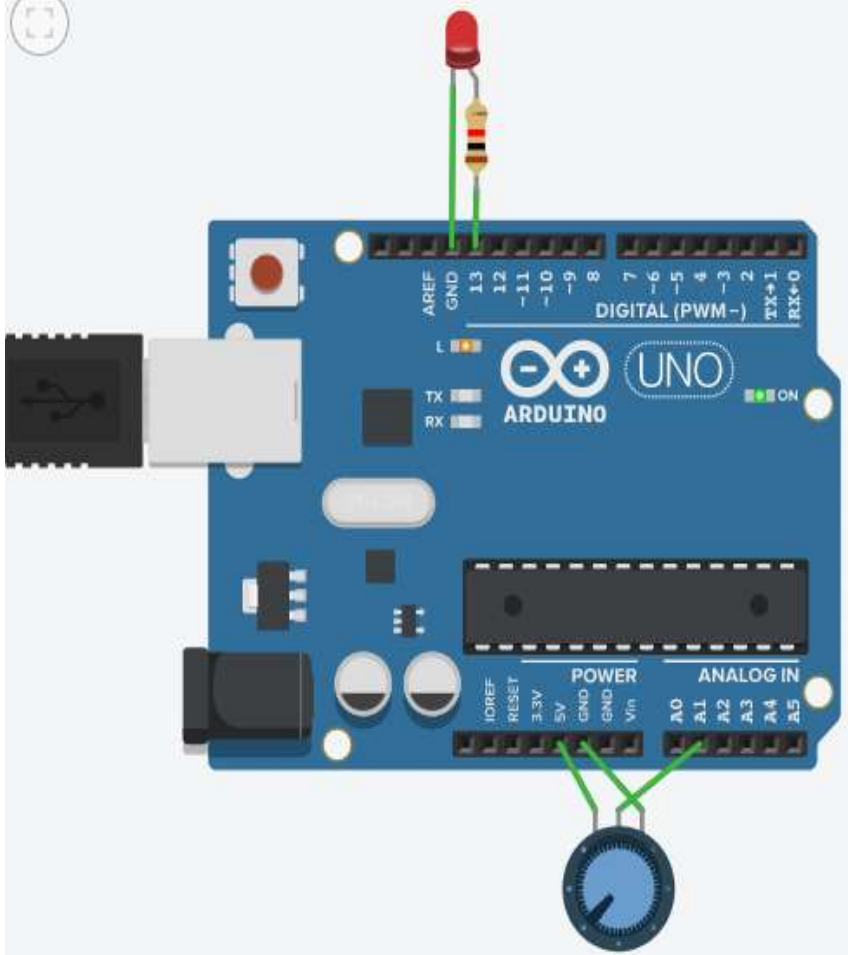
https://www.tinkercad.com/things/2OCPKLPhcOb-incredible-crift-densor/editel?tenant=circuits

Incredible Crift-Densor

All changes saved

Simulator time: 00:00:32

Code | Stop Simulation | Export | Share



Text

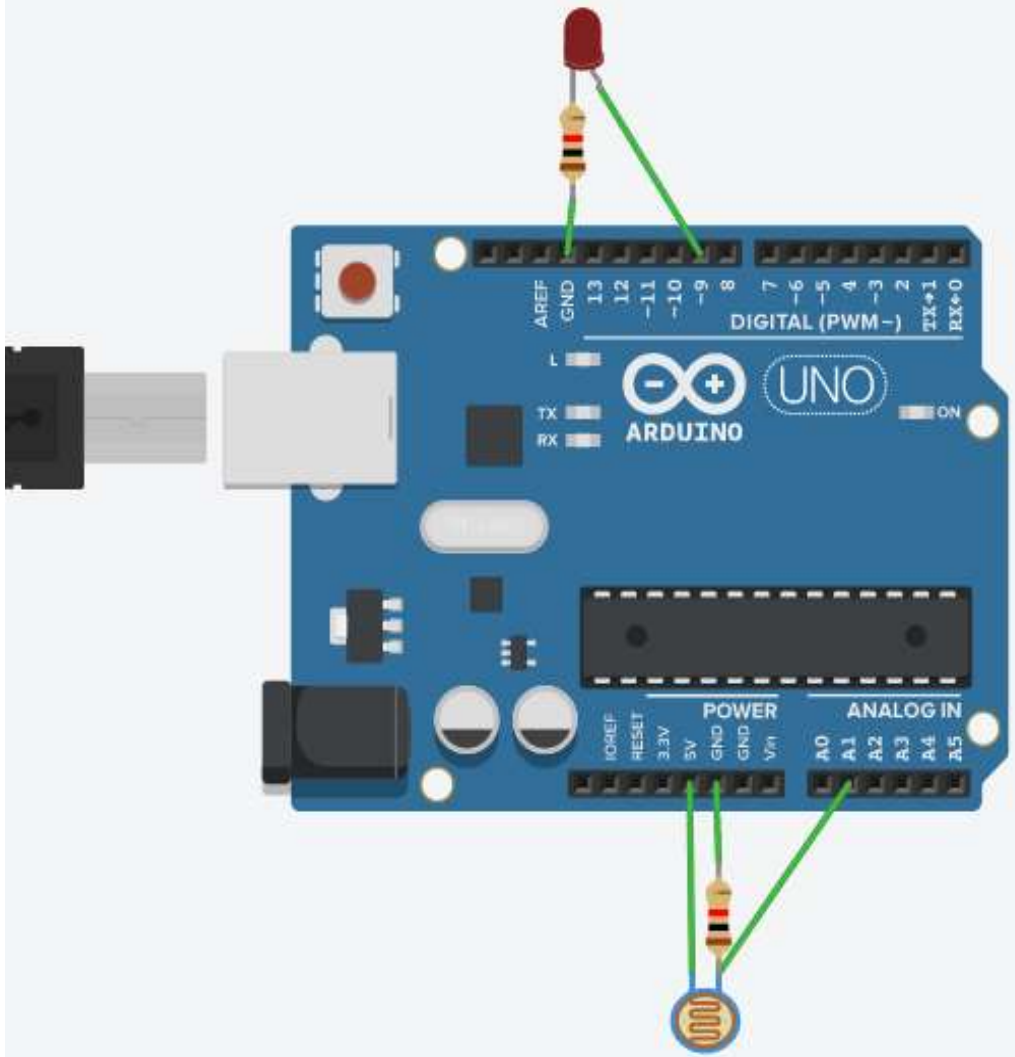
1 (Arduino Uno R3)

```
1  void setup()
2  {
3    pinMode(13, OUTPUT);
4    pinMode(A1, INPUT);
5    Serial.begin(9600);
6  }
7
8  void loop()
9  {
10   int var=0, y=0;
11   var=analogRead(A1);
12   y = map(var, 1, 1023, 1,255);
13   analogWrite(13, y);
14   delay(1000); // Wait for 1000 millisecond(s)
15   Serial.print(var);
16   Serial.print(":");
17   Serial.println(y);
18
```

Serial Monitor

```
1022:254
328:82
1:1
1:1
62:16
409:102
675:168
675:168
```

Send | Clear



Code | Start Simulation | Export | Share

Text | 1 (Arduino Uno R3)

```
1  void setup()
2  {
3    pinMode(9, OUTPUT);
4    pinMode(A1, INPUT);
5    Serial.begin(9600);
6  }
7
8  void loop()
9  {
10     int var; int y;
11     var=analogRead(A1);
12     Serial.print(var);
13     Serial.print(":");
14     y=map(var,1,700,1,256);
15     Serial.println(y);
16     analogWrite(9,y);
17 }
```

Serial Monitor

```
6:2
6:2
6:2
6:2
6:2
6:2
6:2
6:2
```

Send | Clear

Servo Motor

- A Servo Motor is a small device that has an output shaft. This shaft can be positioned to specific angular positions by sending the servo a coded signal. Usually, shaft can turn upto 180 degrees.

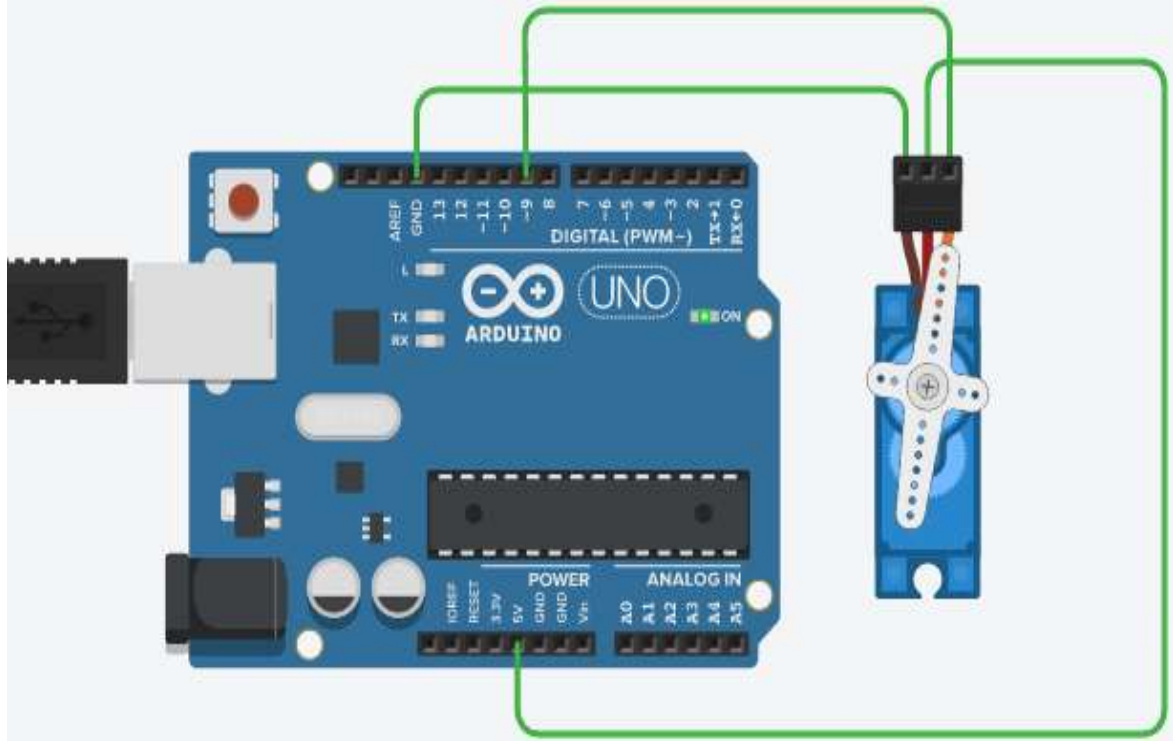
- Servo motors were first used in the Remote Control (RC) world, usually to control the steering of RC cars or the flaps on a RC plane. With time, they found their uses in robotics, automation, and of course, the Arduino world.

Differences between Sensor and Actuator

Sensor	Actuator
A device that detect events or changes in the environment and sends that information to other electronic devices	A component of a machine that is responsible for moving and controlling mechanism
Connected to the input ports of the System	Connected to the output ports of the system
Helps to monitor the changes in the environment	Helps to control the environment or physical changes
Output is an electrical signal.. Examples: BioSensors, image sensors, soil moisture, DHT, Ultrasonic Distance	Output is a movement Example: electric motors,stepper motor, hydraulic cylinders etc.

Simulator time: 00:00:11

Code | Stop Simulation | Export | Share



Text

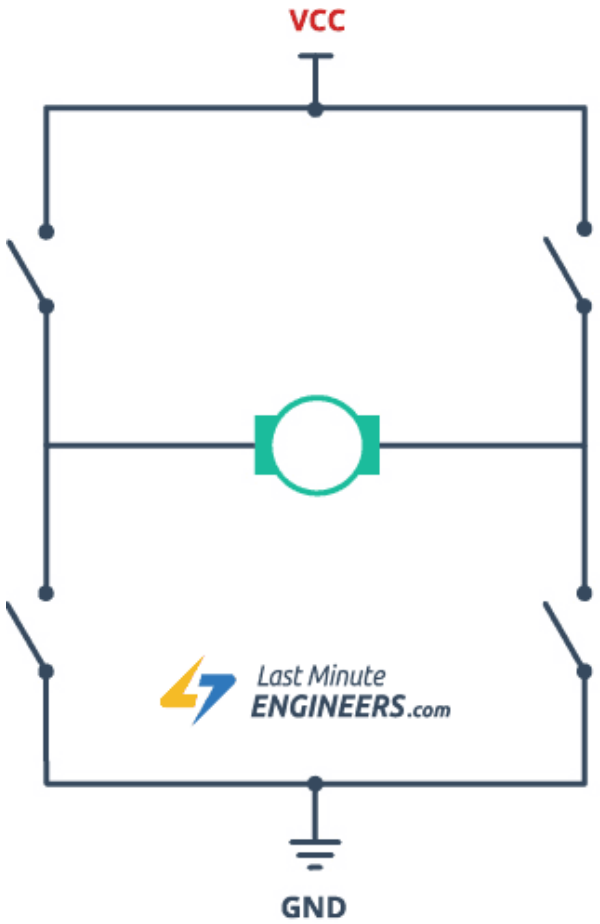
1 (Arduino Uno R3)

```
#include<Servo.h>
Servo myservo;
void setup()
{
  myservo.attach(9);
}

void loop()
{
  delay(1000);
  myservo.write(0);     delay(1000);
  myservo.write(45);    delay(1000);
  myservo.write(90);    delay(1000);
  myservo.write(135);   delay(1000);
  myservo.write(180);   delay(1000);

}
```

Serial Monitor

DC motors

- DC motors are interfaced to Arduino using L293D Motor Driver IC. It can control both speed and spinning direction of two DC motors.
- The L293D is a dual-channel H-Bridge motor driver capable of driving a pair of DC motors or one stepper motor.



IN1	IN2	Spinning Direction
Low(0)	Low(0)	Motor OFF
High(1)	Low(0)	Forward
Low(0)	High(1)	Backward
High(1)	High(1)	Motor OFF

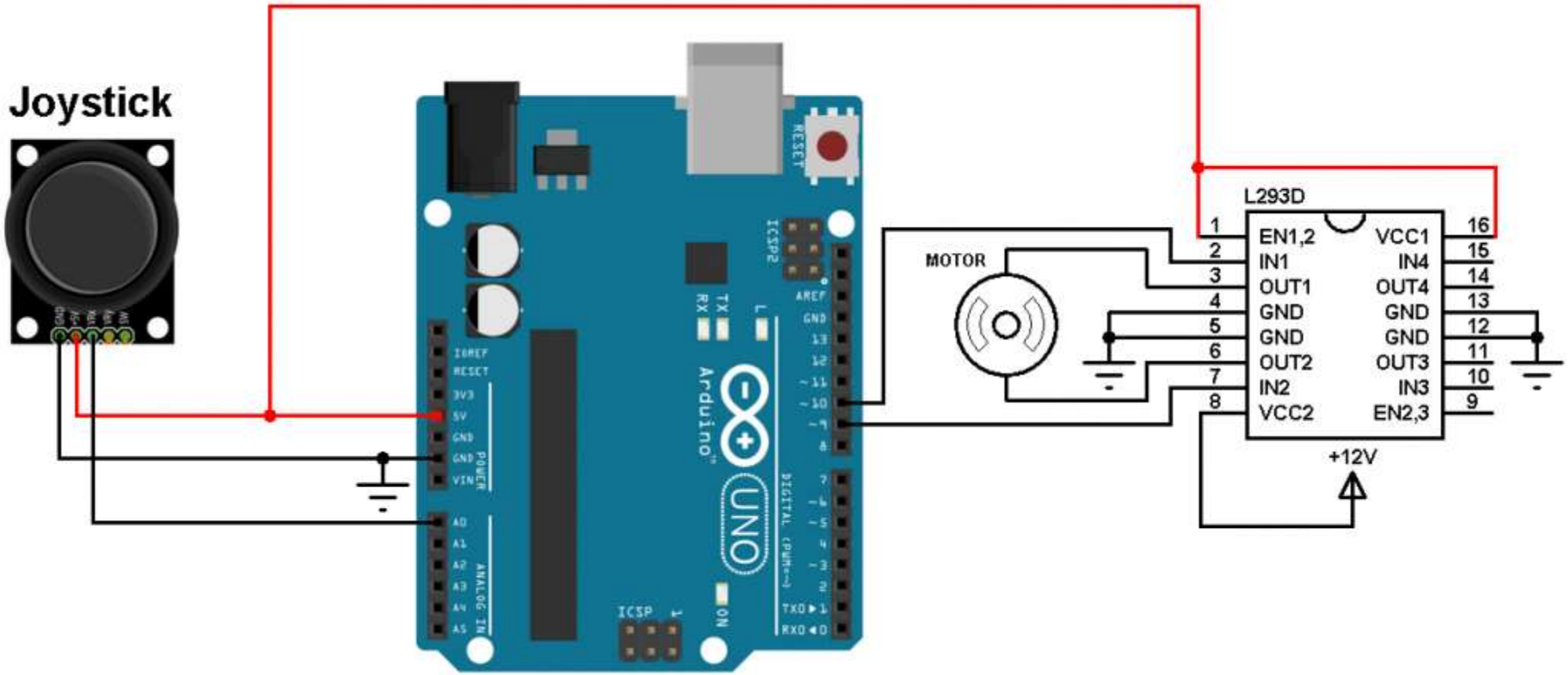
https://lastminuteengineers.com/l293d-dc-motor-arduino-tutorial/

L293D Pinout

Last Minute ENGINEERS.com

The L293D motor driver's output channels for the motor A and B are brought out to pins OUT1,OUT2 and OUT3,OUT4 respectively.

For each of the L293D's channels, there are two types of control pins which allow us to control speed and spinning direction of the DC motors at the same time viz. Direction control pins & Speed control pins. IN1 , IN2 & IN3 , IN4

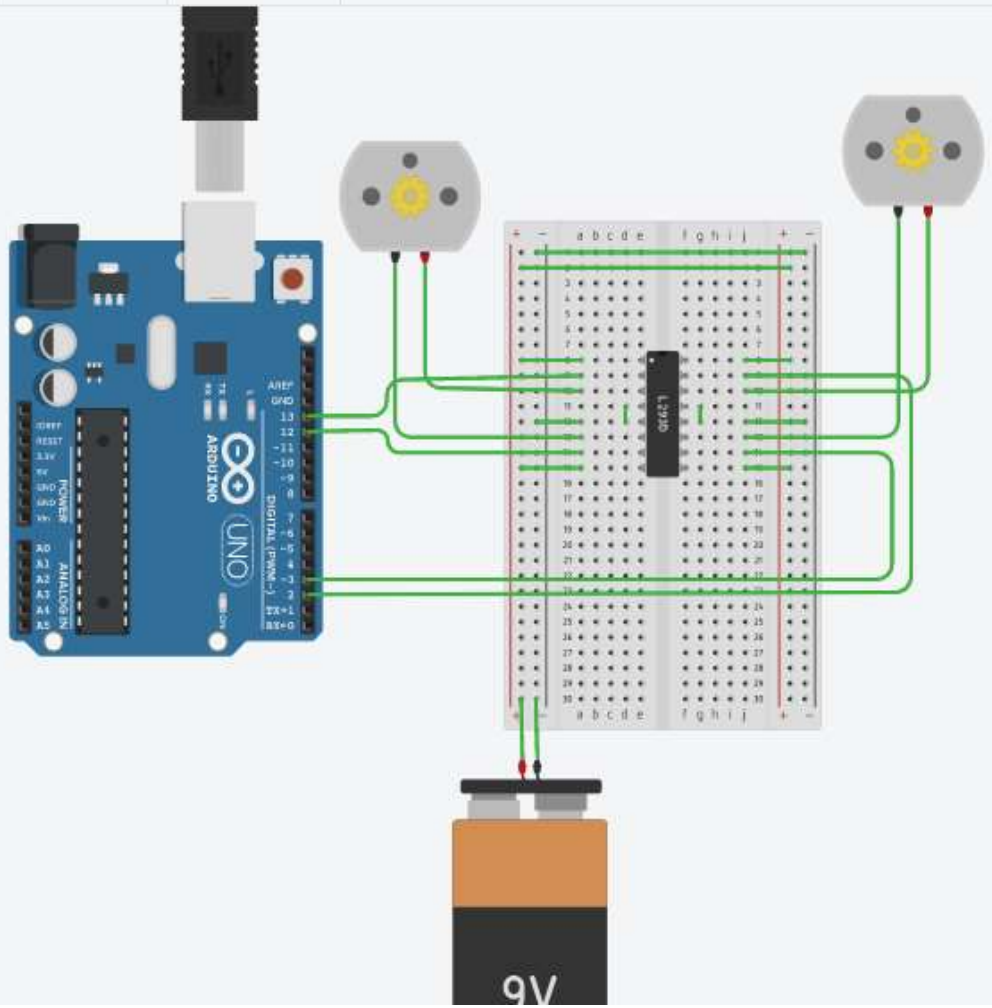

Joystick
L293D
MOTOR
EN1,2
IN1
OUT1
GND
GND
OUT2
IN2
VCC2
VCC1
IN4
OUT4
GND
GND
OUT3
IN3
EN2,3
+12V
1
2
3
4
5
6
7
8
16
15
14
13
12
11
10
9
Arduino
UNO
RESET
ICSP2
AREF
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
ICSP
ON
TX
RX
L
DIGITAL (PWM~)
TX0
RX0

https://www.tinkercad.com/things/34CtzYhrPFw-incredible-trug/editel?tenant=circuits

Incredible Trug

All changes saved

Code | Start Simulation | Export | Share



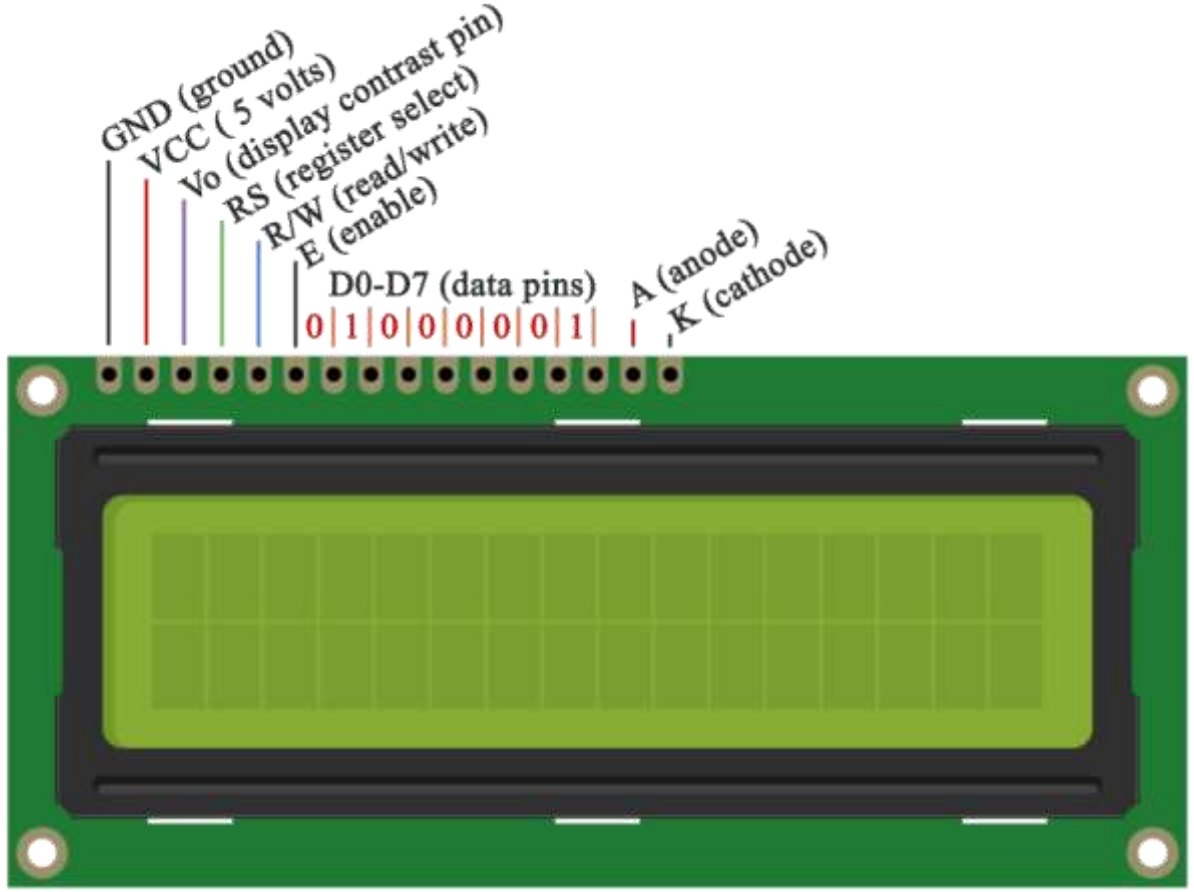
Text | 1 (Arduino Uno R3)

```
1  void setup()
2  {
3  pinMode(2,OUTPUT);
4  pinMode(3,OUTPUT);
5  pinMode(12,OUTPUT);
6  pinMode(13,OUTPUT);}
7
8  void loop()
9  {
10 digitalWrite(2,LOW);
11 digitalWrite(3,HIGH);
12 digitalWrite(12,LOW);
13 digitalWrite(13,HIGH);
14 }
15
16 // IC - L293D
```

Serial Monitor

LCD – Liquid Crystal Display

The library works with in either 4- or 8-bit mode (i.e. using 4 or 8 data lines).
Vo pin on which we can attach a potentiometer for controlling the contrast of the display.
The **RS pin** is set on low state or zero volts, then we are sending commands to the LCD like: set the cursor to a specific location, clear the display, turn off the display and so on.
R/W pin - read/write mode



Liquid Crystal Library - #include <LiquidCrystal.h>
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

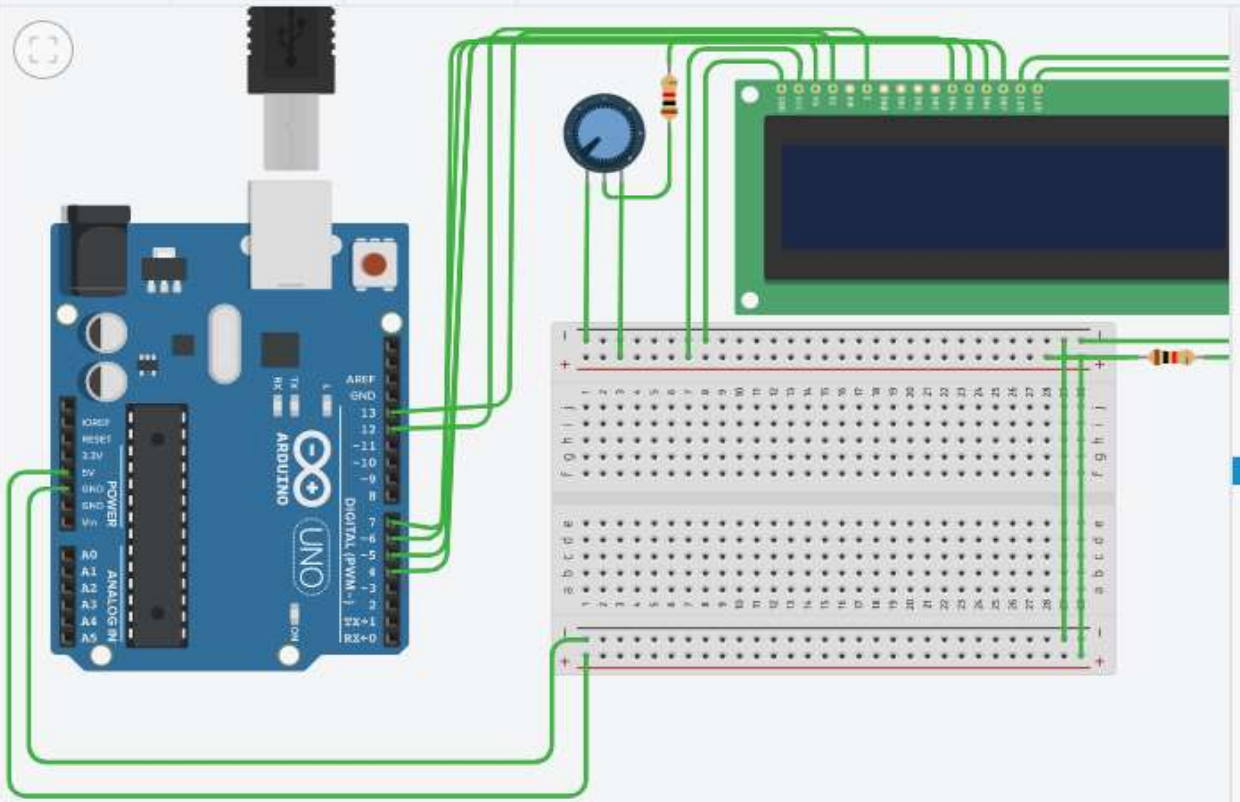
- lcd.begin()
- lcd.print()
- lcd.setCursor()
- lcd.noDisplay();
- lcd.display();

https://www.tinkercad.com/things/7zJS2hA73Gi-glorious-gaaris-borwo/editel

Glorious Gaaris-Borwo

All changes saved

Code | Start Simulation | Export | Share



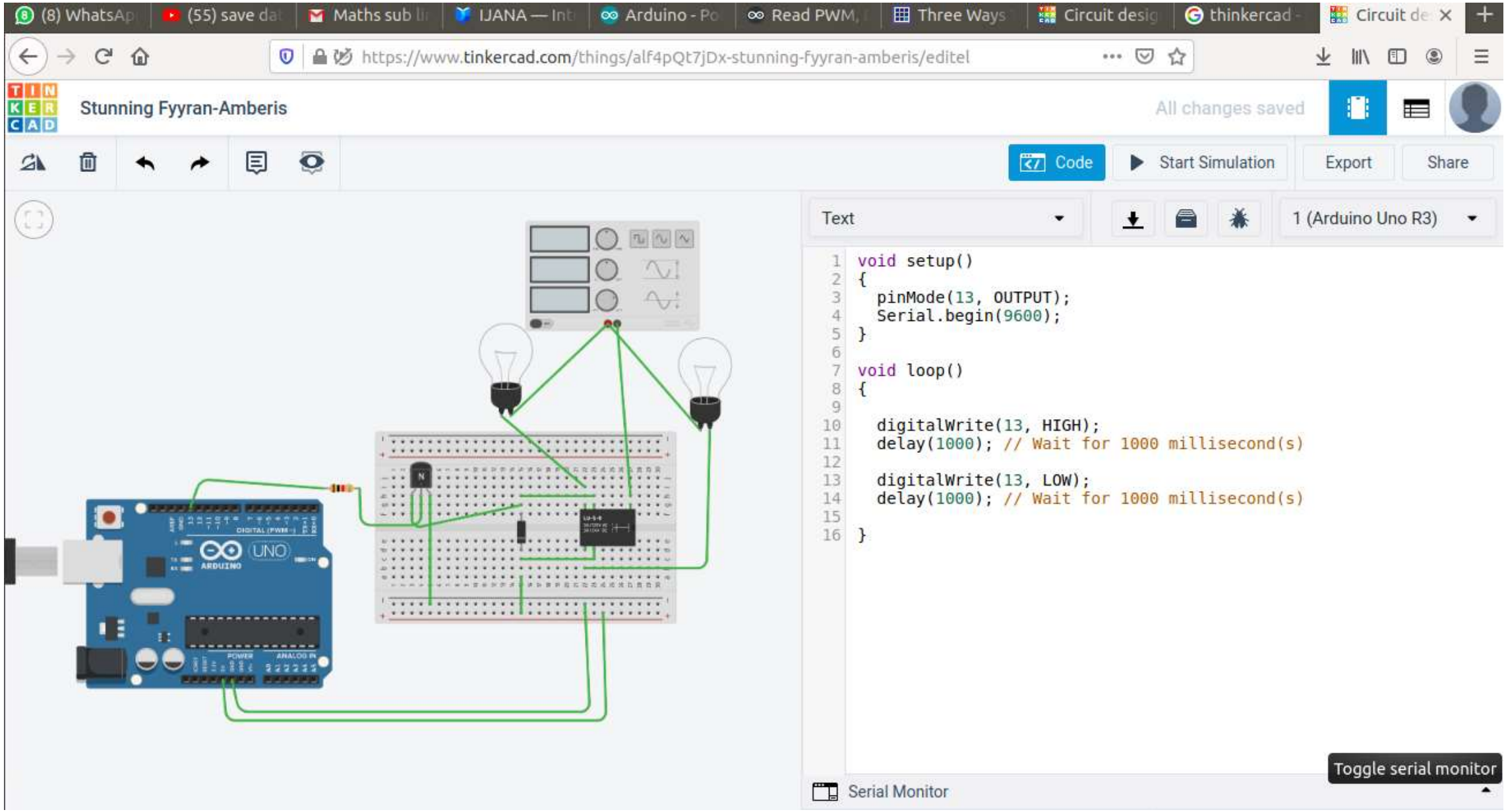
Text

1 (Arduino Uno R3)

```
#include<LiquidCrystal.h>
LiquidCrystal lcd(13,12,4,5,6,7);
void setup()
{


  lcd.begin(16,2);
}

void loop()
{
 lcd.clear();
  lcd.write("Mohan Rhitesh");
  delay(1000);
}
```

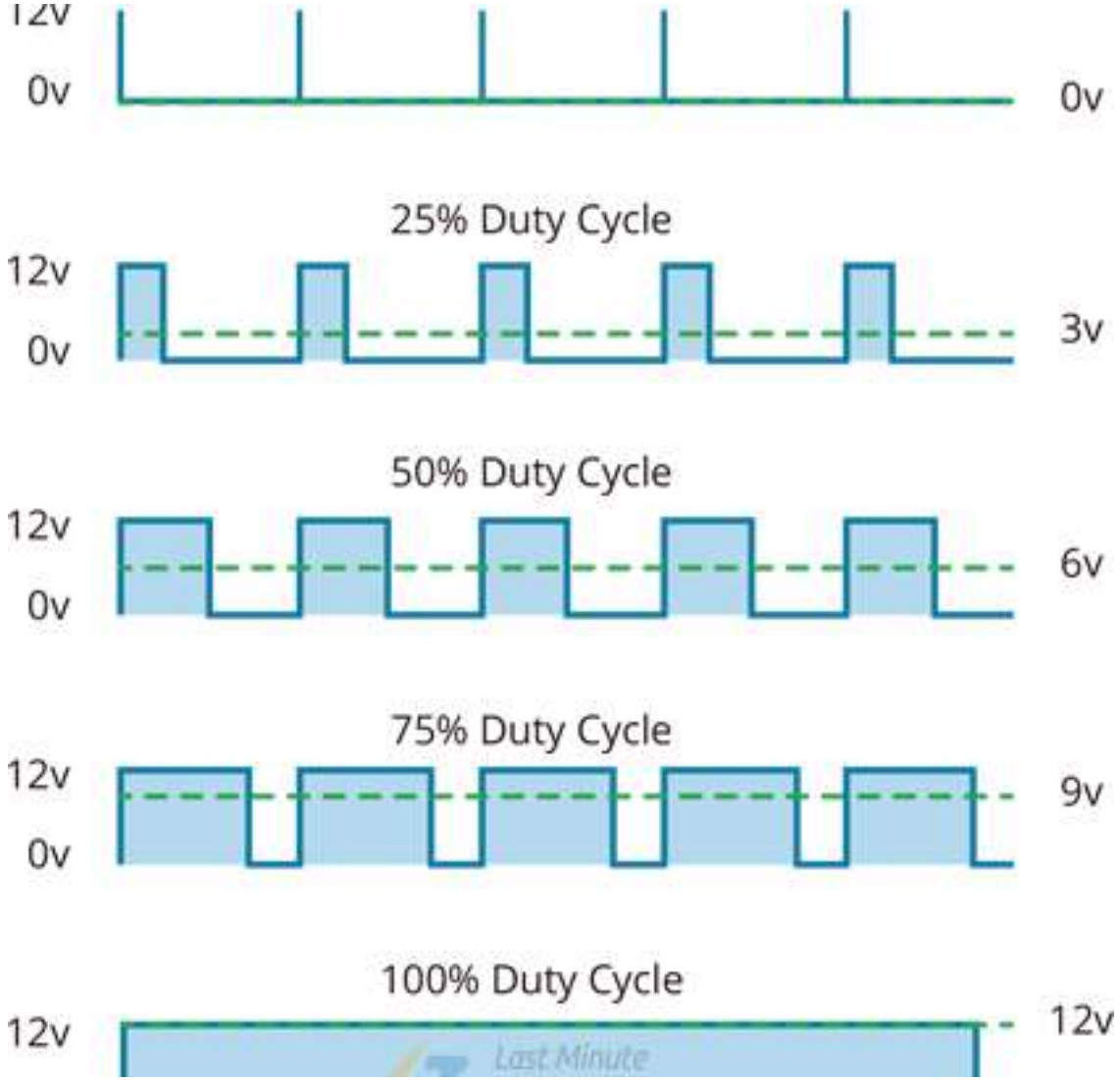
Toggle serial monitor

Serial Monitor


(8) WhatsAp
(55) save dat
Maths sub lir
IJANA — Int
Arduino - Po
Read PWM,
Three Ways
Circuit desig
thinkercad -
Circuit de
https://www.tinkercad.com/things/alf4pQt7jDx-stunning-fyyran-amberis/editel
Stunning Fyyran-Amberis
All changes saved
Code
Start Simulation
Export
Share
Text
1 (Arduino Uno R3)
void setup()
{
pinMode(13, OUTPUT);
Serial.begin(9600);
}
void loop()
{
digitalWrite(13, HIGH);
delay(1000); // Wait for 1000 millisecond(s)
digitalWrite(13, LOW);
delay(1000); // Wait for 1000 millisecond(s)
}
Toggle serial monitor
Serial Monitor

PWM - Pulse Width Modulation

- PWM is a technique where average value of the input voltage is adjusted by sending a series of ON-OFF pulses.
- The average voltage is proportional to the width of the pulses known as Duty Cycle.
- The higher the duty cycle, the greater the average voltage being applied to the dc motor(High Speed) and the lower the duty cycle, the less the average voltage being applied to the dc motor(Low Speed).
- Below image illustrates PWM technique with various duty cycles and average voltages.



Module 3 and 4

IOT COMMUNICATION MODEL AND PROTOCOLS

Outline

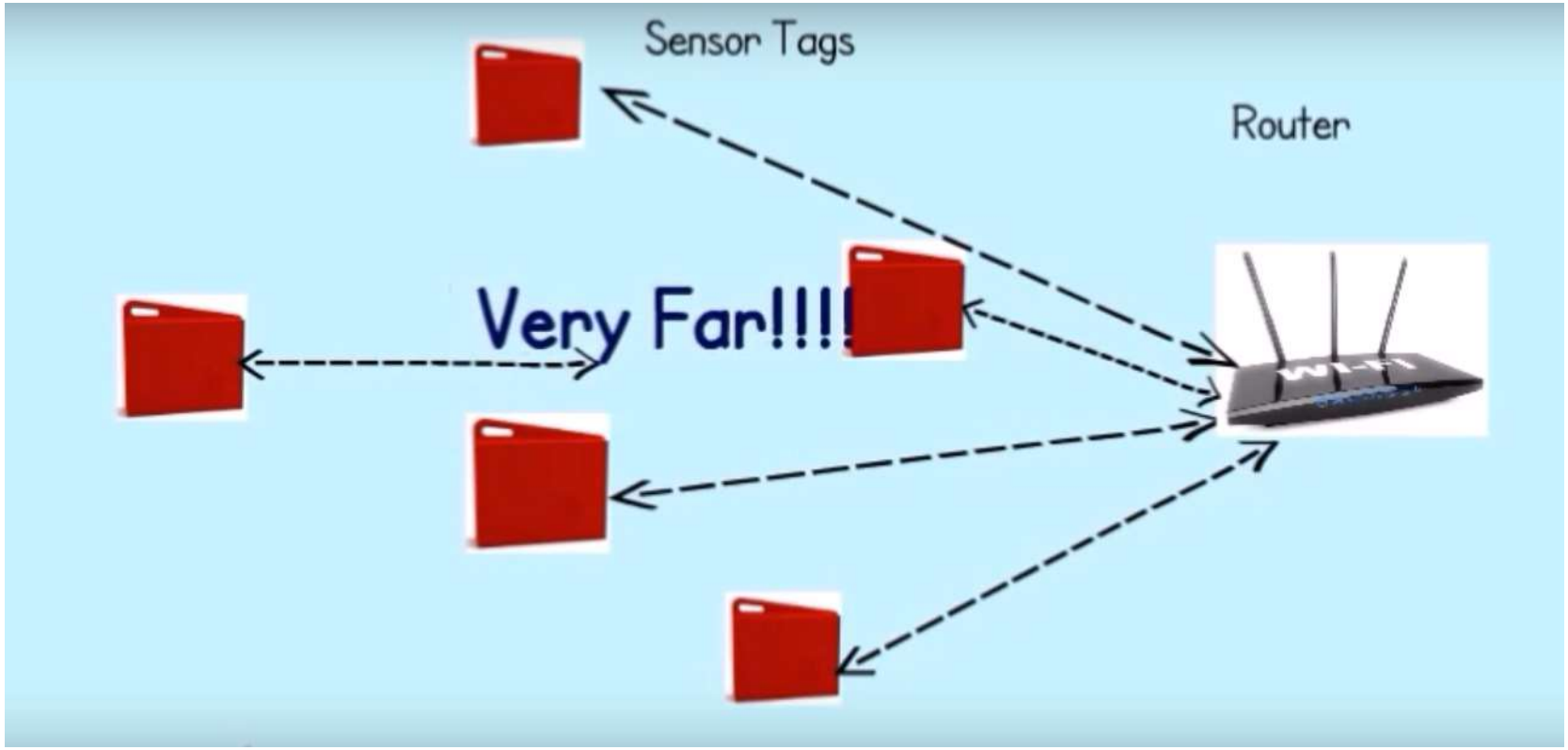
Connectivity Protocols

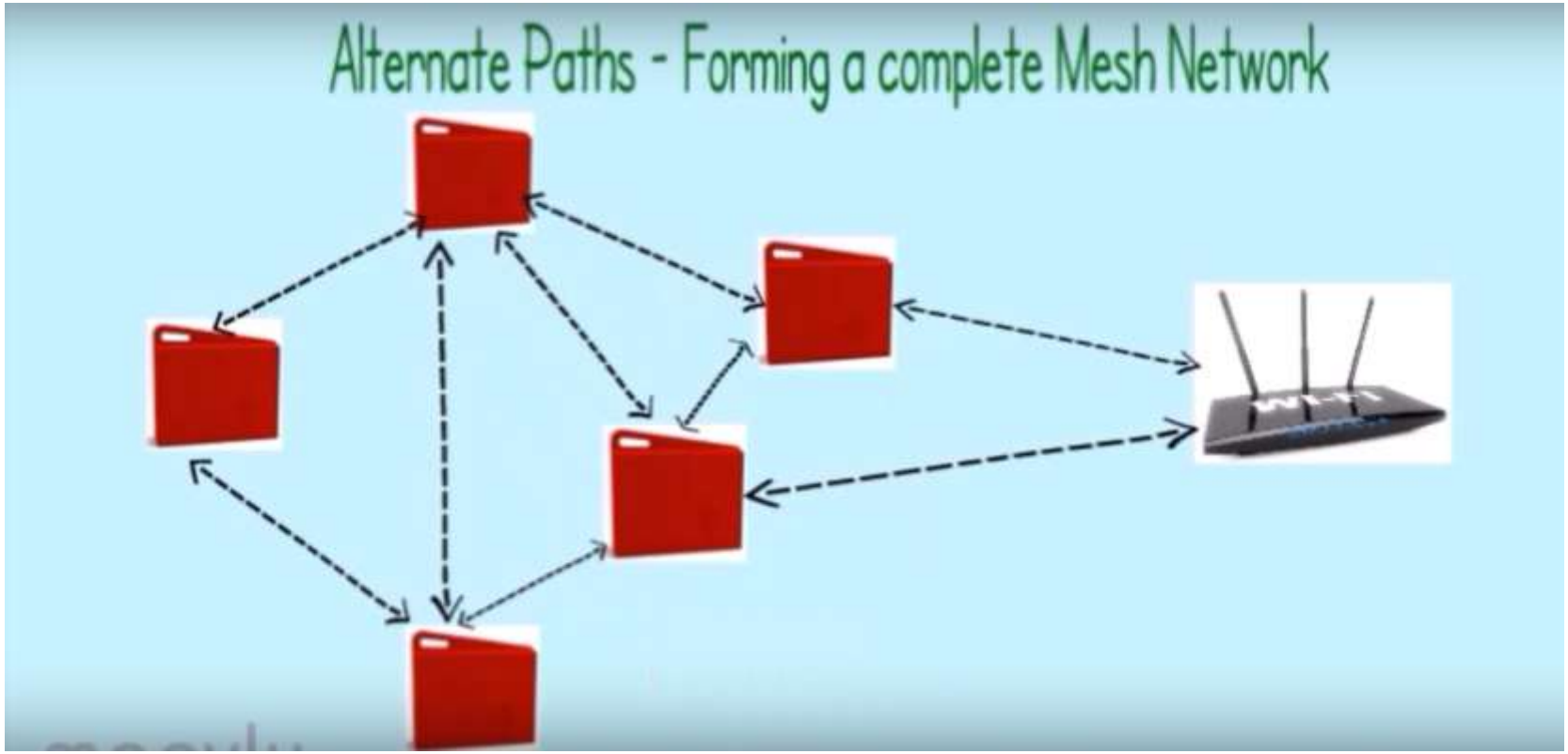
- 6LoWPAN
- IEEE 802.15.4
- Zigbee
- Wireless HART
- Z-Wave
- ISA 100
- NFC

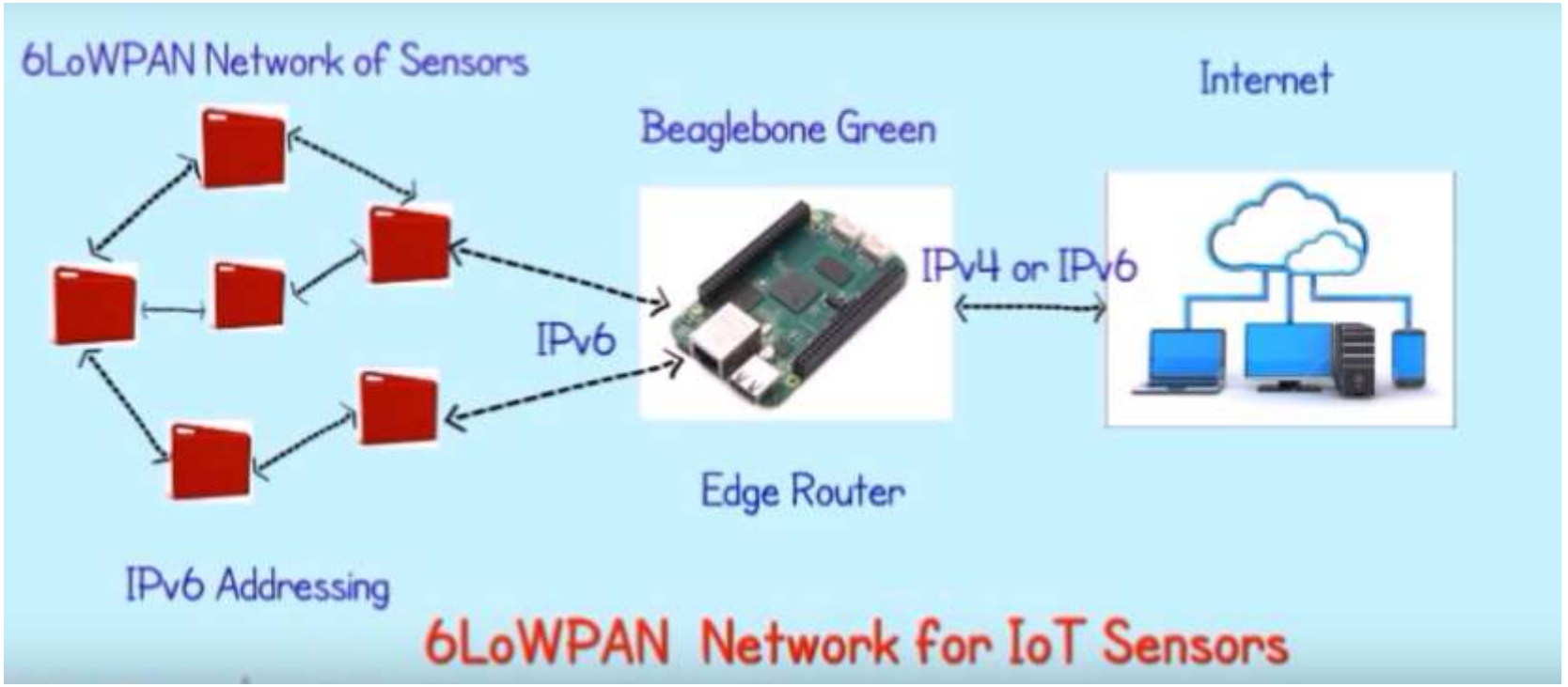
Outline

Data Protocols:

- Message Queue Telemetry Transport (MQTT)
- Constrained Application Protocol (CoAP)
- Advanced Message Queuing Protocol (AMQP)
- XMPP – Extensible Messaging and Presence Protocol.


Sensor Tags
Router
Very Far!!!!


Alternate Paths - Forming a complete Mesh Network


6LoWPAN Network of Sensors
Internet
Beaglebone Green
IPv4 or IPv6
IPv6
Edge Router
IPv6 Addressing
6LoWPAN Network for IoT Sensors

6LoWPAN

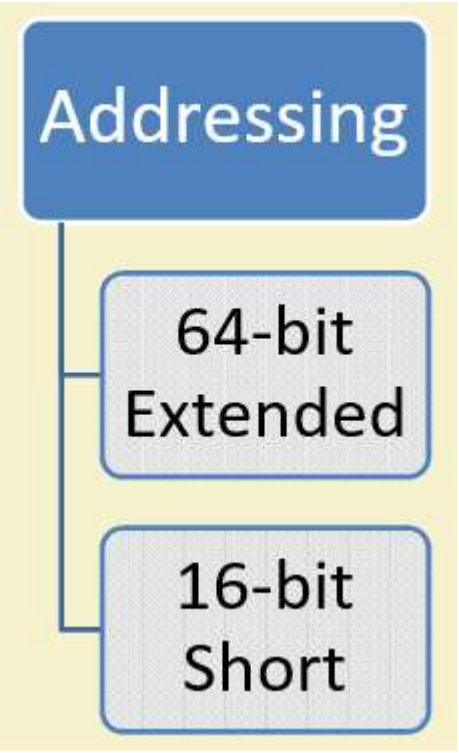
Introduction

- Low-power Wireless Personal Area Networks over IPv6.
- Allows for the smallest devices with limited processing ability to transmit information wirelessly using an Internet protocol.
- Allows low-power devices to connect to the Internet.
- Created by the Internet Engineering Task Force (IETF) - RFC 5933 and RFC 4919.

Features of 6LoWPANs

- Allows IEEE 802.15.4 radios to carry 128-bit addresses of Internet Protocol version 6 (IPv6).
- Header compression and address translation techniques allow the IEEE 802.15.4 radios to access the Internet.
- IPv6 packets compressed and reformatted to fit the IEEE 802.15.4 packet format.
- Uses include IoT, Smart grid, and M2M applications.

Addressing in 6LoWPAN



64-bit addresses: globally unique

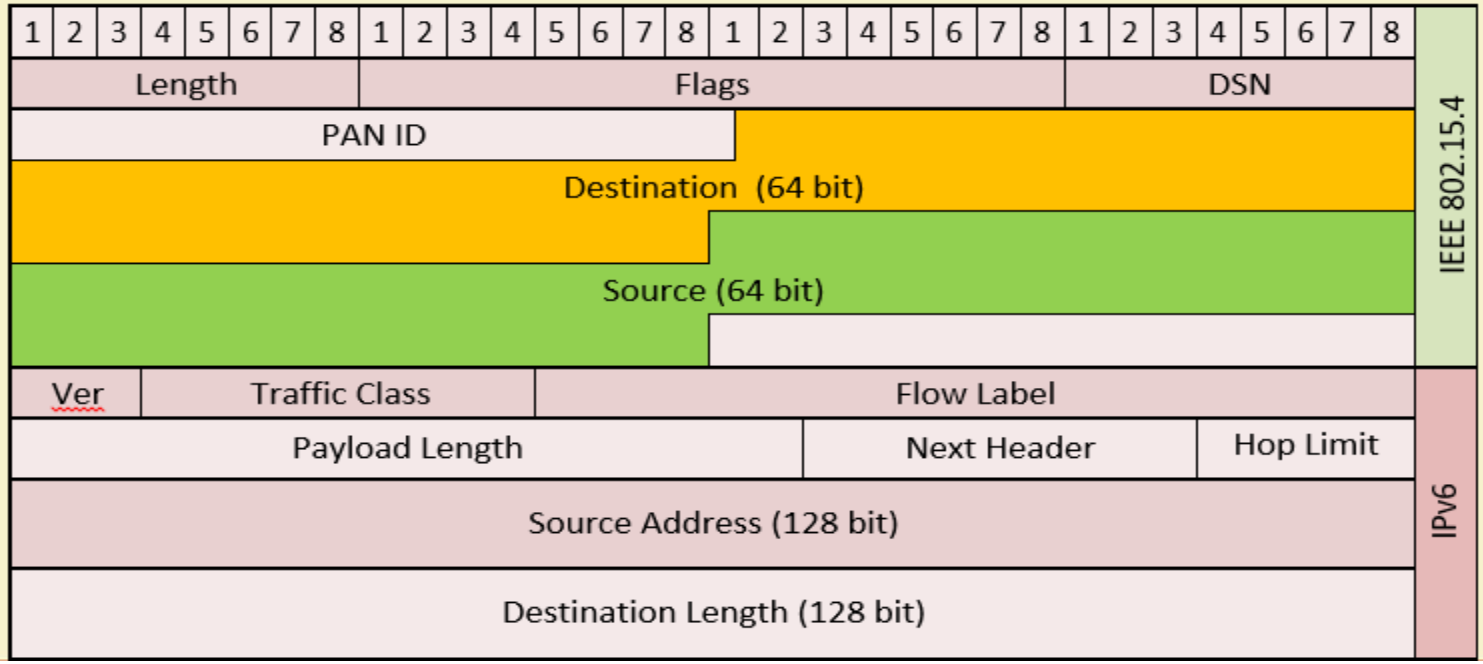
16 bit addresses: PAN specific; assigned by PAN coordinator

IPv6 multicast not supported by 802.15.4

IPv6 packets carried as link layer broadcast frames

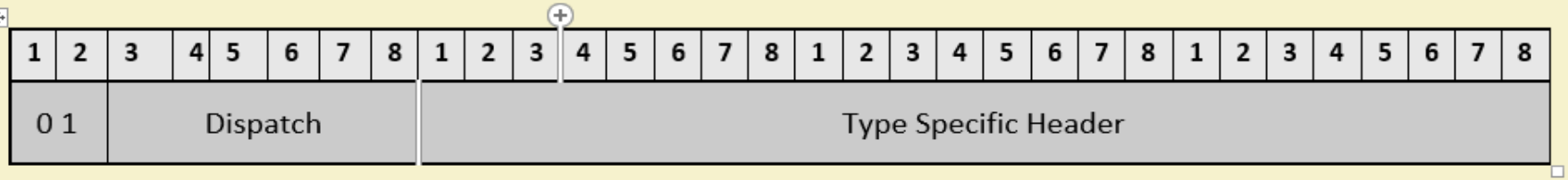
6LoWPAN

6LowPAN Packet Format



6LoWPAN

Header Type: Dispatch Header



- **Dispatch:** Initiates communication
- **0,1**: Identifier for Dispatch Type
- **Dispatch**:
  - 6 bits
  - Identifies the next header type
- **Type Specific Header:**
  - Determined by Dispatch header

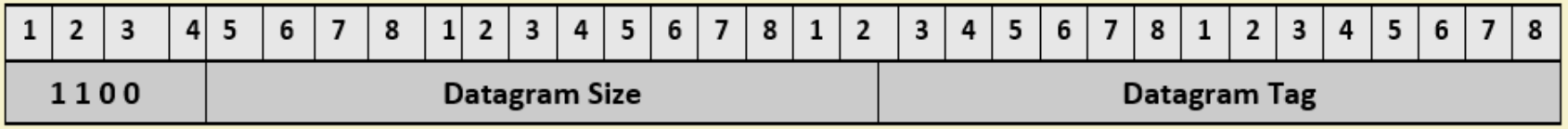
6LoWPAN

Header Type: Mesh Addressing Header

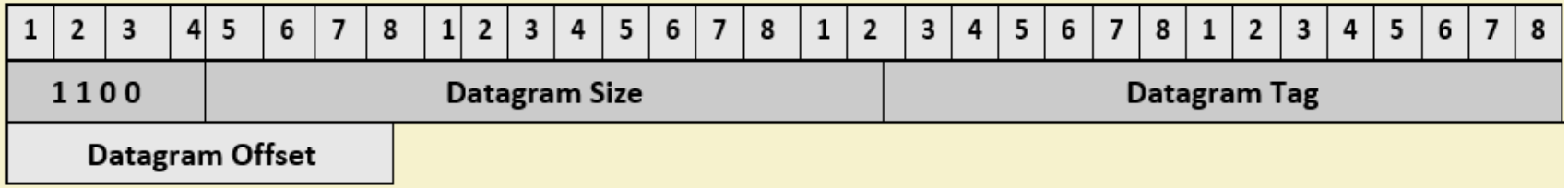
- **1,0**: ID for Mesh Addressing Header
- **V**: '0' if originator is 64-bit extended address, '1' if 16-bit address
- **F**: '0' if destination is 64-bit addr., '1' if 16-bit addr.
- **Hops Left**: decremented by each node before sending to next hop

6LoWPAN

Header Type: Fragmentation Header



(a) First Fragment

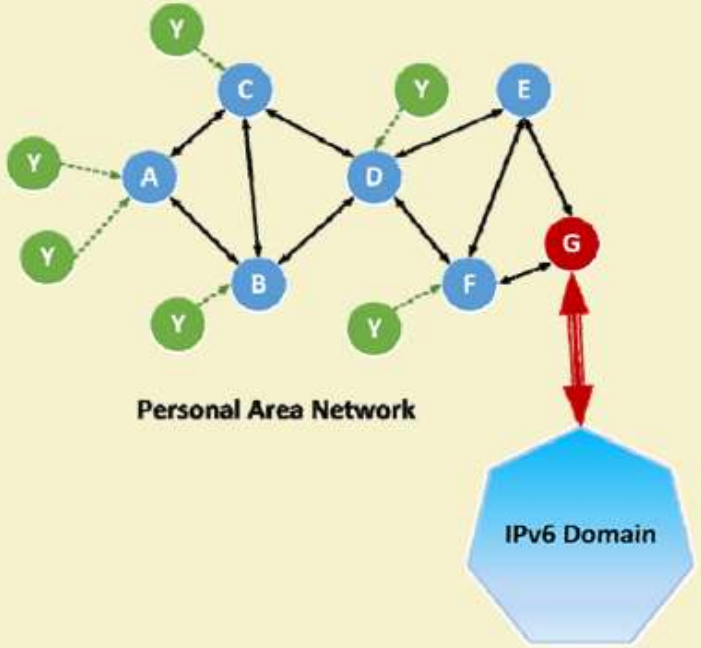


(b) Subsequent Fragment

6LoWPAN

6LoWPAN Routing Considerations

- ✓ Mesh routing within the PAN space.
- ✓ Routing between IPv6 and the PAN domain
- ✓ Routing protocols in use:
  - **LOADng**
  - **RPL**



6LoWPAN

- LOADng Routing
- Derived from AODV and extended for use in IoT.
- Basic operations of LOADng include:
- Generation of **Route Requests (RREQs)** by a LOADng Router (originator) for discovering a route to a destination,
- **Forwarding of such RREQs** until they reach the destination LOADng Router,
- Generation of **Route Replies (RREPs)** upon receipt of an RREQ by the indicated destination, and unicast hop-by-hop forwarding of these RREPs towards the originator.

6LoWPAN

- LOADng Routing
- Derived from AODV and extended for use in IoT.
- Basic operations of LOADng include:
- Generation of **Route Requests (RREQs)** by a LOADng Router (originator) for discovering a route to a destination,
- **Forwarding of such RREQs** until they reach the destination LOADng Router,
- Generation of **Route Replies (RREPs)** upon receipt of an RREQ by the indicated destination, and unicast hop-by-hop forwarding of these RREPs towards the originator.

6LoWPAN

- If a route is detected to be broken, a **Route Error (RERR)** message is returned to the originator of that data packet to inform the originator about the route breakage.
- **Optimized flooding** is supported, reducing the overhead incurred by RREQ generation and flooding.
- Only the destination is permitted to respond to an RREQ.
- Intermediate LOADng Routers are explicitly prohibited from responding to RREQs, even if they may have active routes to the sought destination.
- RREQ/RREP messages generated by a given LOADng Router share a single unique, monotonically increasing sequence number.

6LoWPAN

- RPL Routing
- Distance Vector IPv6 **routing protocol for lossy and low power networks.**
- Maintains routing topology using low rate beaconing.
- Beaconing rate increases on detecting inconsistencies (e.g. node/link in a route is down).
- Routing information included in the datagram itself.
- **Proactive**: Maintaining routing topology.
- **Reactive**: Resolving routing inconsistencies.

6LoWPAN

- RPL separates packet processing and forwarding from the routing optimization objective, which helps in Low power Lossy Networks (LLN)
- RPL supports message confidentiality and integrity.
- Supports Data-Path Validation and Loop Detection
- Routing optimization objectives include
- minimizing energy
- minimizing latency
- satisfying constraints (w.r.t node power, bandwidth, etc.)

6LoWPAN

- RPL operations require bidirectional links.
- In some LLN scenarios, those links may exhibit asymmetric properties.
- It is required that the reachability of a router be verified before the router can be used as a parent.

Functionality-based IoT Protocol Organization

- **Connectivity** (6LowPAN, RPL)
- **Identification** (EPC, uCode, IPv6, URIs)
- **Communication / Transport** (WiFi, Bluetooth, LPWAN)
- **Discovery** (Physical Web, mDNS, DNS-SD)
- **Data Protocols** (MQTT, CoAP, AMQP, Websocket, Node)
- **Device Management** (TR-069, OMA-DM)
- **Semantic** (JSON-LD, Web Thing Model)
- **Multi-layer Frameworks** (Alljoyn, IoTivity, Weave, Homekit)

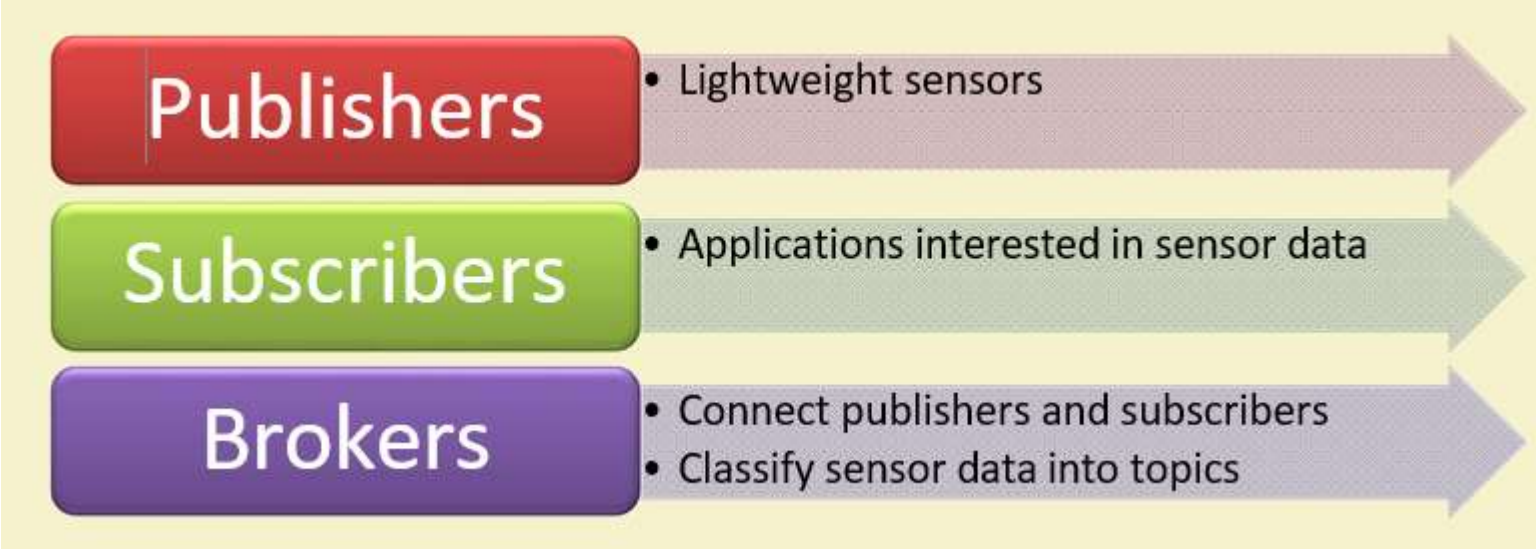
MQTT

- Introduction
- **Message Queue Telemetry Transport.**
- ISO standard (ISO/IEC PRF 20922).
- It is a publish-subscribe-based lightweight messaging protocol for use in conjunction with the TCP/IP protocol.
- MQTT was introduced by IBM in 1999 and standardized by OASIS in 2013.
- Designed to provide connectivity (mostly embedded) between applications and middle-wares on one side and networks and communications on the other side.

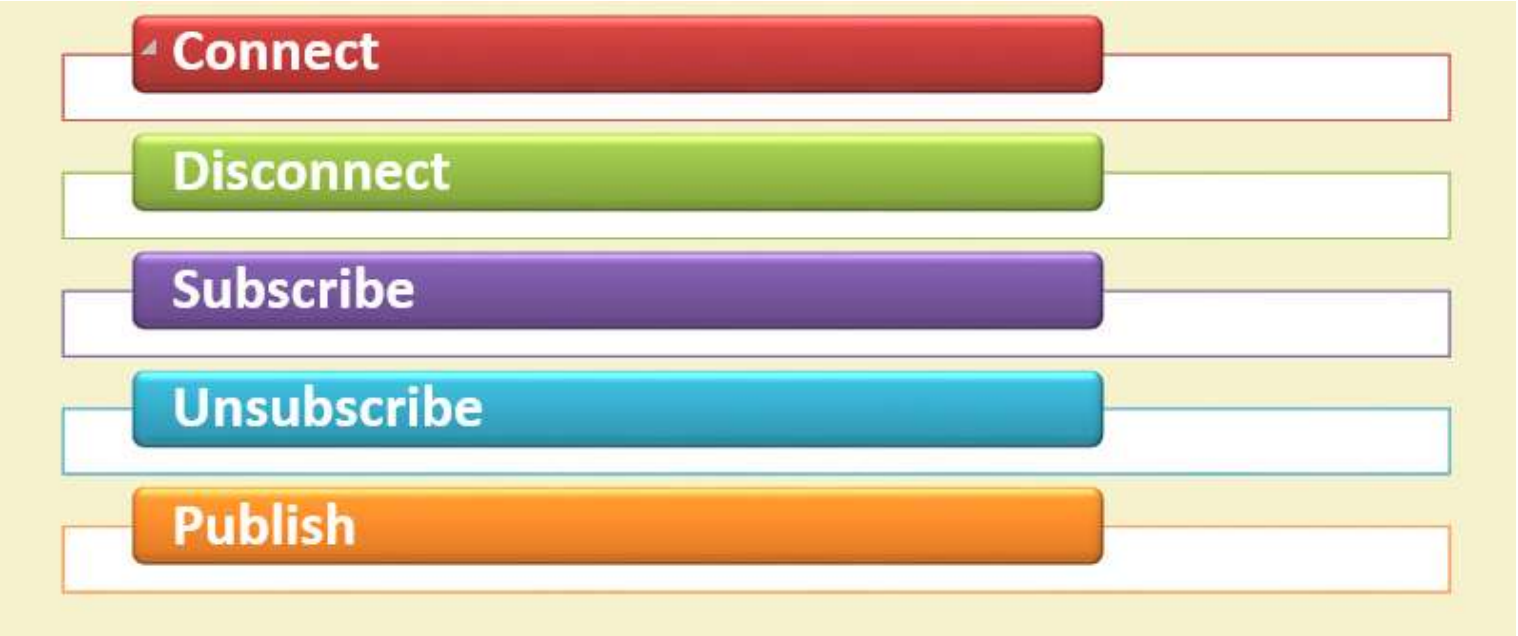
MQTT

- A message broker controls the publish-subscribe messaging pattern.
- A topic to which a client is subscribed is updated in the form of messages and distributed by the message broker.
- Designed for:
- Remote connections
- Limited bandwidth
- Small-code footprint

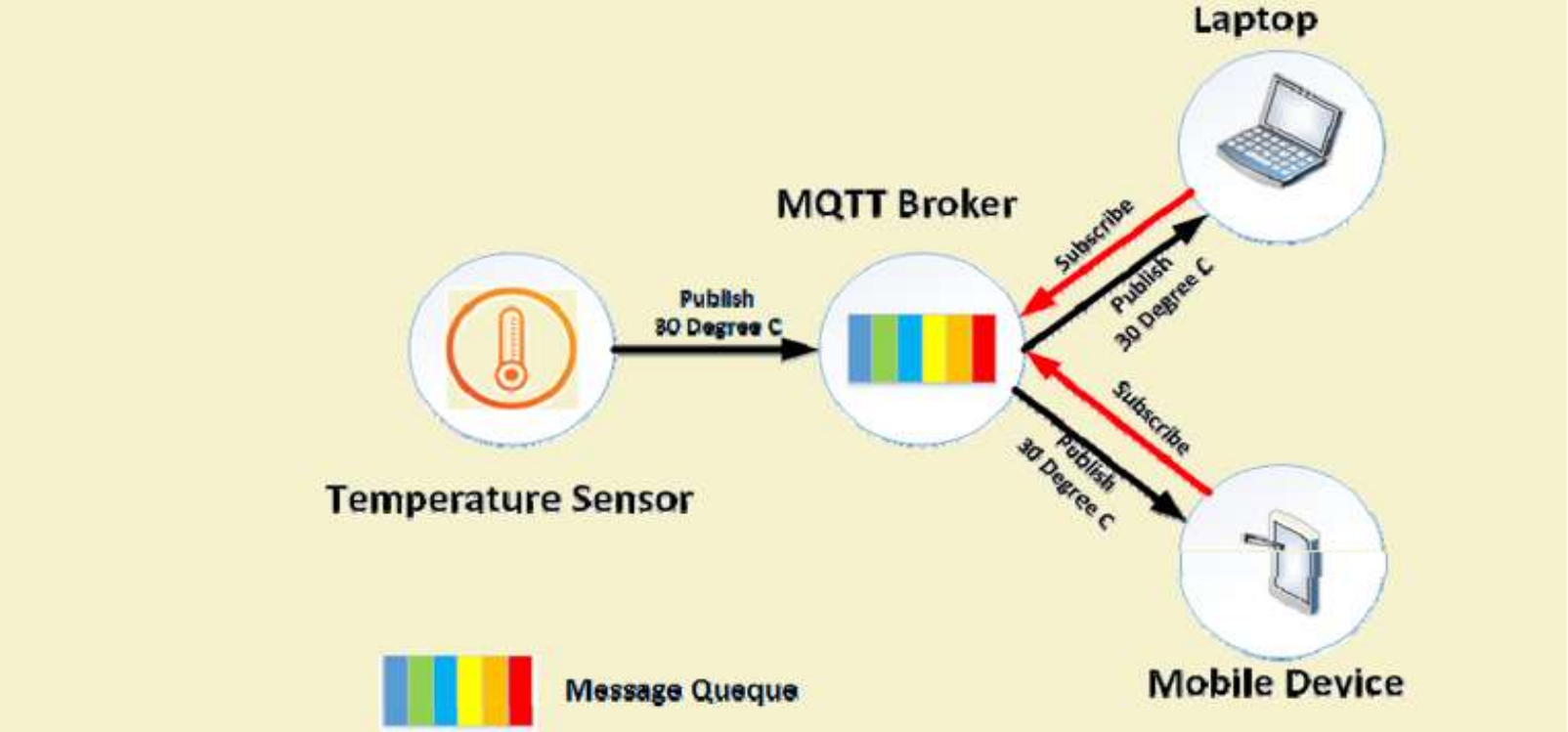
MQTT Components



MQTT Methods



MQTT


Laptop
MQTT Broker
Subscribe
Publish
30 Degree C
Publish
30 Degree C
Subscribe
Publish
30 Degree C
Temperature Sensor
Message Queque
Mobile Device

MQTT Communication

- The protocol uses a **publish/subscribe** architecture (HTTP uses a request/response paradigm).
- Publish/subscribe is **event-driven** and enables messages to be pushed to clients.
- The central **communication point is the MQTT broker**, which is in charge of dispatching all messages between the senders and the rightful receivers.
- Each client that publishes a message to the broker, includes a **topic** into the message. The **topic is the routing information for the broker.**

MQTT Communication

- Each client that wants to receive messages subscribes to a certain topic and the broker delivers all messages with the matching topic to the client.
- Therefore the clients don't have to know each other. They only communicate over the topic.
- This architecture enables highly scalable solutions without dependencies between the data producers and the data consumers.

MQTT Topics

- A topic is a **simple string** that can have more hierarchy levels, which are separated by a slash.
- A sample topic for sending temperature data of the living room could be *house/living-room/temperature*.
- On one hand the client (e.g. mobile device) can subscribe to the exact topic or on the other hand, it can use a **wildcard**.

MQTT Topics

- The subscription to *house/+/temperature* would result in all messages sent to the previously mentioned topic *house/living-room/temperature,* as well as any topic with an arbitrary value in the place of living room, such as *house/kitchen/temperature.*
- The plus sign is a **single level wild card** and only allows arbitrary values for one hierarchy.
- If more than one level needs to be subscribed, such as, the entire sub-tree, there is also a **multilevel wildcard** (#).
- It allows to subscribe to all underlying hierarchy levels.
- For example *house/#* is subscribing to all topics beginning with *house*.

MQTT Applications

- **Facebook Messenger** uses MQTT for online chat.
- **Amazon Web Services** use Amazon IoT with MQTT.
- **Microsoft Azure** IoT Hub uses MQTT as its main protocol for telemetry messages.
- The **EVRYTHNG IoT platform** uses MQTT as an M2M protocol for millions of connected products.
- **Adafruit** launched a free MQTT cloud service for IoT experimenters called Adafruit IO.

CoAP
Constrained Application Protocol

- Introduction

  - CoAP – **Constrained Application Protocol**.
  - **Web transfer protocol** for use with constrained nodes and networks.
  - **Designed for Machine to Machine** (M2M) applications such as smart energy and building automation.
  - Based on **Request-Response model** between end-points
  - Client-Server interaction is **asynchronous over a datagram oriented transport protocol** such as UDP

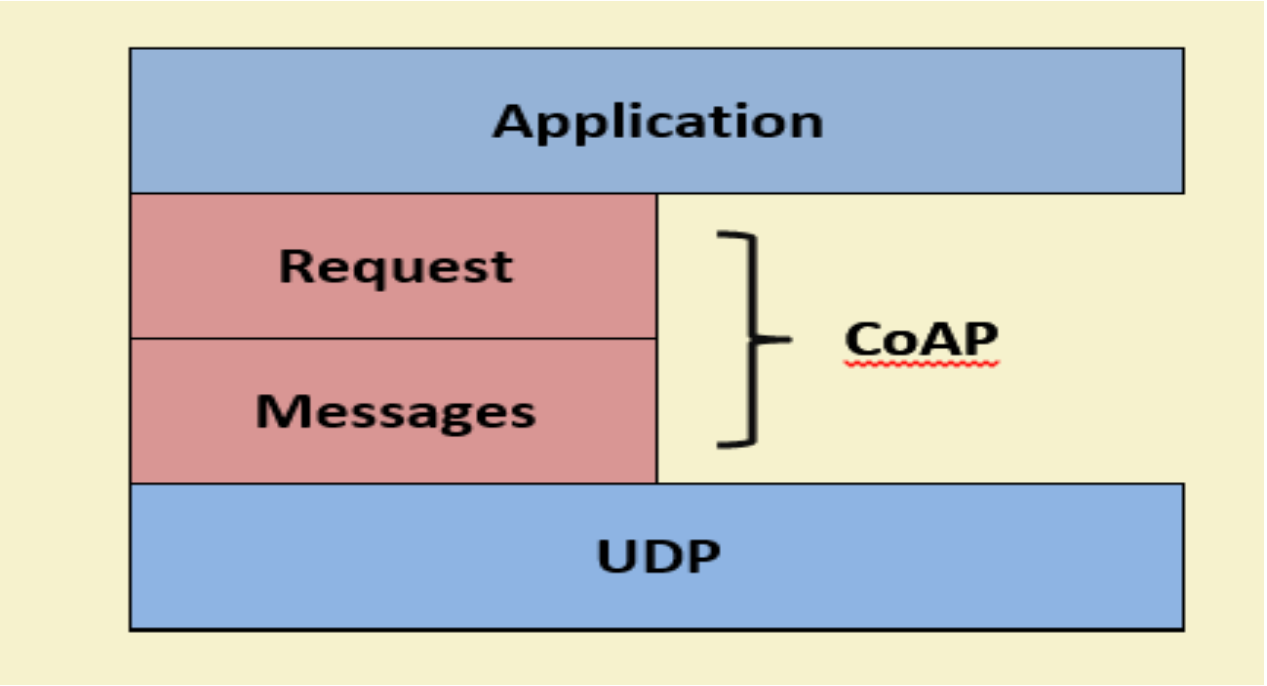
CoAP
Constrained Application Protocol

- The Constrained Application Protocol (CoAP) is a session layer protocol designed by IETF Constrained RESTful Environment (CoRE) working group to provide lightweight RESTful (HTTP) interface.
- Representational State Transfer (REST) is the standard interface between HTTP client and servers.
- Lightweight applications such as those in IoT, could result in significant overhead and power consumption by REST.
- CoAP is designed to enable low-power sensors to use RESTful services while meeting their power constraintshtweight RESTful (HTTP) interface.

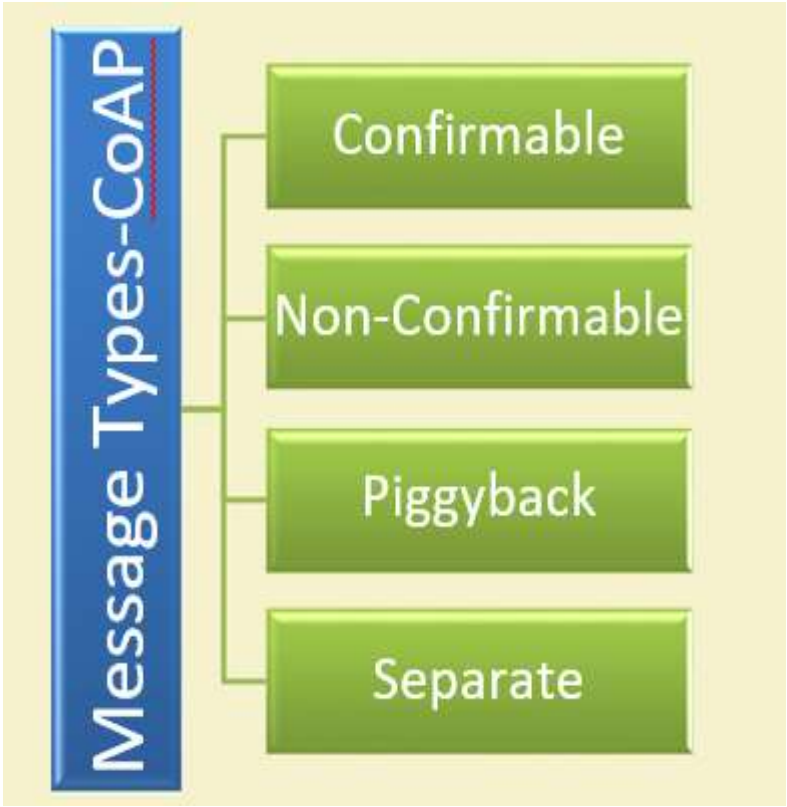
CoAP
Constrained Application Protocol

- Built over UDP, instead of TCP (which is commonly used with HTTP) and has a light mechanism to provide reliability
- CoAP architecture is divided into two main sub-layers:
    - Messaging
    - Request/response
  - The messaging sub-layer is responsible for reliability and duplication of messages, while the request/response sub-layer is responsible for communication.
  - CoAP has four messaging modes:
    - Confirmable
    - Non-confirmable
    - Piggyback
    - Separate

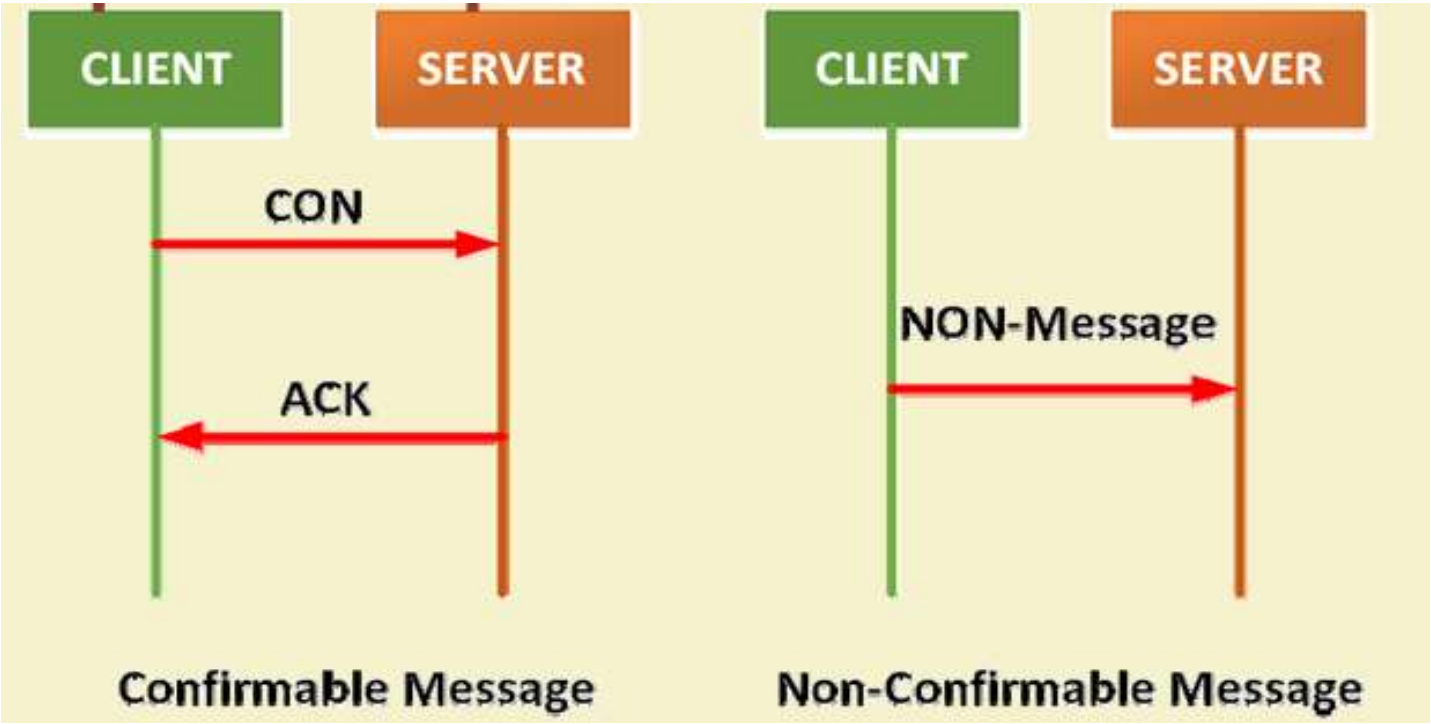
CoAP Position



CoAP Message Type



CoAP Request-Response Model



Request

Response

(XML/JSON)

servlet

XML

```
<country>
<city>
   <temp>33</temp>
   <humidity>40</humidity>
</city>
</country>
```

JSON

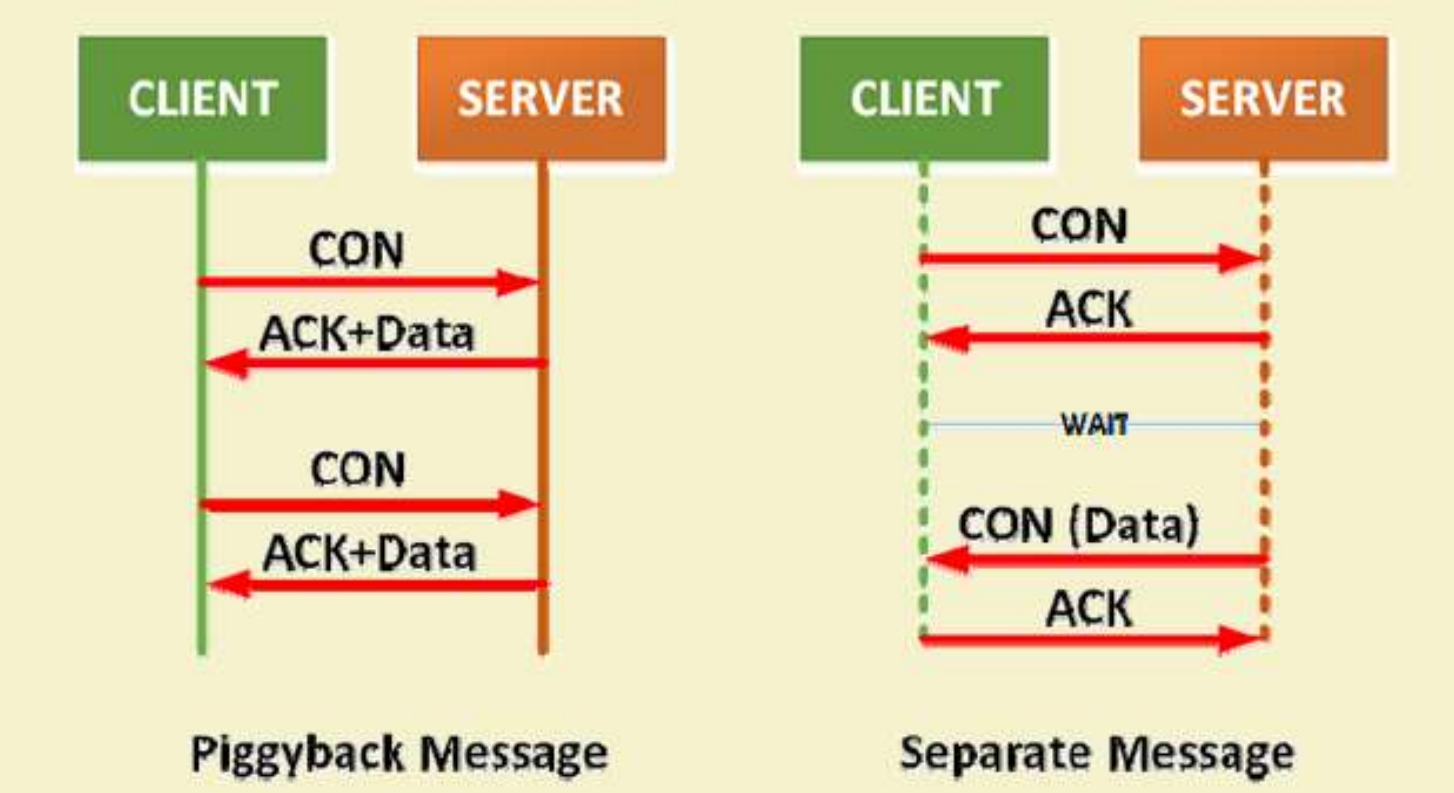
```
{"country":{
        "city":{
         "temp":33,
         "humidity":40
              }
          }
}
```

Navin Reddy

CoAP Request-Response Model

- Confirmable and non-confirmable modes represent the reliable and unreliable transmissions, respectively, while the other modes are used for request/response.
- Piggyback is used for client/server direct communication where the server sends its response directly after receiving the message, i.e., within the acknowledgment message.
- On the other hand, the separate mode is used when the server response comes in a message separate from the acknowledgment, and may take some time to be sent by the server.
- Similar to HTTP, CoAP utilizes GET, PUT, PUSH, DELETE messages requests to retrieve, create, update, and delete, respectively

CoAP Request-Response Model



CoAP Request-Response Model

Features

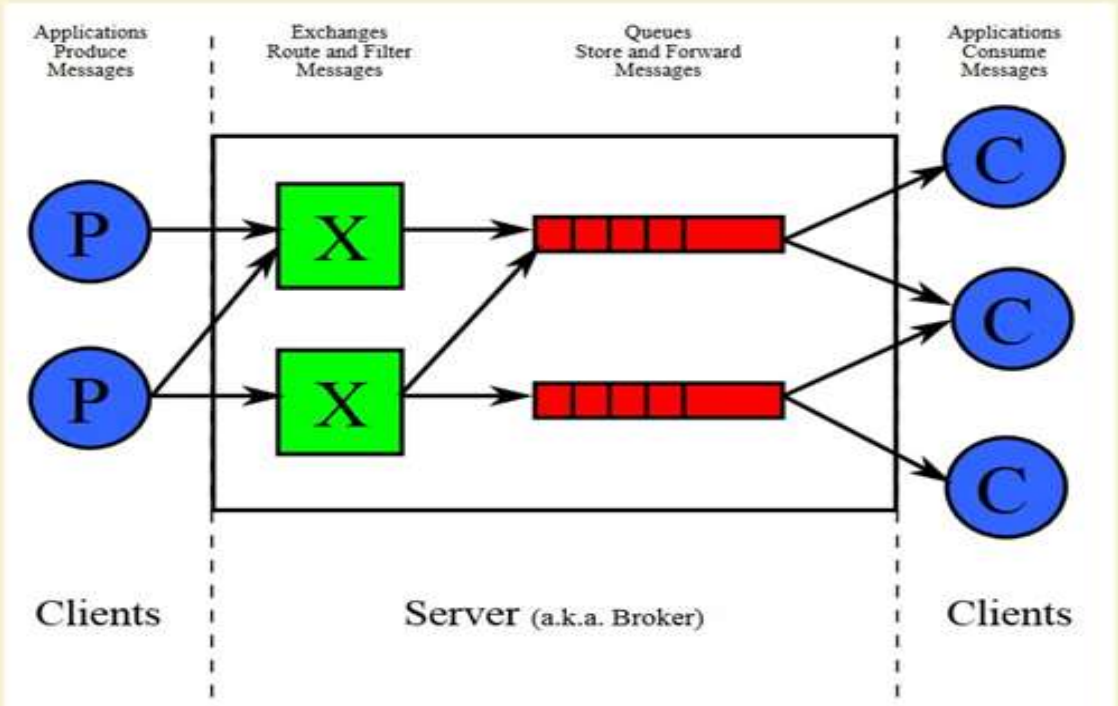
- Reduced overheads and parsing complexity.
- URL and content-type support.
- Support for the discovery of resources provided by known CoAP services.
- Simple subscription for a resource, and resulting push notifications.
- Simple caching based on maximum message age

AMQP

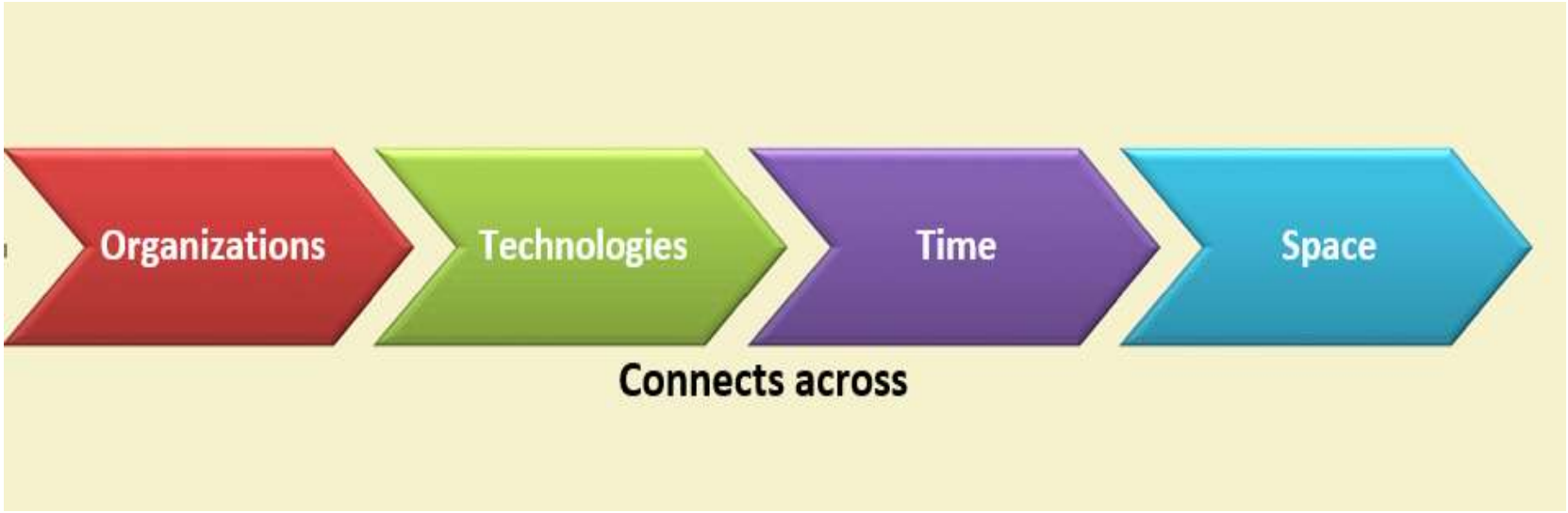
Introduction

- **Advanced Message Queuing Protocol.**
- **Open standard for passing business messages** between applications or organizations.
- Connects between systems and business processes.
- It is a binary application layer protocol.
- Basic unit of data is a *frame*.
- ISO standard: **ISO/IEC 19464**

AMQP


Applications
Produce
Messages
Exchanges
Route and Filter
Messages
Queues
Store and Forward
Messages
Applications
Consume
Messages
P
P
X
X
C
C
C
Clients
Server (a.k.a. Broker)
Clients

AMQP Features



AMQP Features

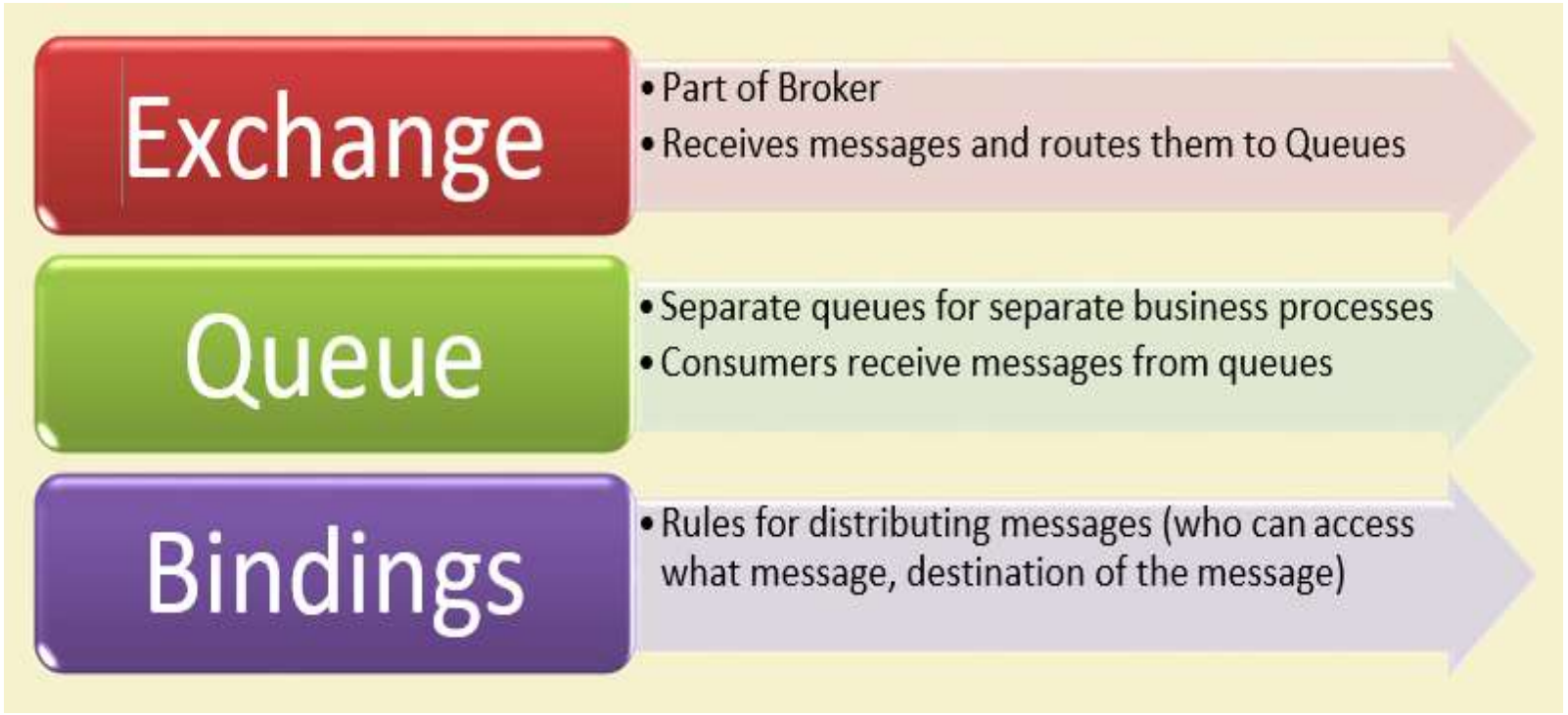
Message Delivery Guarantees

- *At-most-once*
  - each message is delivered once or never
- *At-least-once*
  - each message is certain to be delivered, but may do so multiple times
- *Exactly-once*
  - message will always certainly arrive and do so only once

AMQP Frame Types

- Nine AMQP frame types are defined that are used to initiate, control and tear down the transfer of messages between two peers:
  - Open (connection open)
  - Begin (session open)
  - Attach (initiate new link)
  - Transfer (for sending actual messages)
  - Flow (controls message flow rate)
  - Disposition (Informs the changes in state of transfer)
  - Detach (terminate the link)
  - End (session close)
  - Close (connection close)

AMQP Components



Zigbee

Features of ZigBee

- ✓ Most widely deployed enhancement of IEEE 802.15.4.
- ✓ The ZigBee protocol is defined by **layer 3 and above**. It works with the 802.15.4 layers 1 and 2.
- ✓ The standard uses layers 3 and 4 to define additional communication enhancements.
- ✓ These enhancements include authentication with valid nodes, encryption for security, and a data routing and forwarding capability that enables mesh networking.
- ✓ The most popular use of ZigBee is wireless sensor networks using the mesh topology.

Source: L.Fenzel, "What's The Difference Between IEEE 802.15.4 And ZigBee Wireless?", Electronic Design (Online), Mar. 2013


Application Layer
Presentation Layer
Session Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer
Zigbee
IEEE 802.15.4

Important Components

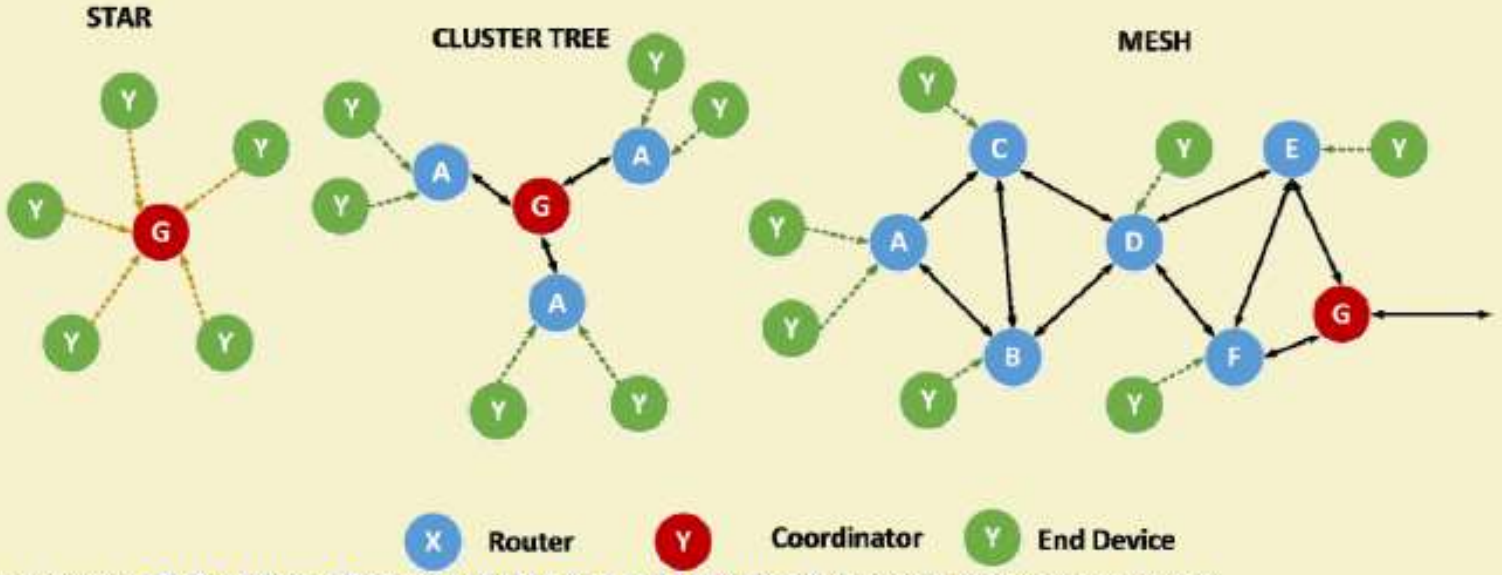
ZDO

- ZigBee Device Object
  (Device management, Security, Policies)

APS

- Application Support Sub-layer
  (Interfacing and control services, bridge between network and other layers)

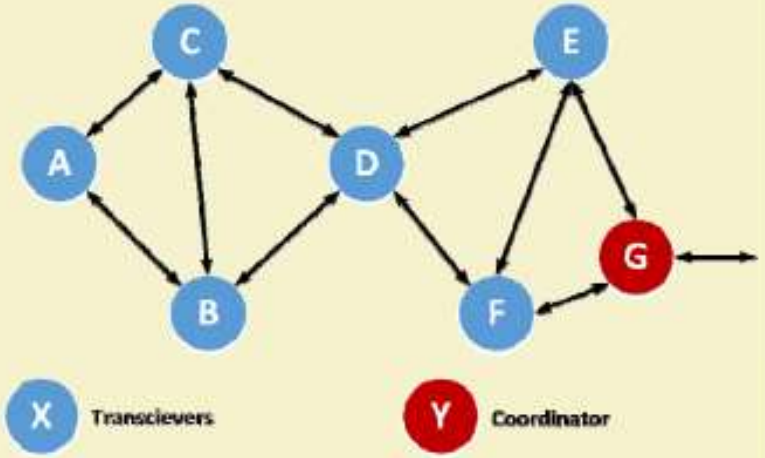
ZigBee Topologies



Source: T. Agarwal, "ZigBee Wireless Technology Architecture and Applications", Electronics Projects Focus (Online)

ZigBee Mesh

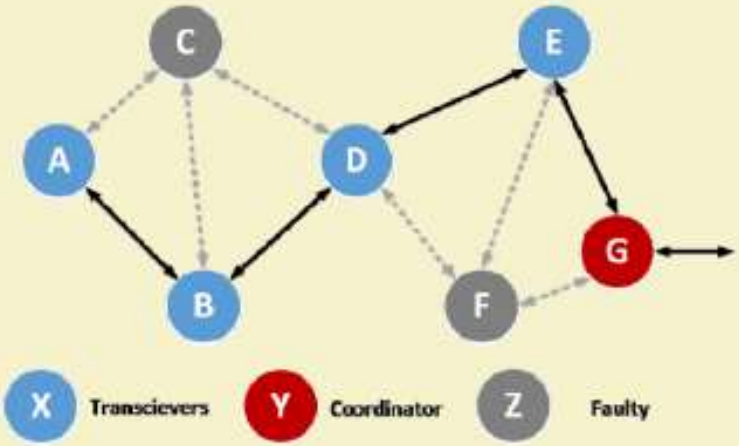
- ✓ In a mesh, any node can communicate with any other node within its range.
- ✓ If nodes are not in range, messages are relayed through intermediate nodes.
- ✓ This allows the network deployment over large areas.



Source: L.Fenzel, "What's The Difference Between IEEE 802.15.4 And ZigBee Wireless?", Electronic Design (Online), Mar. 2013

ZigBee Mesh (Contd.)

- ✓ Meshes have increased network reliability.
- ✓ For example, if nodes C and F are down, the message packets from A can still be relayed to G via B and E.
- ✓ ZigBee mesh networks are self-configuring and self-healing.



Source: L.Fenzel, "What's The Difference Between IEEE 802.15.4 And ZigBee Wireless?", Electronic Design (Online), Mar. 2013

ZigBee Types

- ✓ *ZigBee Coordinator (ZC)*:
  - The Coordinator forms the root of the ZigBee network tree and might act as a bridge between networks.
  - There is a single ZigBee Coordinator in each network, which originally initiates the network.
  - It stores information about the network under it and outside it.
  - It acts as a Trust Center & repository for security keys.

Sources:

- *"Wireless Sensor Networks Research Group". Sensor-networks.org. 2010-04-15.*
- *"Wireless Sensor Networks Research Group". Sensor-networks.org. 2009-02-05.*

ZigBee Types

- ✓ *ZigBee Router (ZR)*:
  - Capable of running applications, as well as relaying information between nodes connected to it.
- ✓ *ZigBee End Device (ZED)*:
  - It contains just enough functionality to talk to the parent node, and it cannot relay data from other devices.
  - This allows the node to be asleep a significant amount of the time thereby enhancing battery life.
  - Memory requirements and cost of ZEDs are quite low, as compared to ZR or ZC.

Sources:

- *"Wireless Sensor Networks Research Group". Sensor-networks.org. 2010-04-15.*
- *"Wireless Sensor Networks Research Group". Sensor-networks.org. 2009-02-05.*

ZigBee Network Layer

- ✓ The network layer uses Ad Hoc On-Demand Distance Vector (AODV) routing.
- ✓ To find the final destination, the AODV broadcasts a route request to all its immediate neighbors.
- ✓ The neighbors relay the same information to their neighbors, eventually spreading the request throughout the network.
- ✓ Upon discovery of the destination, a low-cost path is calculated and informed to the requesting device via unicast messaging.

Source: "Zigbee", Wikipedia (Online)

Applications

- ✓ Building automation
- ✓ Remote control (RF4CE or RF for consumer electronics)
- ✓ Smart energy for home energy monitoring
- ✓ Health care for medical and fitness monitoring
- ✓ Home automation for control of smart homes
- ✓ Light Link for control of LED lighting
- ✓ Telecom services

Source: L.Fenzel, "What's The Difference Between IEEE 802.15.4 And ZigBee Wireless?", Electronic Design (Online), Mar. 2013

XMPP

Introduction

- ✓ **XMPP – Extensible Messaging and Presence Protocol.**
- ✓ A communication protocol for **message-oriented middleware** based on XML (Extensible Markup Language).
- ✓ Real-time exchange of structured data.
- ✓ It is an open standard protocol.

Source: "XMPP", Wikipedia (Online)

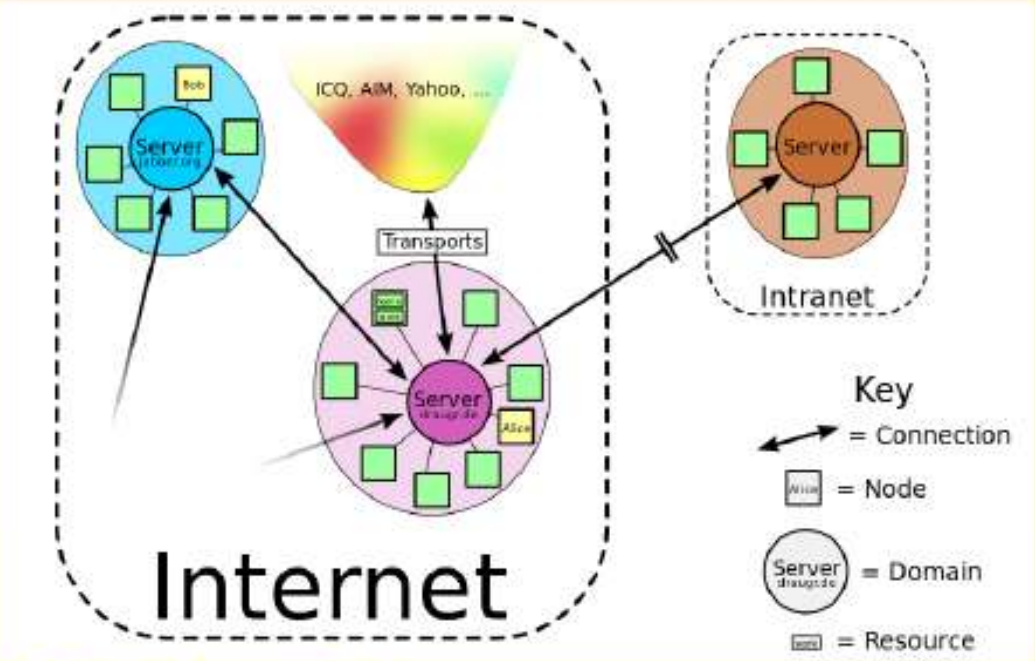
- ✓ XMPP uses a **client-server architecture.**
- ✓ As the model is **decentralized**, no central server is required.
- ✓ XMPP provides for the **discovery of services** residing locally or across a network, and the **availability information** of these services.
- ✓ Well-suited for cloud computing where virtual machines, networks, and firewalls would otherwise present obstacles to alternative service discovery and presence-based solutions.
- ✓ Open means to support machine-to-machine or peer-to-peer communications across a diverse set of networks.

Source: "XMPP", Wikipedia (Online)

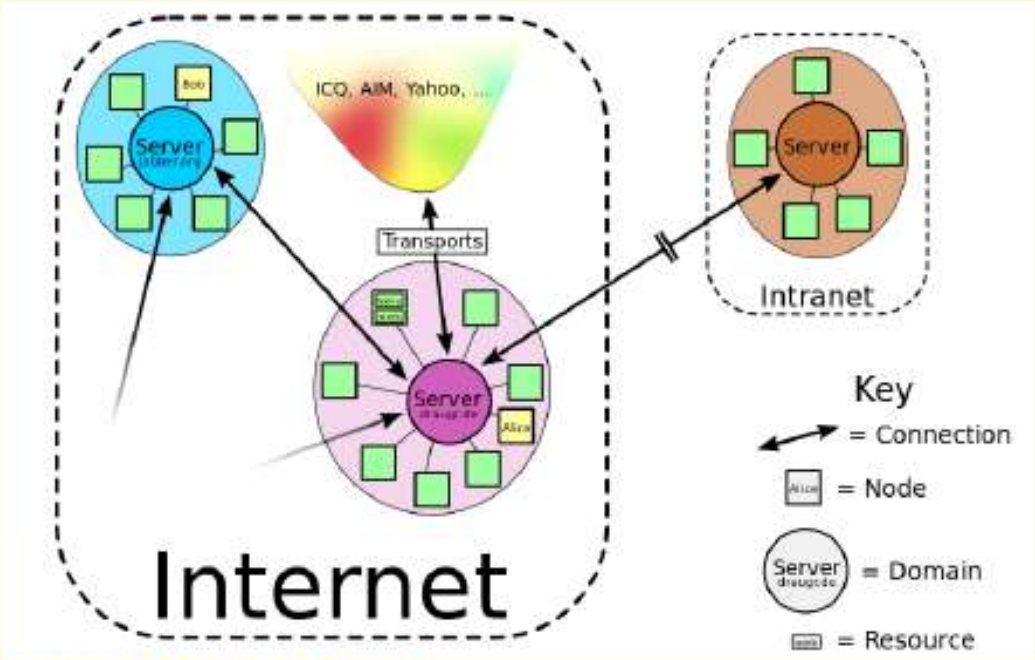
Highlights

- ✓ Decentralization – No central server; anyone can run their own XMPP server.
- ✓ Open standards – No royalties or granted permissions are required to implement these specifications
- ✓ Security – Authentication, encryption, etc.
- ✓ Flexibility – Supports interoperability

Source: "XMPP", Wikipedia (Online)



Source: "JabberNetwork.svg", Wikimedia Commons (Online)



Source: "JabberNetwork.svg", Wikimedia Commons (Online)

Core XMPP Technologies

Core
- information about the core XMPP technologies for XML streaming

Jingle
- multimedia signalling for voice, video, file transfer

Multi-user Chat
- flexible, multi-party communication

PubSub
- alerts and notifications for data syndication

BOSH
- HTTP binding for XMPP

Source: "XMPP: Technology Overview", XMPP.org (Online)

Weaknesses

- ✓ Does not support QoS.
- ✓ Text based communications induces higher network overheads.
- ✓ Binary data must be first encoded to base64 before transmission.

Weaknesses

- ✓ Does not support QoS.
- ✓ Text based communications induces higher network overheads.
- ✓ Binary data must be first encoded to base64 before transmission.

Applications

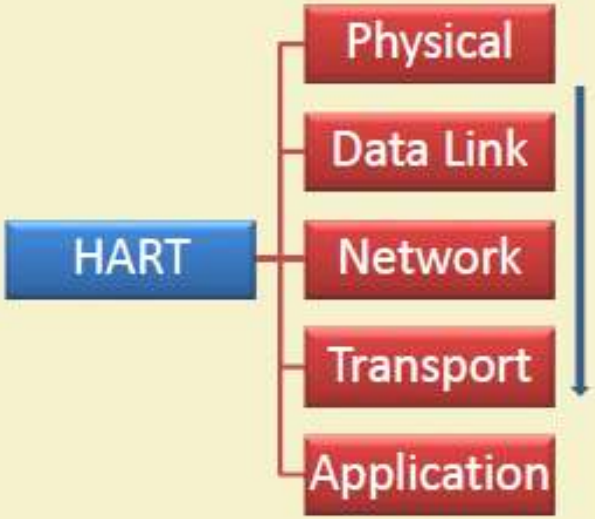
- ✓ Publish-subscribe systems
- ✓ Signaling for VoIP
- ✓ Video
- ✓ File transfer
- ✓ Gaming
- ✓ Internet of Things applications
  - Smart grid
  - Social networking services

HART & Wireless HART

Introduction

- ✓ WirelessHART is the latest release of **Highway Addressable Remote Transducer** (HART) Protocol.
- ✓ HART standard was developed for **networked smart field devices.**
- ✓ The wireless protocol makes the implementation of HART **cheaper and easier.**
- ✓ HART encompasses the most number of field devices incorporated in any field network.

- ✓ Wireless HART enables device placements more accessible and cheaper– such as the top of a reaction tank, inside a pipe, or at widely separated warehouses.
- ✓ Main difference between wired and unwired versions is in the physical, data link and network layers.
- ✓ Wired HART lacks a network layer.



Source: A. Feng, "WirelessHART- Made Easy", AwiaTech Blog (Online), Nov. 2011

HART Physical Layer

- ✓ Derived from IEEE 802.15.4 protocol.
- ✓ It operates only in the 2.4 GHz ISM band.
- ✓ Employs and exploits 15 channels of the band to increase reliability.

Source: A. Feng, "WirelessHART- Made Easy", AwiaTech Blog (Online), Nov. 2011

HART Data Link Layer

- ✓ Collision free and deterministic communication achieved by means of super-frames and TDMA.
- ✓ Super-frames consist of grouped 10ms wide timeslots.
- ✓ Super-frames control the timing of transmission to ensure collision free and reliable communication.
- ✓ This layer incorporates channel hopping and channel blacklisting to increase reliability and security.
- ✓ Channel blacklisting identifies channels consistently affected by interference and removes them from use.

Source: A. Feng, "WirelessHART- Made Easy", AwiaTech Blog (Online), Nov. 2011

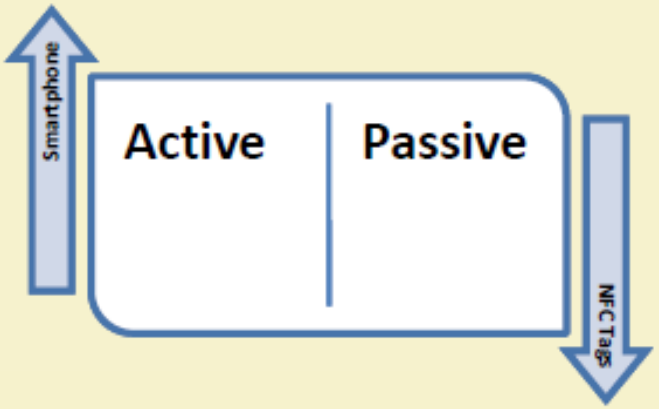
HART Congestion Control

- ✓ Restricted to 2.4Ghz ISM band with channel 26 removed, due to its restricted usage in certain areas.
- ✓ Interference-prone channels avoided by using channel switching post every transmission.
- ✓ Transmissions synchronized using 10ms slots.
- ✓ During each slot, all available channels can be utilized by the various nodes in the network allowing for the propagation of 15 packets through the network at a time, which also minimizes the risk of collisions.

Source: A. Feng, "WirelessHART- Made Easy", AwiaTech Blog (Online), Nov. 2011

NFC Types

- ✓ **Passive devices** contain information which is readable by other devices, however it cannot read information itself.
- ✓ NFC tags found in supermarket products are examples of passive NFC.
- ✓ **Active devices** are able to collect as well as transmit information.
- ✓ Smartphones are a good example of active devices.



Source: "How NFC Works", NFC (Online)

NFC Specifications

- ✓ NFC's data-transmission frequency is 13.56MHz.
- ✓ NFC can transmit data at a rate of either 106, 212 or 424 Kbps (kilobits per second).
- ✓ Tags typically store between 96 and 512 bytes of data.
- ✓ Communication range is less than 20cms.

Source: "Inside NFC: how near field communication works", APC (Online), Aug. 2011

NFC Applications

- ✓ Smartphone based payments.
- ✓ Parcel tracking.
- ✓ Information tags in posters and advertisements.
- ✓ Computer game synchronized toys.
- ✓ Low-power home automation systems.

Bluetooth

Bluetooth

Features

- ✓ Bluetooth technology operates in the unlicensed industrial, scientific and medical (ISM) band at 2.4 to 2.485 GHZ.
- ✓ Uses spread spectrum hopping, full-duplex signal at a nominal rate of 1600 hops/sec.
- ✓ Bluetooth supports 1Mbps data rate for version 1.2 and 3Mbps data rate for Version 2.0 combined with Error Data Rate.

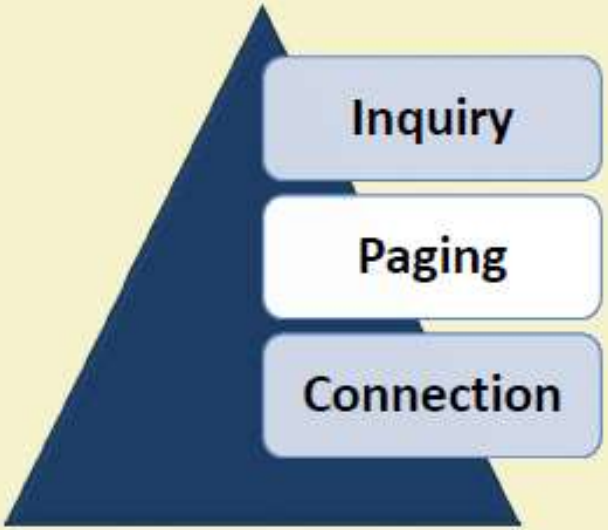
Source: "Wireless Communication - Bluetooth", Tutorials Point (Online)

Features

- ✓ Bluetooth operating range depends on the device:
  - **Class 3** radios have a range of up to 1 meter or 3 feet
  - **Class 2** radios are most commonly found in mobile devices have a range of 10 meters or 30 feet
  - **Class 1** radios are used primarily in industrial use cases have a range of 100 meters or 300 feet.

Source: "Wireless Communication - Bluetooth", Tutorials Point (Online)

Connection Establishment



Inquiry run by one Bluetooth device to try to **discover other devices near it.**

Process of **forming a connection** between two Bluetooth devices.

A device either actively **participates** in the network or enters a low-power sleep mode.

Source: "Bluetooth Basics", Tutorials, Sparkfun.com (Online)

Modes

Actively transmitting or receiving data.

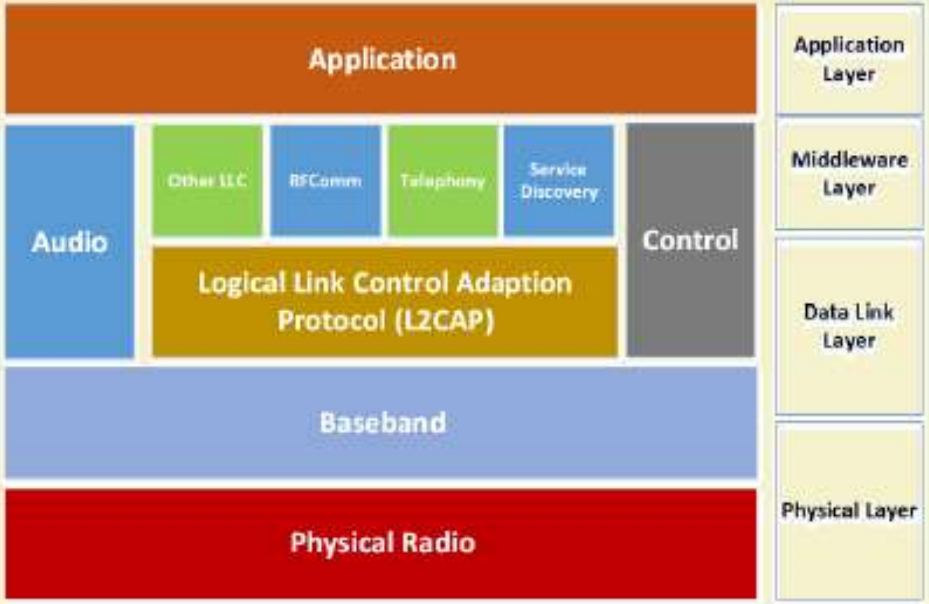
Sleeps and only listens for transmissions at a set interval .

Power-saving mode where a device sleeps for a defined period and then returns back to active mode .

Slave will become inactive until the master tells it to wake back up.

Source: "Bluetooth Basics", Tutorials, Sparkfun.com (Online)

Protocol Stack



Baseband

- ✓ Physical layer of the Bluetooth.
- ✓ Manages physical channels and links.
- ✓ Other services include:
  - Error correction
  - Data whitening
  - Hop selection
  - Bluetooth security
- ✓ Manages asynchronous and synchronous links.
- ✓ Handles packets, paging and inquiry.

Source: "Bluetooth", Wikipedia (Online)

L2CAP

- ✓ The Logical Link Control and Adaptation Protocol (L2CAP).
- ✓ Layered over the Baseband Protocol and resides in the data link layer.
- ✓ Used to multiplex multiple logical connections between two devices.
- ✓ Provides connection-oriented and connectionless data services to upper layer protocols.
- ✓ Provides:
  - Protocol multiplexing capability
  - Segmentation and reassembly operation
  - Group abstractions

Source: "Bluetooth", Wikipedia (Online)

RFComm

- ✓ Radio Frequency Communications (RFCOMM).
- ✓ It is a cable replacement protocol used for generating a virtual serial data stream.
- ✓ RFCOMM provides for binary data transport .
- ✓ Emulates EIA-232 (formerly RS-232) control signals over the Bluetooth baseband layer, i.e. it is a serial port emulation.
- ✓ RFCOMM provides a simple reliable data stream to the user, similar to TCP.
- ✓ Supports up to 60 simultaneous connections between two BT devices.

Source: "Bluetooth", Wikipedia (Online)

Service Discovery Protocol (SDP)

- ✓ Enables applications to discover available services and their features.
- ✓ Addresses the unique characteristics of the Bluetooth environment such as, dynamic changes in the quality of services in RF proximity of devices in motion.
- ✓ Can function over a reliable packet transfer protocol.
- ✓ Uses a request/response model.

Source: "Bluetooth", Wikipedia (Online)

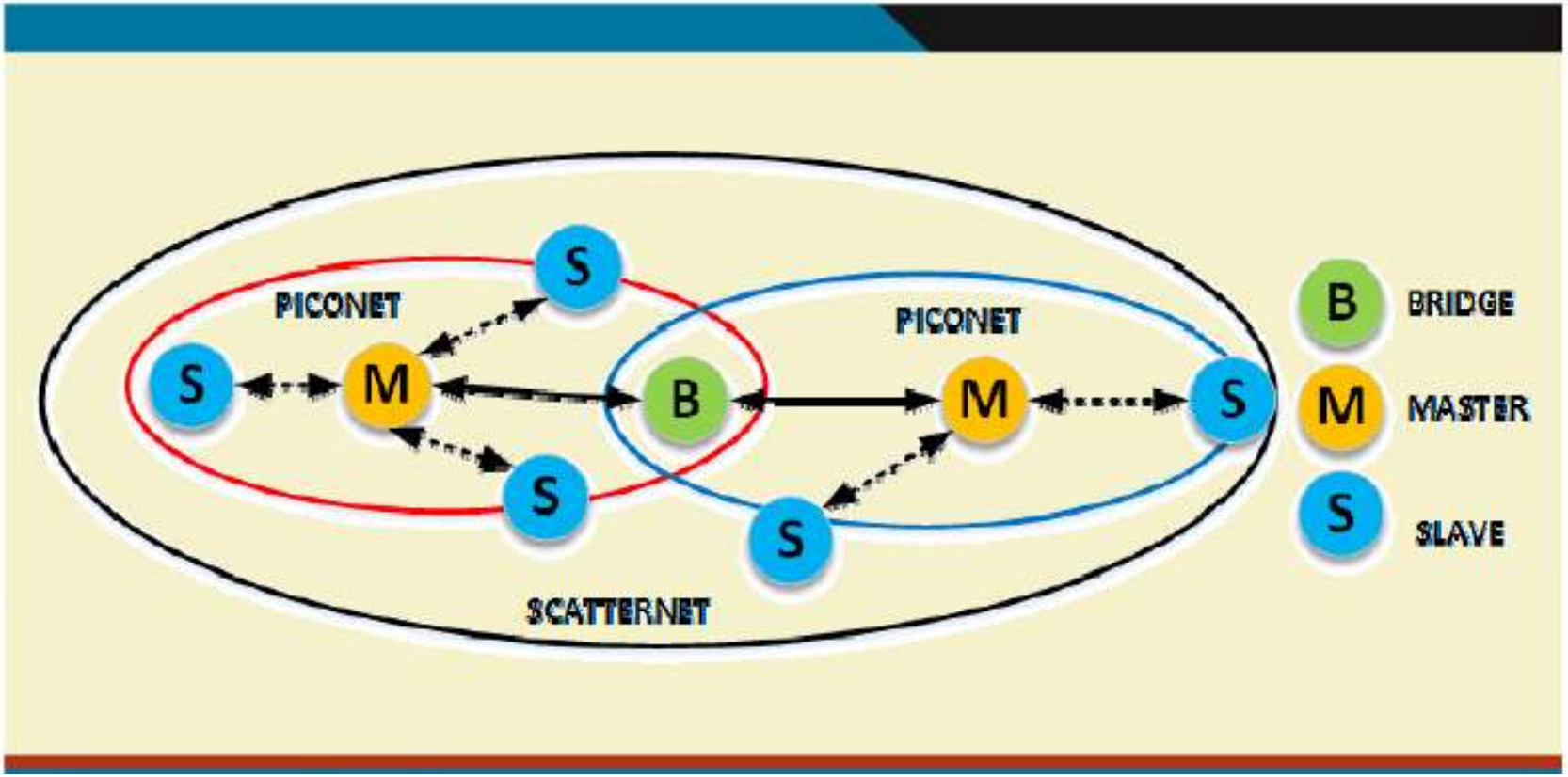
Piconets

- ✓ Bluetooth enabled electronic devices connect and communicate wirelessly through short range networks known as **Piconets**.
- ✓ Bluetooth devices exist in small ad-hoc configurations with the ability to act either as master or slave.
- ✓ Provisions are in place, which allow for a **master** and a **slave** to switch their roles.
- ✓ The simplest configuration is a point to point configuration with one master and one slave.

Source: "Wireless Communication - Bluetooth", Tutorials Point (Online)

- ✓ When more than two Bluetooth devices communicate with one another, it is called a **PICONET**.
- ✓ A Piconet can contain up to seven slaves clustered around a single master.
- ✓ The device that initializes establishment of the Piconet becomes the **master**.
- ✓ The master is responsible for transmission control by dividing the network into a series of time slots amongst the network members, as a part of **time division multiplexing** scheme.

Source: "Wireless Communication - Bluetooth", Tutorials Point (Online)


PICONET
PICONET
SCATTERNET
S
M
B
S
S
M
S
S
B
BRIDGE
M
MASTER
S
SLAVE

Features of Piconet

- ✓ Within a Piconet, the clock and unique **48-bit address** of master determines the timing of various devices and the frequency hopping sequence of individual devices.
- ✓ Each Piconet device supports 7 simultaneous connections to other devices.
- ✓ Each device can communicate with several piconets simultaneously.
- ✓ Piconets are established dynamically and automatically as Bluetooth enabled devices enter and leave piconets.

Source: "Wireless Communication - Bluetooth", Tutorials Point (Online)

- ✓ There is no direct connection between the slaves.
- ✓ All connections are either master-to-slave or slave-to-master.
- ✓ Slaves are allowed to transmit once these have been polled by the master.
- ✓ Transmission starts in the slave-to-master time slot immediately following a polling packet from the master.
- ✓ A device can be a member of two or more Piconets.

Source: "Wireless Communication - Bluetooth", Tutorials Point (Online)

- ✓ A device can be a slave in one Piconet and master in another. It however cannot be a master in more than once Piconets.
- ✓ Devices in adjacent Piconets provide a bridge to support inner-Piconet connections, allowing assemblies of linked Piconets to form a physically extensible communication infrastructure known as **Scatternet**.

Source: "Wireless Communication - Bluetooth", Tutorials Point (Online)

Applications

- ✓ Audio players
- ✓ Home automation
- ✓ Smartphones
- ✓ Toys
- ✓ Hands free headphones
- ✓ Sensor networks

Z Wave

Introduction

- ✓ Zwave (or Z wave or Z-wave) is a protocol for communication among devices used for home automation.
- ✓ It uses RF for signaling and control.
- ✓ Operating frequency is 908.42 MHz in the US & 868.42 MHz in Europe.
- ✓ Mesh network topology is the main mode of operation, and can support 232 nodes in a network.

Source: "What is Z-Wave?", Smart Home (Online)

Zwave Global Operating Frequency

Frequency in MHz	Used in
865.2	India
868.1	Malaysia
868.42 ; 869.85	Europe
868.4	China, Korea
869.0	Russia
908.4 ; 916.0	USA
915.0 - 926.0	Israel
919.8	Hong Kong
921.4 ; 919.8	Australia, New Zealand
922.0 - 926.0	Japan

Source: "Z-Wave", Wikipedia (Onlire)

- ✓ Zwave utilizes GFSK modulation and Manchester channel encoding.
- ✓ A central network controller device sets-up and manages a Zwave network.
- ✓ Each logical Zwave network has 1 Home (Network) ID and multiple node IDs for the devices in it.
- ✓ Nodes with different Home IDs cannot communicate with each other.
- ✓ Network ID length=4 Bytes, Node ID length=1 Byte.

Source: "What is Z-Wave?", Smart Home (Online)

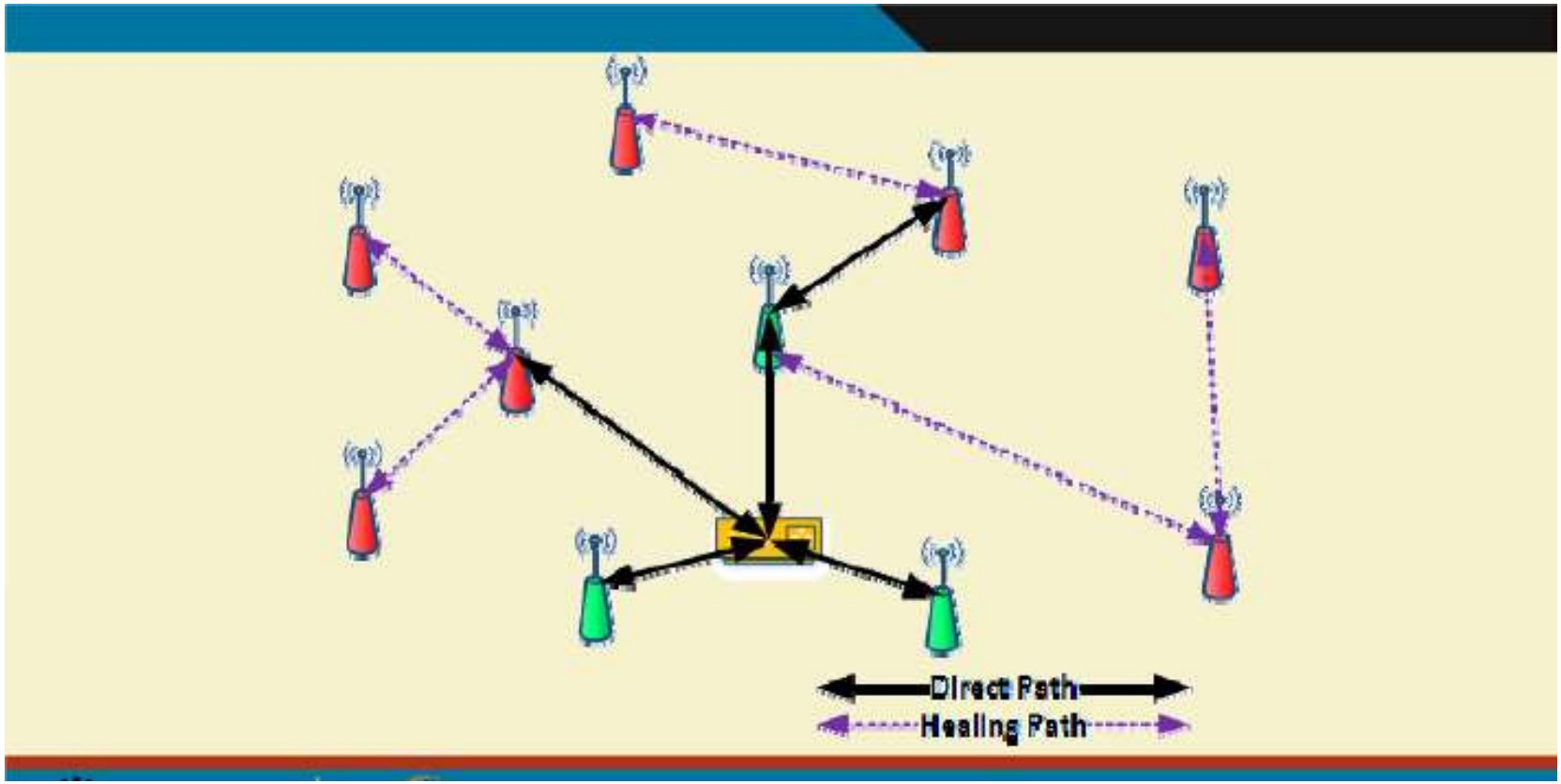
GFSK

- ✓ Gaussian Frequency Shift Keying.
- ✓ Baseband pulses are passed through a Gaussian filter prior to modulation.
- ✓ Filtering operation smoothens the pulses consisting of streams of -1 and 1, and is known as **Pulse shaping**.
- ✓ Pulse shaping limits the modulated spectrum width.


BEDROOM #3
DINING ROOM
KITCHEN
BATH #2
BATH #1
BEDROOM #2
LIVING ROOM
BEDROOM #1
Zwave Controller
Zwave Node

- ✓ Uses source routed network mesh topology using 1 primary controller.
- ✓ Devices communicate with one another when in range.
- ✓ When devices are not in range, messages are routed though different nodes to bypass obstructions created by household appliances or layout.
- ✓ This process of bypassing radio dead-spots is done using a message called **Healing.**
- ✓ As Zwave uses a source routed static network, mobile devices are excluded from the network and only static devices are considered.

Source: "What is Z-Wave?", Smart Home (Online)


Direct Path
Healing Path

Zwave vs. Zigbee

Zwave

- ✓ User friendly and provides a simple system that users can set up themselves.
- ✓ Ideal for someone with a basic understanding of technology who wants to keep their home automation secure, efficient, simple to use, and easy to maintain.

Zigbee

- ✓ Requires so little power that devices can last up to seven years on one set of batteries.
- ✓ Ideal for technology experts who want a system they can customize with their preferences and install themselves.

Source: Sarah Brown, "ZigBee vs. Z-Wave Review: What's the Best Option for You?", The SafeWise Report (Online), Mar 2016

Zwave vs. Zigbee

Zwave

- ✓ Expensive.
- ✓ Nine out of ten leading security and communication companies in the U.S. use Z-Wave in their smart home solutions

Zigbee

- ✓ Cheaper than Zwave.
- ✓ ZigBee Alliance consists of nearly 400 member organizations that use, develop, and improve ZigBee's open-standard wireless connection

Source: Sarah Brown, "ZigBee vs. Z-Wave Review: What's the Best Option for You?", The SafeWise Report (Online), Mar 2016

ISA 100.11A

Introduction

- ✓ International Society of Automation.
- ✓ Designed mainly for large scale industrial complexes and plants.
- ✓ More than 1 billion devices use ISA 100.11A
- ✓ ISA 100.11A is designed to support native and tunneled application layers.
- ✓ Various transport services, including 'reliable,' 'best effort,' 'real-time' are offered.

Source: "The ISA 100 Standards : Overview and Status" ISA, 2008

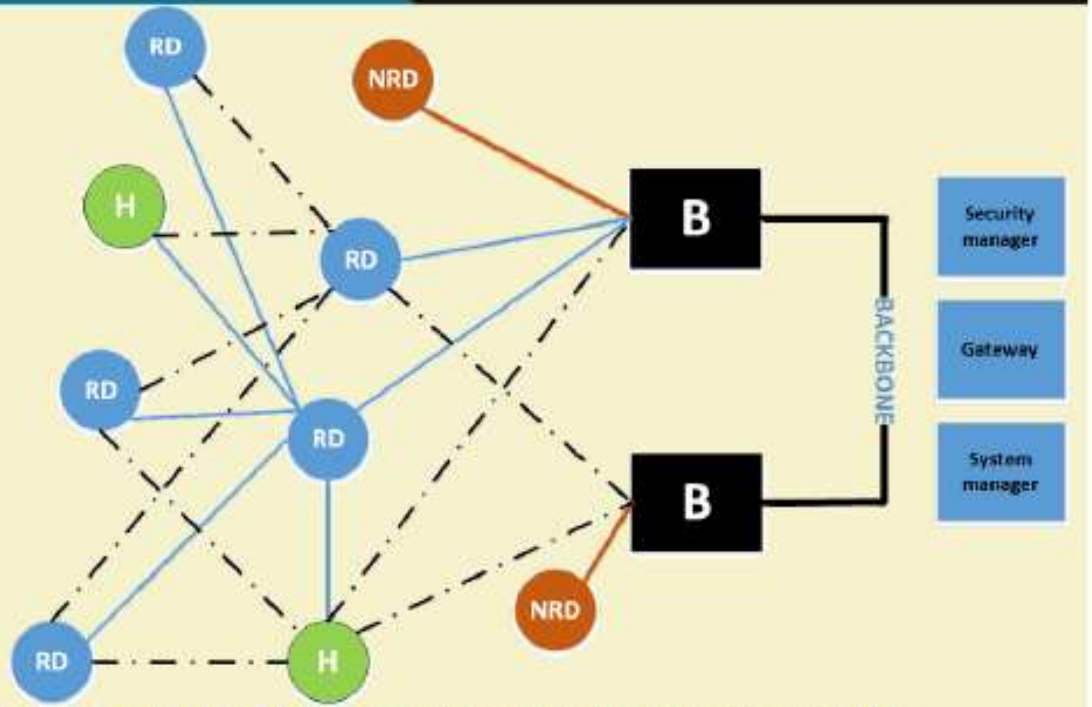
- ✓ Network and transport layers are based on TCP or UDP / IPv6.
- ✓ Data link layer supports mesh routing and Frequency hopping.
- ✓ Physical and MAC layers are based on IEEE 802.15.4
- ✓ Topologies allowed are:
  - Star/tree
  - Mesh
- ✓ Permitted networks include:
  - Radio link
  - ISA over Ethernet
  - Field buses

Source: Cambridge Whitepaper, http://portal.etsi.org/docbox/Workshop/2008/200812_WIRELESSFACTORY/CAMBRIDGE_WHITTAKER.pdf

- ✓ Application Support Layer delivers communications services to user and management processes.
- ✓ It can pass objects (methods, attributes) natively within the ISA 100.11A protocol.
- ✓ A tunneling mode is available to allow legacy data through the ISA100.11A network.

Source: Tim Whittaker , "What do we expect from Wireless in the Factory?"Cambridge Whitepaper, Cambridge Consultants, 2008

- ✓ RD=routing device
- ✓ NRD=Non-routing device
- ✓ H=Handheld device
- ✓ B=backbone device



Source: Tim Whittaker , "What do we expect from Wireless in the Factory?"Cambridge Whitepaper, Cambridge Consultants, 2008

Features

- ✓ Flexibility
- ✓ Support for multiple protocols
- ✓ Use of open standards
- ✓ Support for multiple applications
- ✓ Reliability (error detection, channel hopping)
- ✓ Determinism (TDMA, QoS support)
- ✓ Security

Security

- ✓ Security is fully built-in to the standard.
- ✓ Authentication and confidentiality services are independently available.
- ✓ A network security manager manages and distributes keys.
- ✓ Twin data security steps in each node:
  - Data link layer encrypts each hop.
  - Transport layer secures peer-to-peer communications.

Source: Tim Whittaker , "What do we expect from Wireless in the Factory?"Cambridge Whitepaper, Cambridge Consultants, 2008

ISA100.11A Usage Classes

Category	Class	Application	Description
Safety	0	Emergency action	Always critical
Control	1	Closed loop regulatory control	Often critical
	2	Closed loop supervisory control	Usually non-critical
	3	Open loop control	Human-in-the-loop
Monitoring	4	Alerting	Short term operational consequence
	5	Logging/ Downloading	No immediate operational consequence

Thank You!!