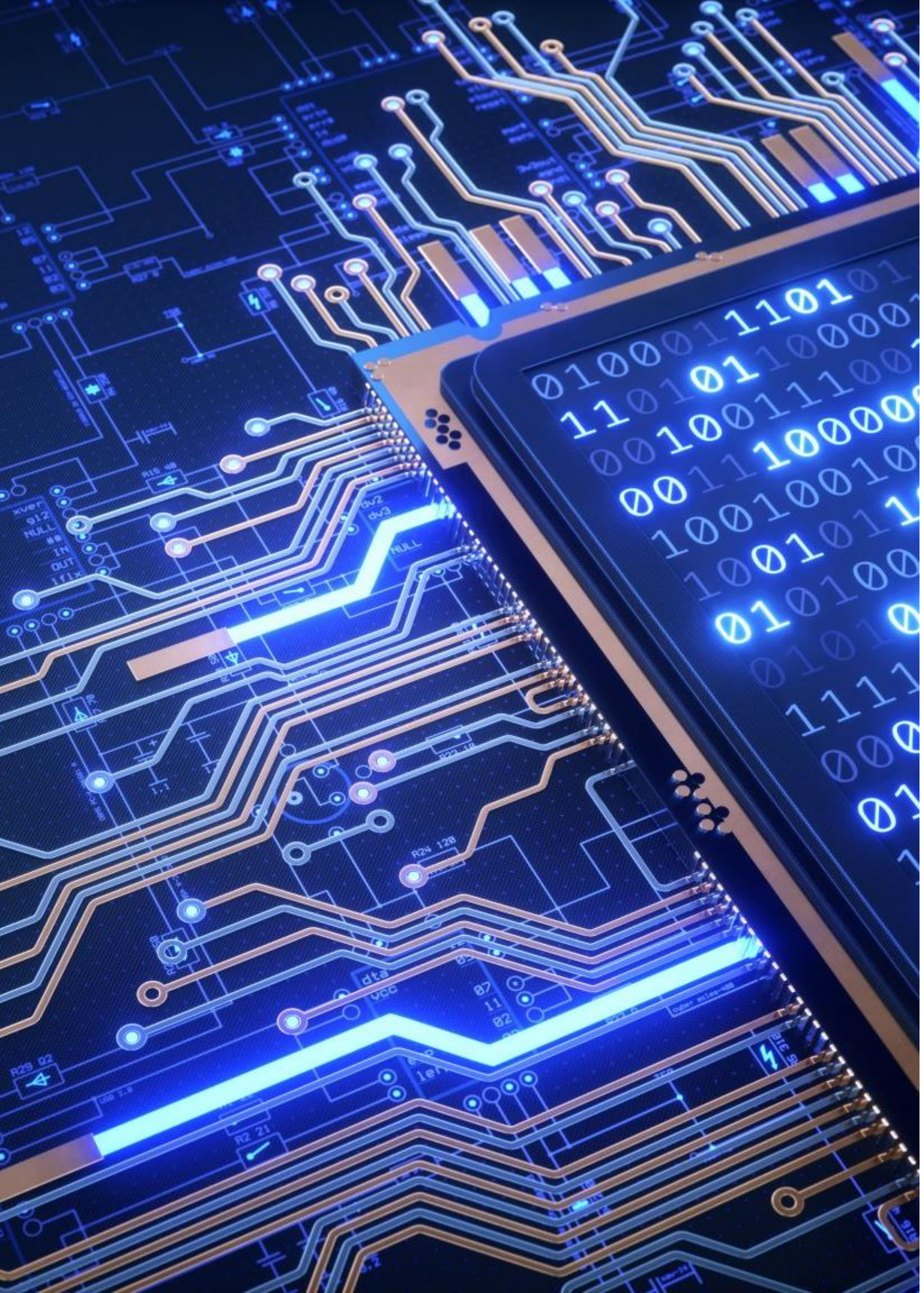
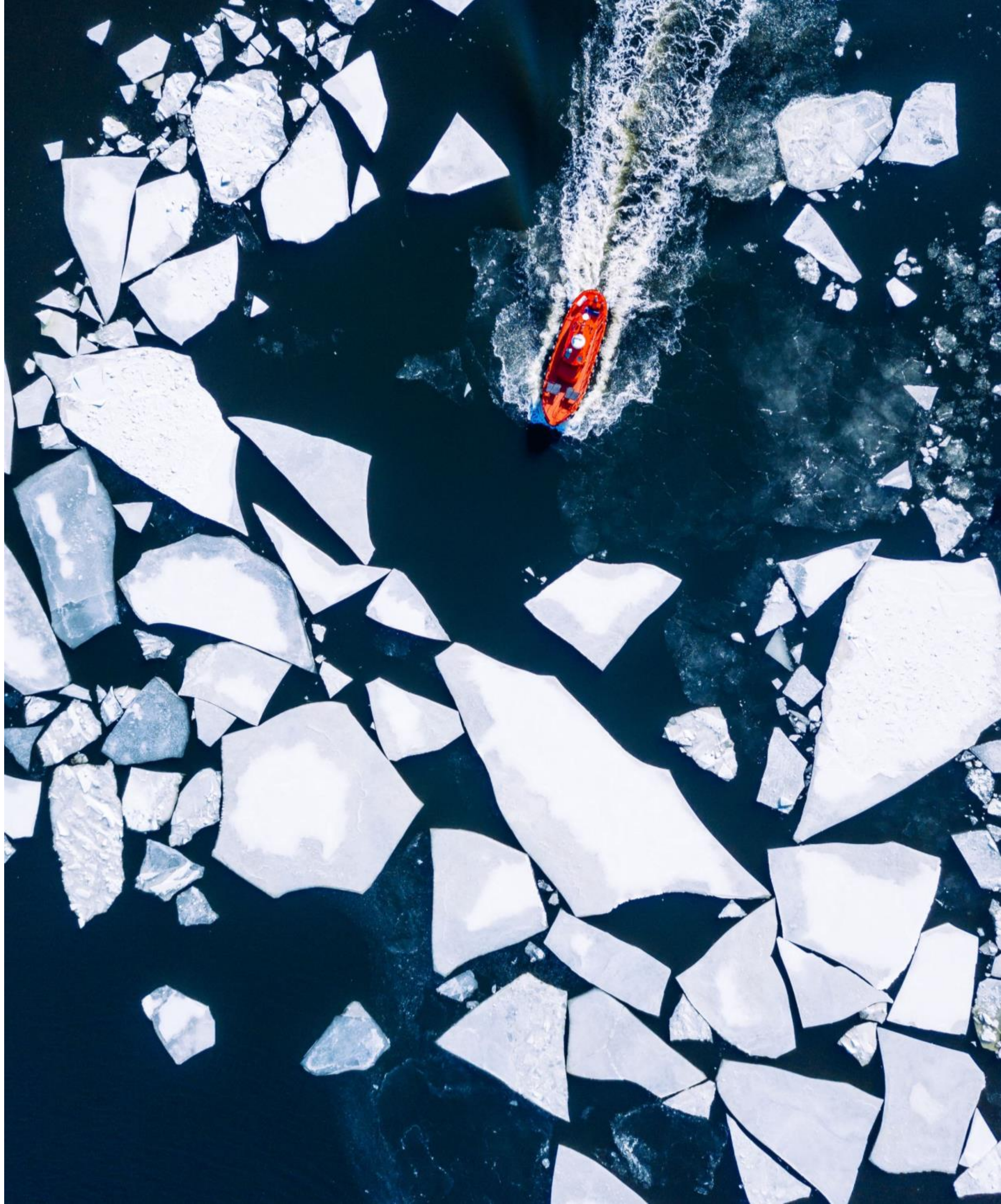


# IOT SESSION 1

Session 1

# INTROS AND ICEBREAKER



# ABOUT ME

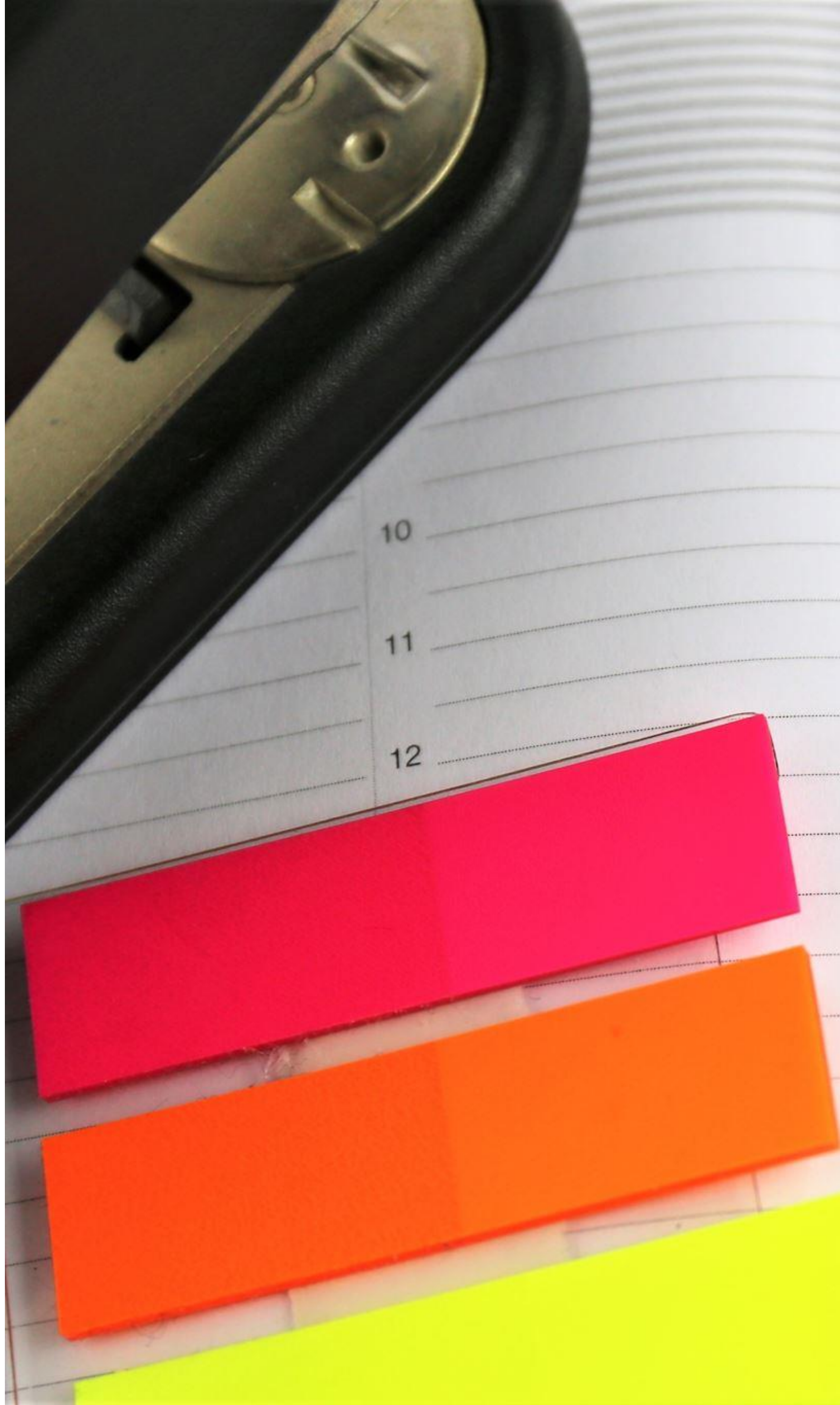
Ritvi Mishra

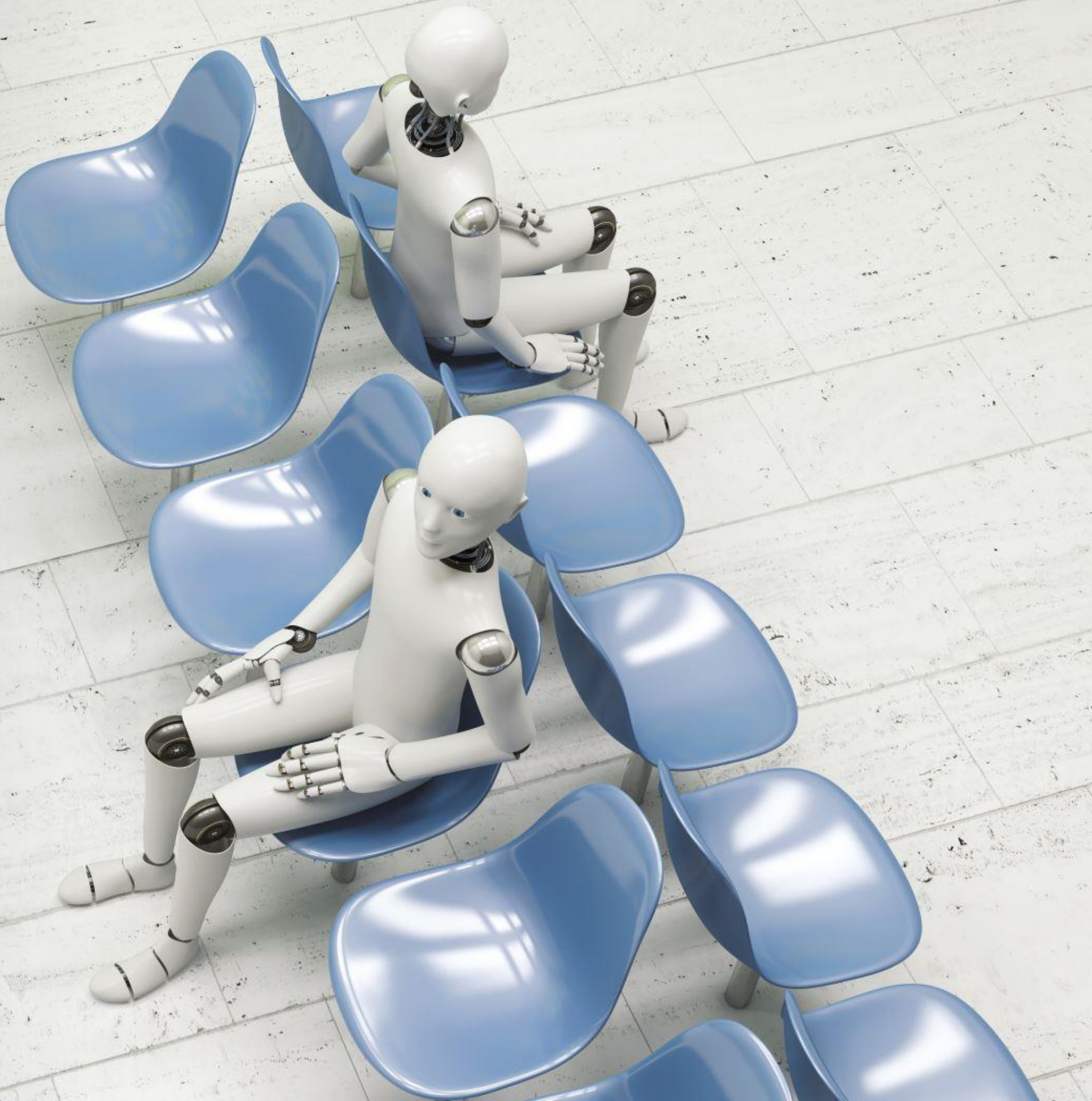
- Maker
- Background in IoT and Embedded Systems
- 4 Internships in Firmware and IoT design
- Passionate Community Builder and Lead
- MLH Coach:
  - Join me this week @ https://www.hackp.ac/ritvi
- Participated in over 35 Hackathons
- Find me on these links:
  - https://www.linkedin.com/in/ritvimishra  Ritvi Mishra
  - https://www.twitter.com/frenzyritz13
  - https://www.instagram.com/frenzy.works

# SESSION 1

# AGENDA

- ~~Intros and icebreakers 10min~~
- Outcome 5min
- What is IoT 5min
- Components of IoT 10min
- IoT and IIoT 5min
- Devices and architectures (Overview) 5min
- Embedded systems and RTOS (Overview) 5min
- Azure Cloud Setup 10min
- Q & A 10min
- Next Steps 10min





# SESSION 2+3

# TOPICS

## Session 2: Protocols, design and setting up the cloud

- Protocols Overview
- Choosing Protocols
- Choosing the Device
- Designing the system

## Session 3 : Getting everything connected and up and running

- Simulating the system
- Power constraints
- Connecting and Deploying the project

# OUTCOME

# WHAT YOU WILL LEARN

A deployed IoT project, with a good understanding of:

- RTOS

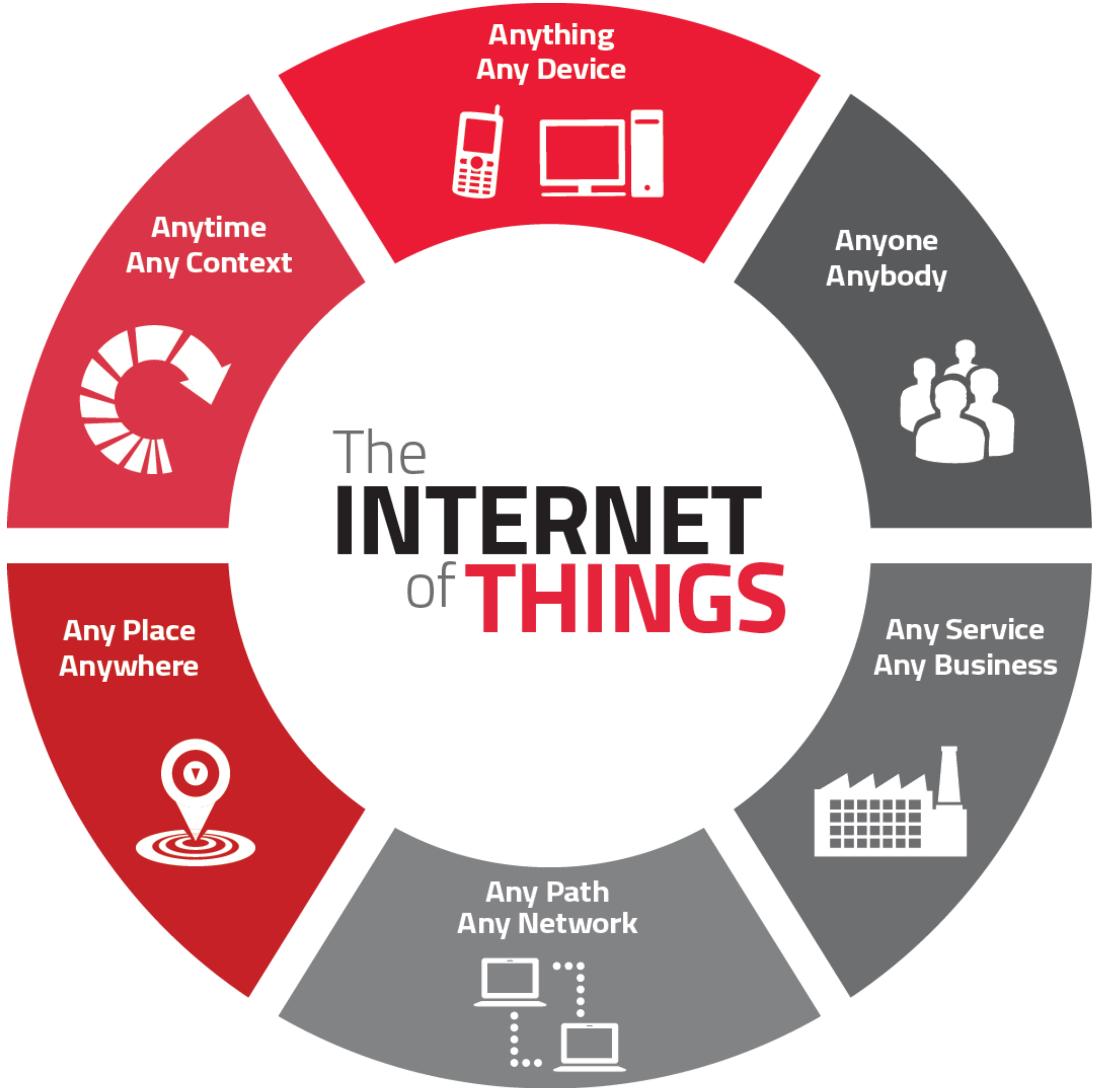
- Cloud Connectivity

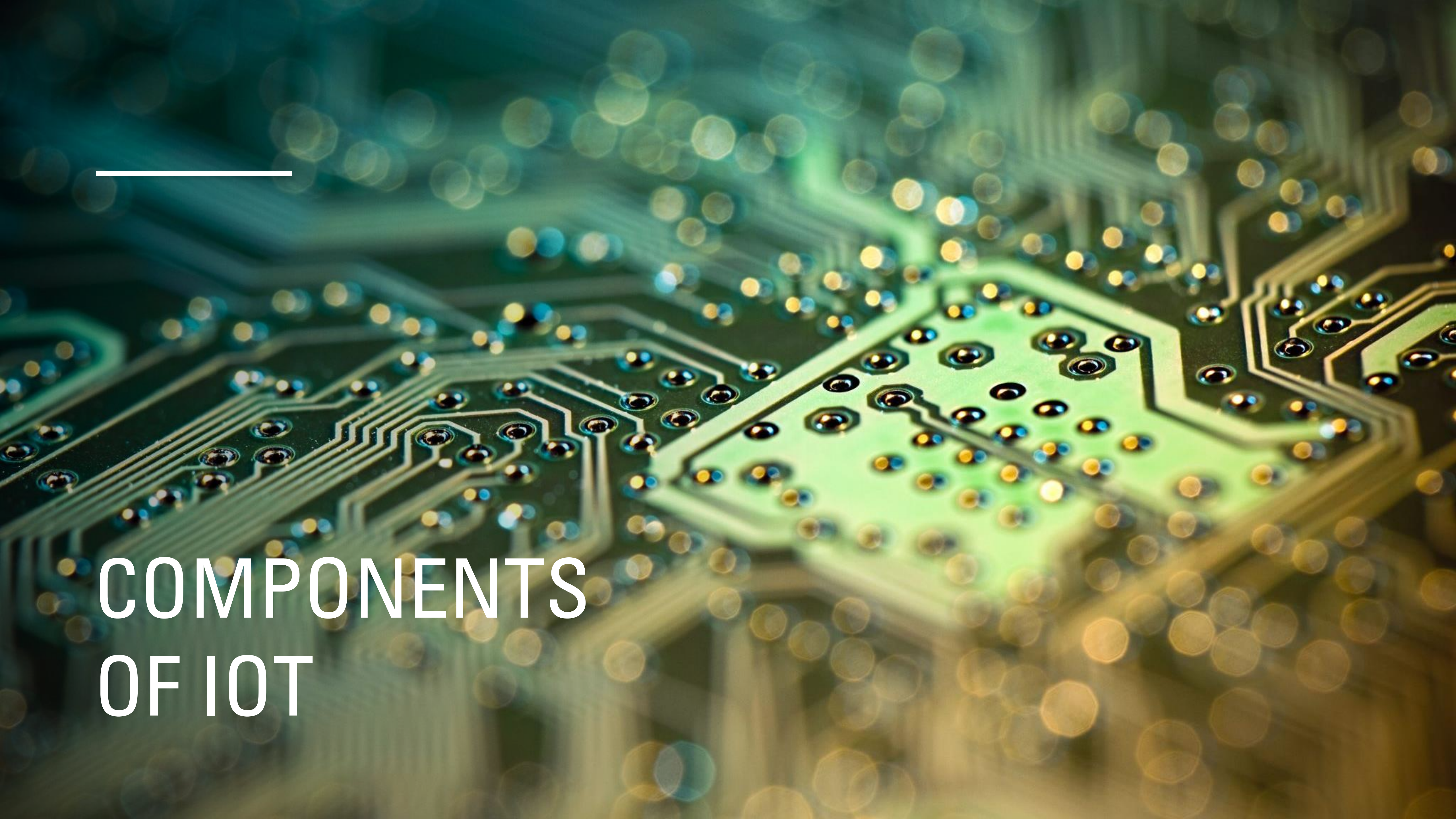
- Sensors Signals

- Industrial solutions

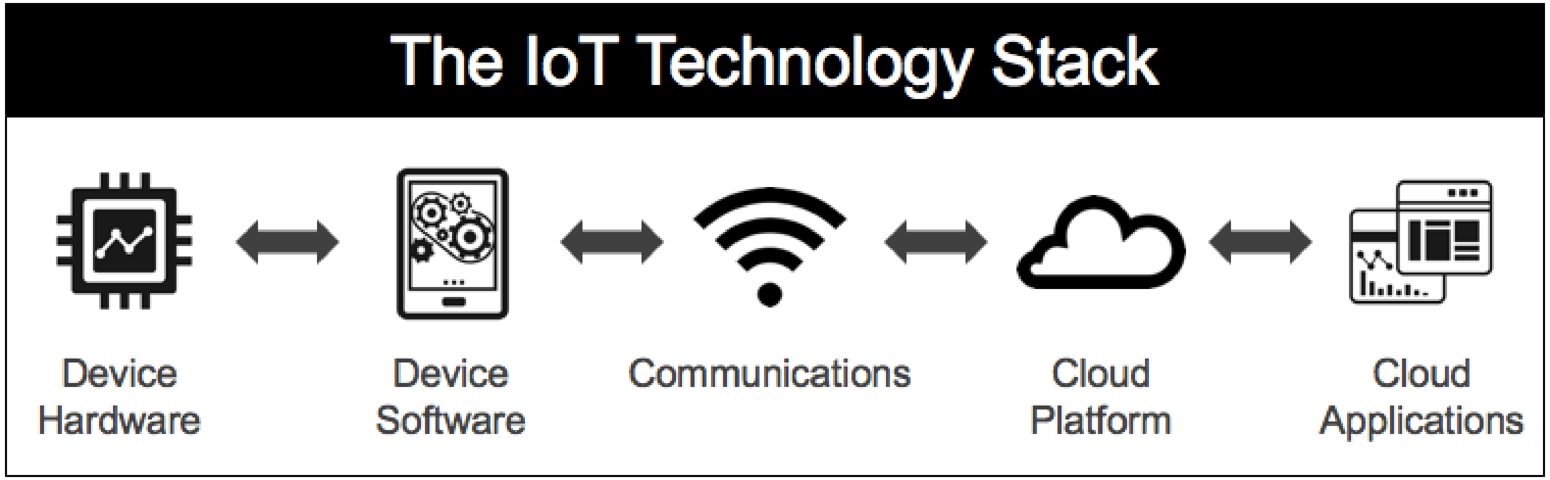
# WHAT IS IOT?

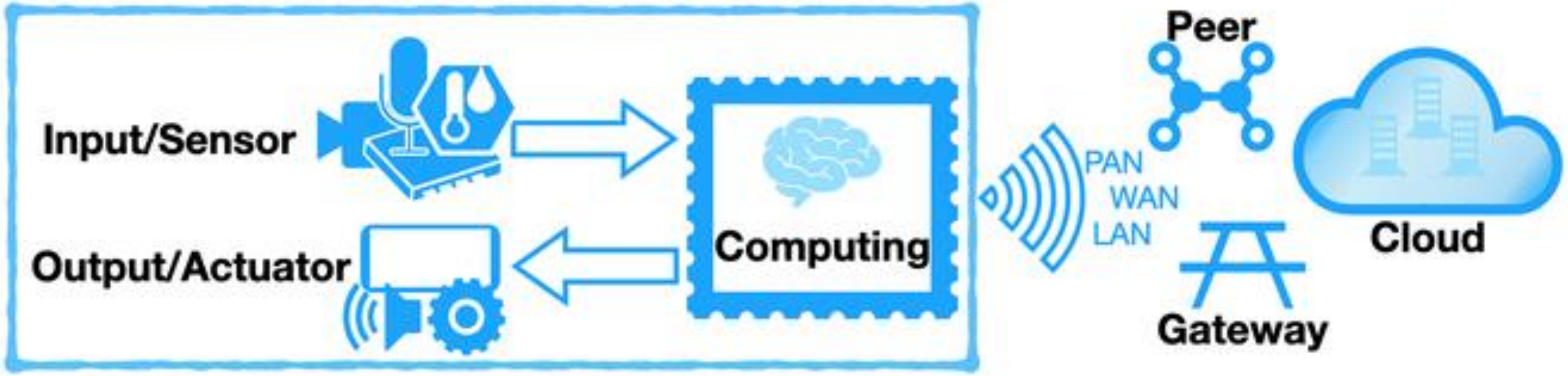
- Basically: Human independent communication



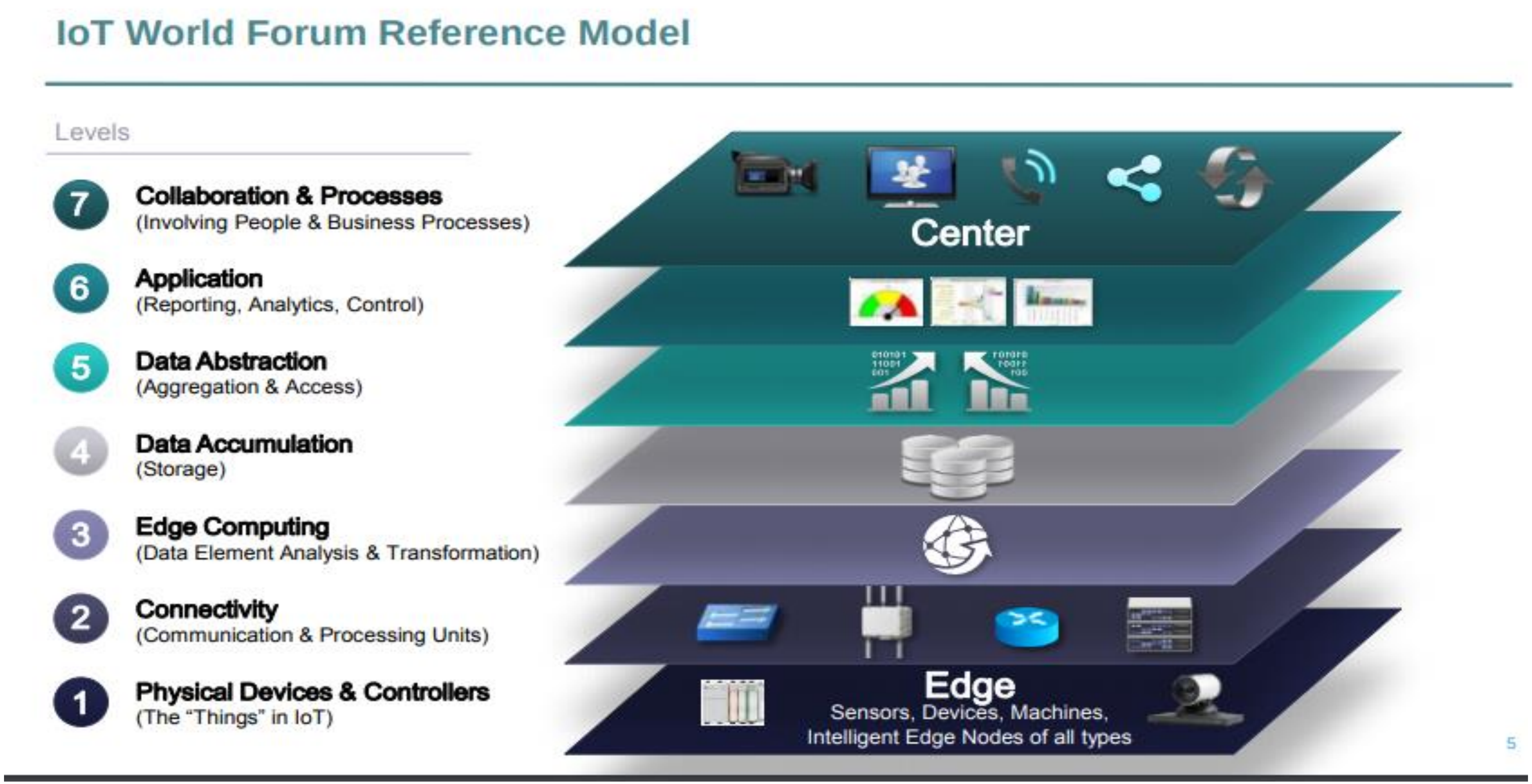


# COMPONENTS OF IOT

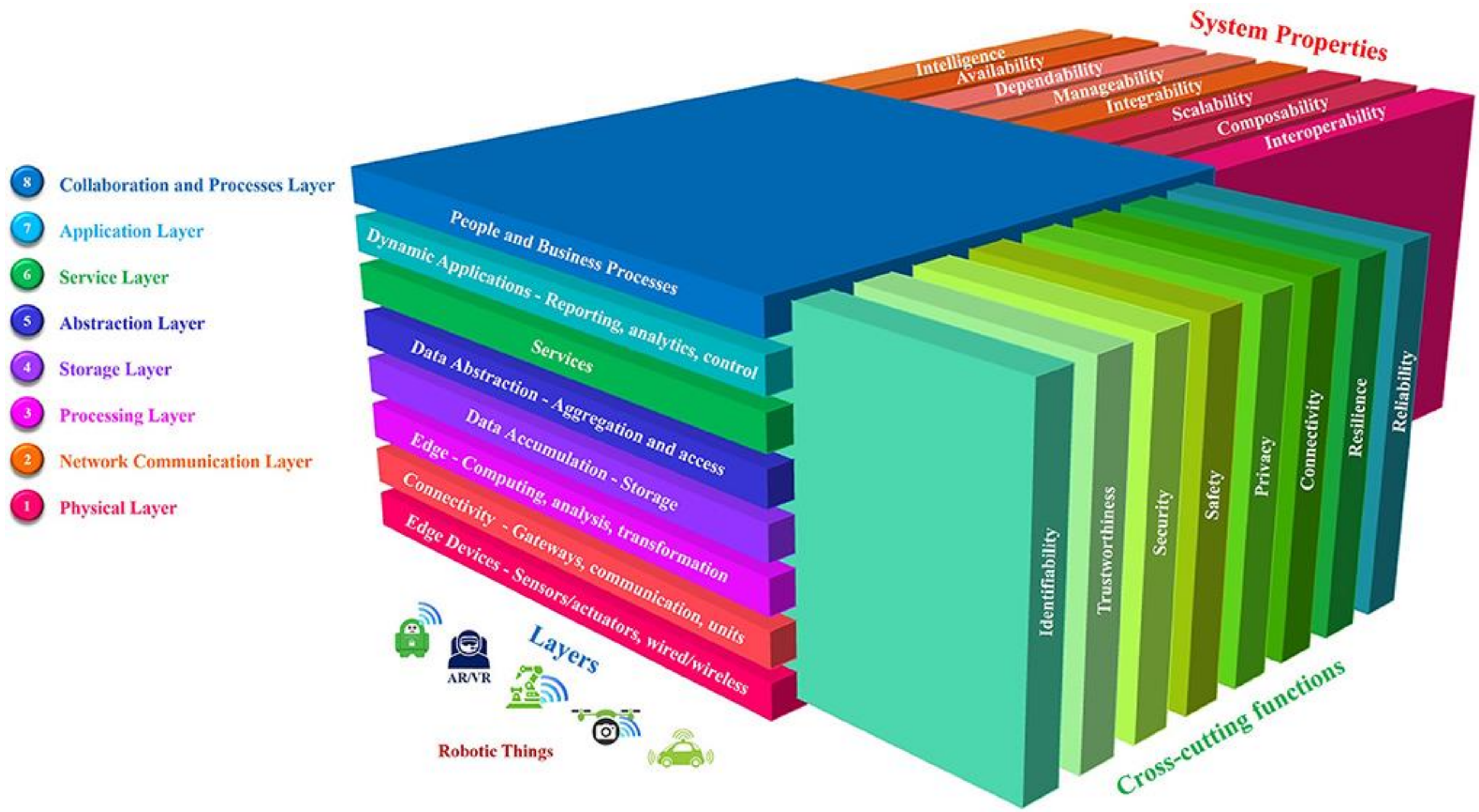

The IoT Technology Stack
Device Hardware
Device Software
Communications
Cloud Platform
Cloud Applications


Input/Sensor
Output/Actuator
Computing
PAN
WAN
LAN
Peer
Gateway
Cloud

# IOT ARCHITECTURE



# IOT AND IIOT



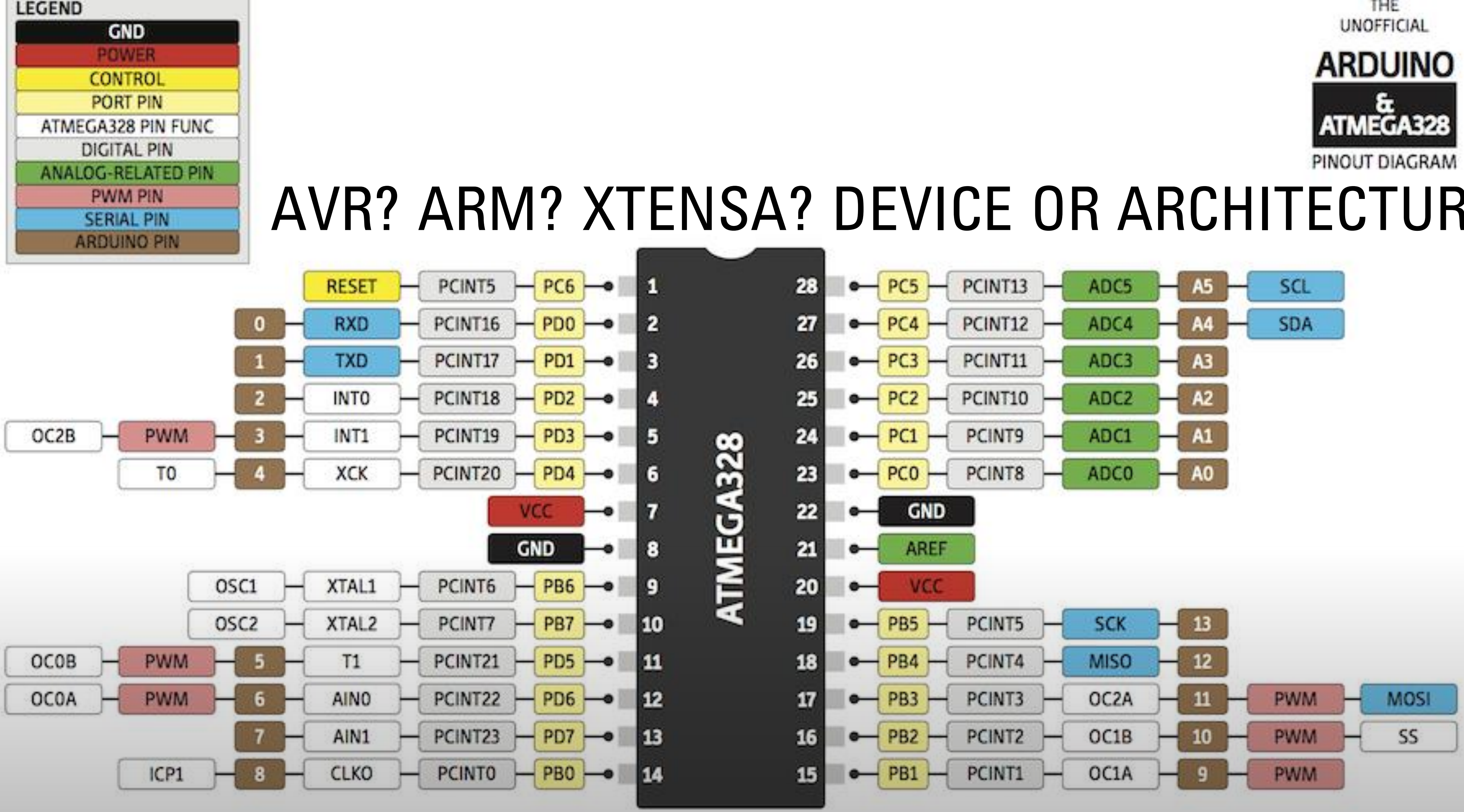
System Properties
Intelligence
Availability
Dependability
Manageability
Integrability
Scalability
Composability
Interoperability
8 Collaboration and Processes Layer
7 Application Layer
6 Service Layer
5 Abstraction Layer
4 Storage Layer
3 Processing Layer
2 Network Communication Layer
1 Physical Layer
People and Business Processes
Dynamic Applications - Reporting, analytics, control
Services
Data Abstraction - Aggregation and access
Data Accumulation - Storage
Edge - Computing, analysis, transformation
Connectivity - Gateways, communication, units
Edge Devices - Sensors/actuators, wired/wireless
Identifiability
Trustworthiness
Security
Safety
Privacy
Connectivity
Resilience
Reliability
Cross-cutting functions
Layers
AR/VR
Robotic Things

| | | |
|---|---|---|
| Commercial or Consumer Convenience | **Area of Focus** | Monitoring and Managing Systems for High-stake Industries- Defense, Manufacturing, Health care & Others |
| Smart Devices | **Focus Development** | Sophisticated Machines |
| Sensitive Sensors, Advanced Controls and Analytics | **Degree of Application** | Simple Application with Low-risk Impacts |
| Utility-centric | **Security and Risk Measures** | Advanced and Robust |
| Functionally Independent | **Interoperability** | Integration with Co-existing Legacy Operations Systems |

| | | |
|---|---|---|
| Low Scale Networks | **Scalability** | Large Scale Networks |
| Critically Monitored | **Precision and Accuracy** | Synchronized to Milliseconds |
| Easy Off-site Programming | **Programmability** | Remote on-site Reprogramming Required to Support New Processes |
| Convenience | **Output** | Economic Growth |
| Not Required | **Resilience** | Must be Automated to Support Fault Tolerance |

# DEVICES AND ARCHITECTURES

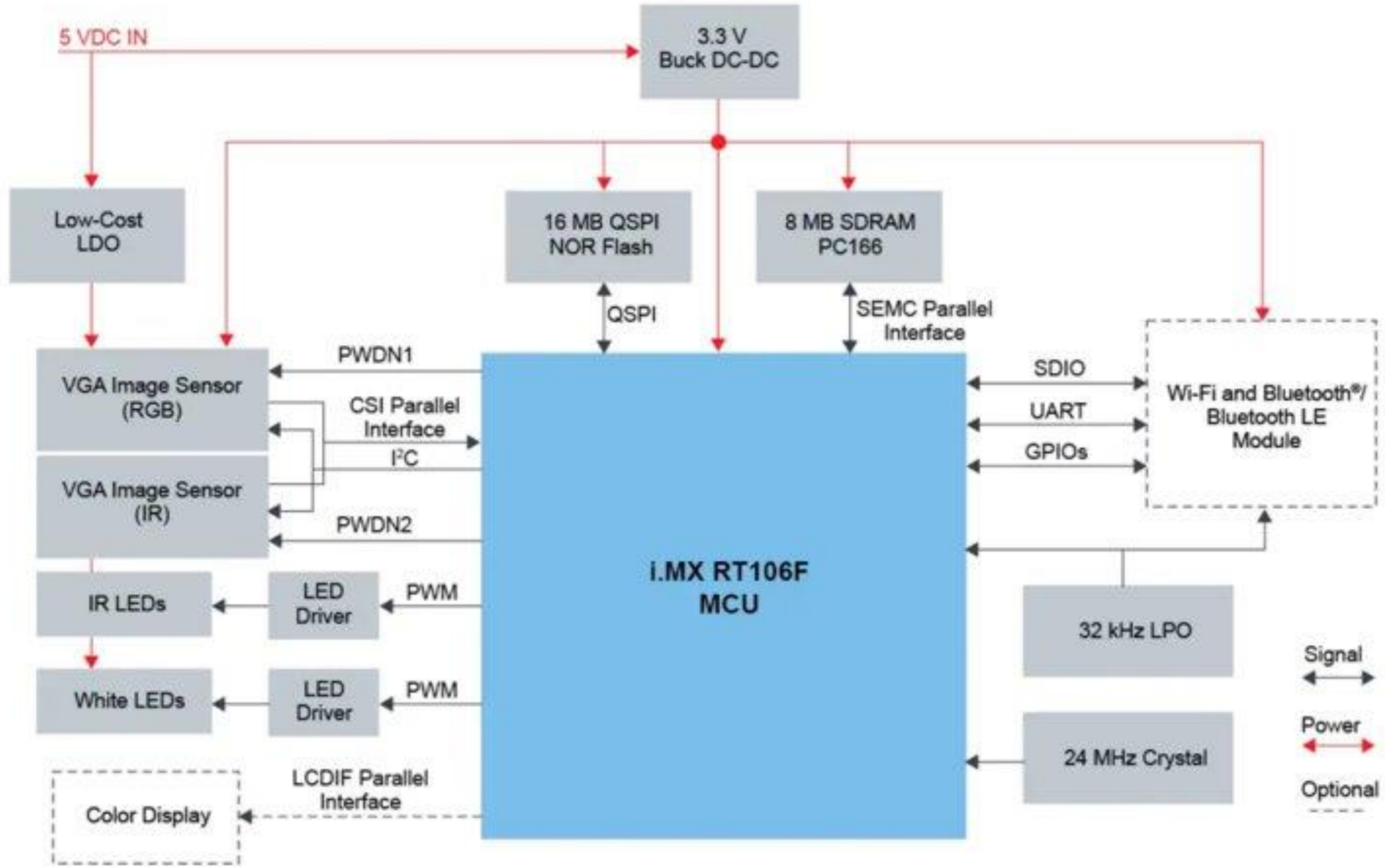
# AVR? ARM? XTENSA? DEVICE OR ARCHITECTURE?

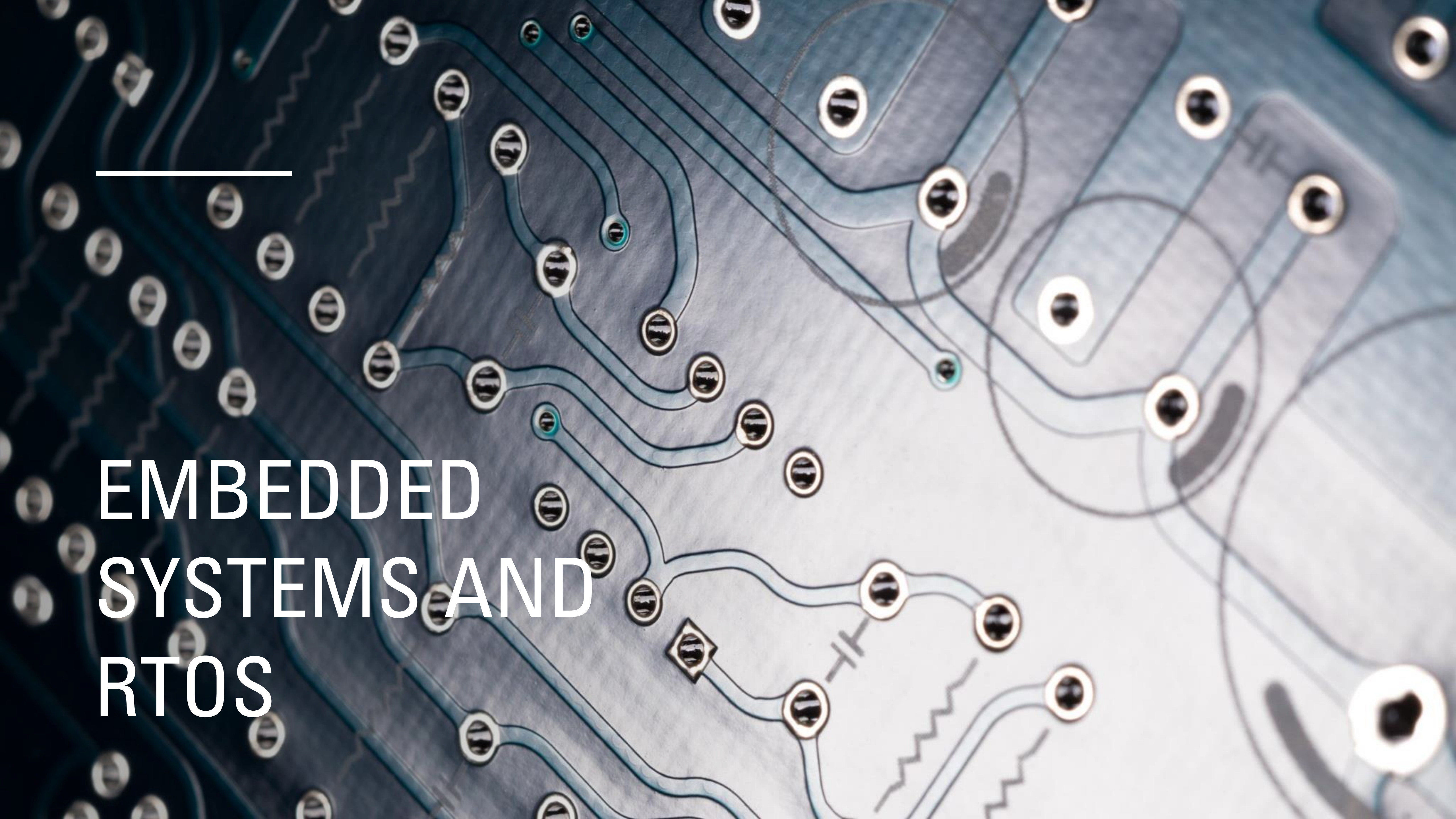


| Specifications | ESP8266 | ESP32 |
|---|---|---|
| MCU | Xtensa® Single-Core 32-bit L106 | Xtensa® Dual-Core 32-bit LX6 600 DMIPS |
| 802.11 b/g/n Wi-Fi | Yes, HT20 | Yes, HT40 |
| Bluetooth | None | Bluetooth 4.2 and below |
| Typical Frequency | 80 MHz | 160 MHz |
| SRAM | 160 kBytes | 512 kBytes |
| Flash | SPI Flash , up to 16 MBytes | SPI Flash , up to 16 MBytes |
| GPIO | 17 | 36 |
| Hardware / Software PWM | None / 8 Channels | 1 / 16 Channels |
| SPI / I2C / I2S / UART | 2/1/2/2 | 4/2/2/2 |
| ADC | 10-bit | 12-bit |
| CAN | None | 1 |
| Ethernet MAC Interface | None | 1 |
| Touch Sensor | None | Yes |
| Temperature Sensor | None | Yes |
| Working Temperature | - 40℃ ~ 125℃ | - 40℃ ~ 125℃ |

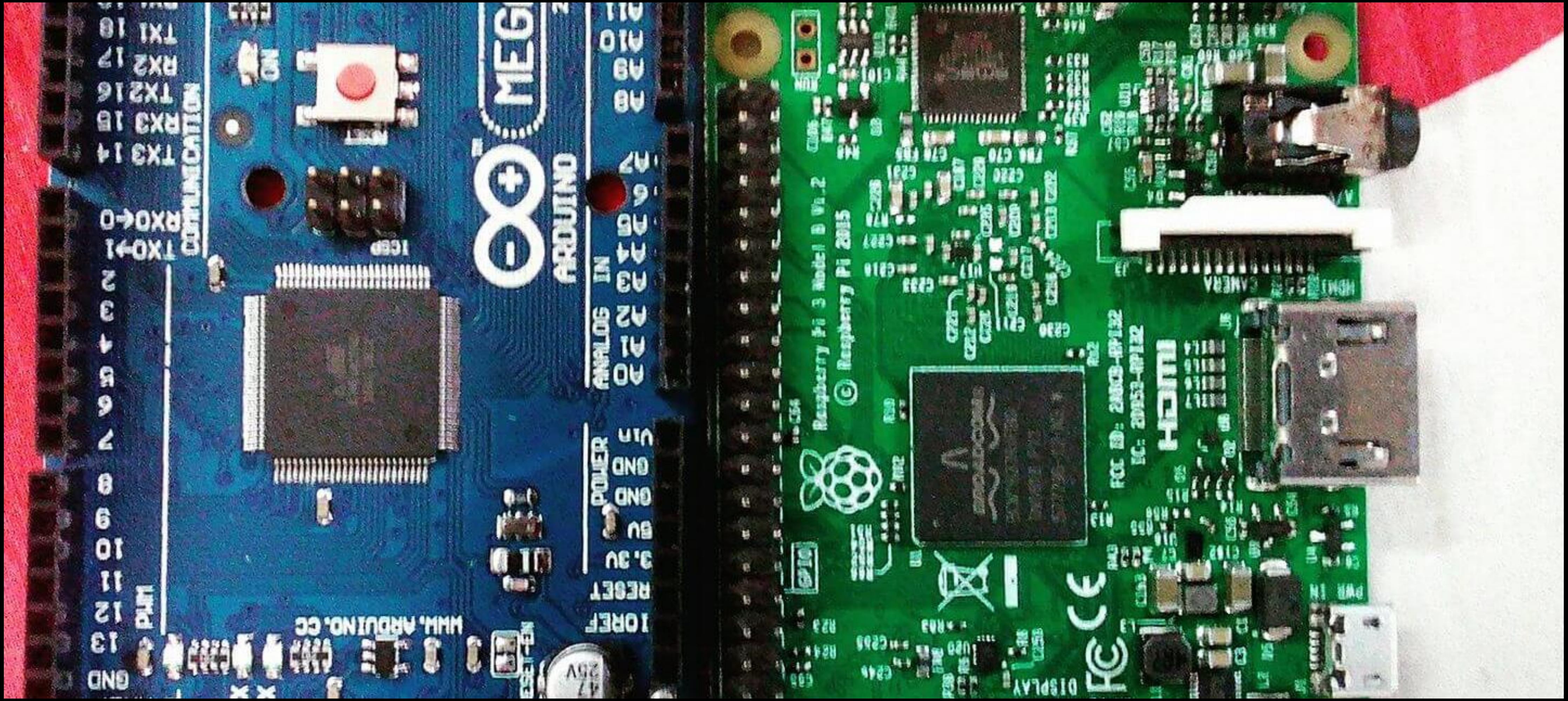
# PARAMETERS OF CHOOSING THE DEVICE

| Architecture | Memory/Addressing | Power Consumption | Miscellaneous |
|---|---|---|---|
| ARM | Flash | Modes available | Networking options |
| x86 | EEPROM | Wake time | Peripherals |
| AVR | 8 bit | | Signal Processing |
| RISC V | 16 bit | | Magnitudes |
| XTENSA | 32 bit | | Costs |


5 VDC IN
3.3 V
Buck DC-DC
Low-Cost
LDO
16 MB QSPI
NOR Flash
8 MB SDRAM
PC166
QSPI
SEMC Parallel
Interface
PWDN1
VGA Image Sensor
(RGB)
CSI Parallel
Interface
I²C
VGA Image Sensor
(IR)
PWDN2
SDIO
UART
GPIOs
Wi-Fi and Bluetooth®/
Bluetooth LE
Module
i.MX RT106F
MCU
IR LEDs
LED
Driver
PWM
White LEDs
LED
Driver
PWM
32 kHz LPO
24 MHz Crystal
Signal
Power
Optional
Color Display
LCDIF Parallel
Interface



# EMBEDDED SYSTEMS AND RTOS


ARDUINO
WWW.ARDUINO.CC
CAMERA
HDMI
DISPLAY

# RTOS

*Real-Time Operating System*

- Deterministic: no random execution pattern
- Predictable Response Times
- Time Bound
- Preemptive Kernel

**Examples:**

Contiki source code, FreeRTOS™, Zephyr™ Project

**Use Case:**

Embedded Computing

# GPOS

*General-Purpose Operating System*

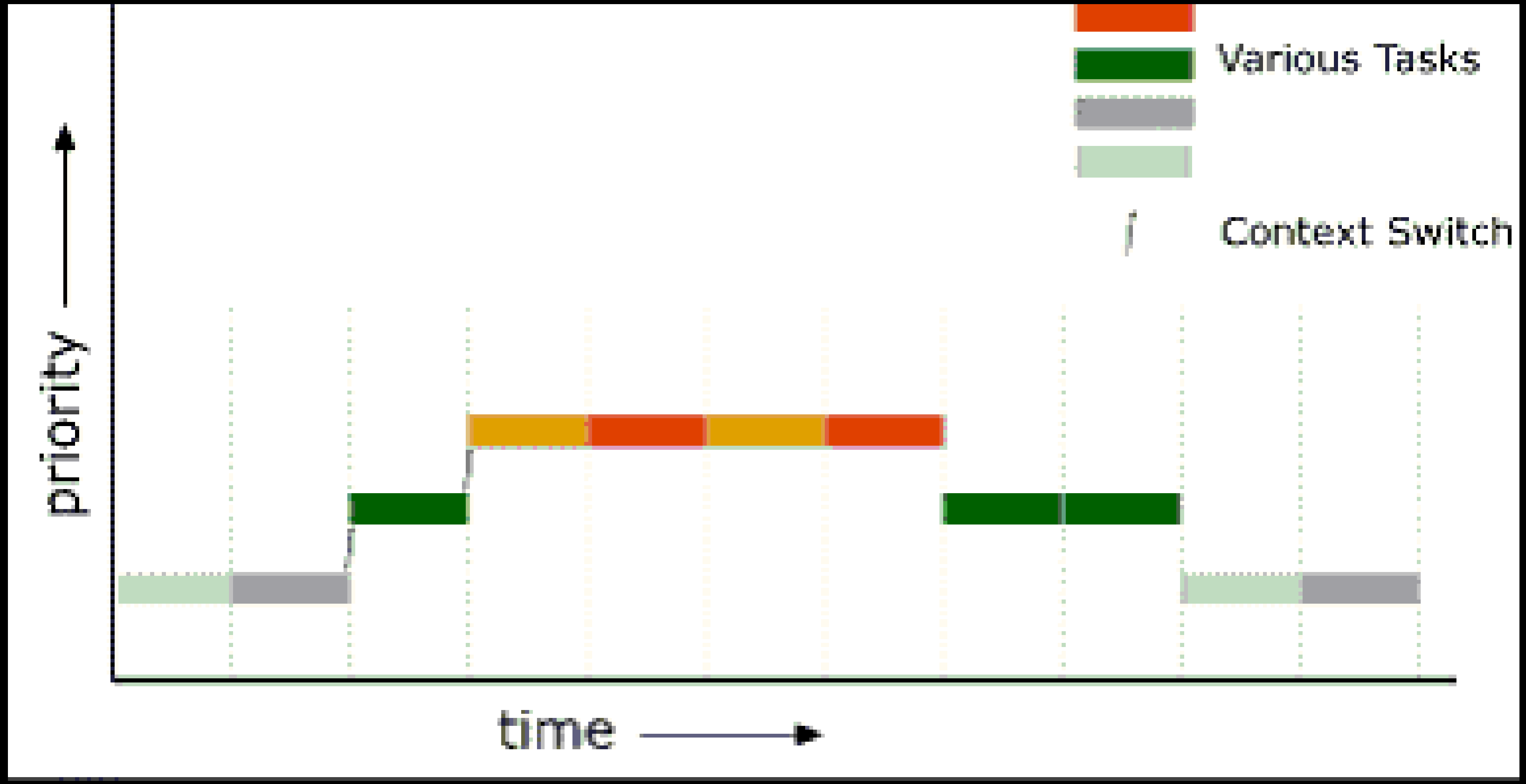
- Dynamic memory mapping
- Random Execution Pattern
- Response Times not Guaranteed

**Examples:**

Microsoft® Windows® operating system, Apple® macOS® operating system, Red Hat® Enterprise Linux® operating system

**Use Case:**

Desktop, Laptop, Tablet computers



Various Tasks
Context Switch
priority
time

# HOW DO YOU CHOOSE AN RTOS

**Schedulers** – Operating system schedulers are responsible for deciding which task should be executed at which time. They can suspend any task and resume it later, depending on availability of resources and priority. An embedded RTOS should have a scheduler built into it, which will enable it to perform real-time execution of the tasks.

**Software Timers** – A software timer allows a function to be executed within a set time-window. Manually creating software timers within an RTOS can become very complex and error-prone . Hence, it is always advisable to select an RTOS with in-built software timer functionality.

**Queuing Mechanism** – Queues are software mechanisms for exchanging information between tasks. In other words, queues facilitate smooth inter-task communication, or between tasks and interrupts. With queues built into an RTOS, different tasks will be able to exchange messages with each other

# HOW DO YOU CHOOSE AN RTOS

**Execution Trace Tools** – To fully understand the runtime behavior of an RTOS based system, you need the ability to observe its real-time behavior at the RTOS level. This can be achieved with the help of a tracing tool. If an RTOS has execution trace tools built into it, development teams can track the execution of tasks and detect discrepancies if any.

**Low Memory Footprint** – Since an RTOS needs to be integrated into a MCU platform, it should have a very low memory footprint. That is to say that its total size should not be in excess of 10 percent of the total size of the MCU.In addition to having the above-mentioned features, an embedded RTOS should be compatible with the target MCU architecture.


Q & A

# NEXT STEPS