

## DELIVERABLE 2.2

### Teaching Materials - Internet of Things Presentation Slides

Written by	Responsibility
Marios Raspopoulos (UCLAN)	WP2 Leader
Nearchos Paspallis (UCLAN)	Member
Stelios Ioannou (UCLAN)	Member
Josephina Antoniou (UCLAN)	Member
Eliana Stavrou (UCLAN)	Member
Fabrizio Granelli (UNINT)	Member
Claudio Sacchi (UNINT)	Member
Omar R Daoud (PU)	Member
Mohammed Bani Younis (PU)	Member
Saleh Saraireh (PU)	Member
Rasha Gh. Freehat (PU)	Member
Jonathan Rodriguez (IT)	WP5 Leader
Georgios Mantas (IT)	Member
Maria Papaioannou (IT)	Member
Claudia Barbosa (IT)	Member
Felipe Gil-Castiñeira (UVIGO)	WP4 Leader
Cristina López-Bravo (UVIGO)	Member
René Lastra Cid (UVIGO)	Member
Saud Althunibat (AHU)	Project Coordinator
Moath Safasfeh (AHU)	Member
Samiha Falahat (AHU)	Member
<b>Edited by</b>	
Marios Raspopoulos (UCLAN)	WP2 Leader
<b>Approved by</b>	
Saud Althunibat (AHU)	Project Coordinator

This publication was produced with the financial support of the European Union. Its contents are the sole responsibility of the partners of IREEDER project and do not necessarily reflect the views of the European Union



# This week's topics...

- What Is the Internet of Things?
  - History of IoT
- Overview IoT Enabling Technologies
  - Sensory, data storage, connectivity, etc.
- IoT Vertical Applications
  - Industrial
  - Commercial Medical/Healthcare
  - Automotive
  - Energy/Utilities
  - Financial
  - Open Source applications.
- Identification of key research directions and connections

# Introduction

- The IoT technology or Internet of Things is a network of connected things which can communicate data without human involvement, such as smart appliances, electronics, and machines.
- The Internet of Things is not the "hype" or a "buzzword" anymore, it now has the power to change our world.
- Regardless of the actual number of connected devices (Gartner estimated 50B devices connected by 2020), there is a concrete indication that IoT will play an important role in our everyday life.
- This course describes the basics and applications of the IoT technology.

# Section Outline

- What is the Internet of Things?
- History of IoT



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

## Section 1

What is the Internet of Things?

---

# What is the Internet of Things?

- The Internet of things (IoT) is a system of interrelated computing devices, mechanical and digital machines provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. ("Internet of Things Global Standards Initiative". ITU. Retrieved 26 June 2015.)

# History

- 1982: a modified Coca-Cola vending machine at Carnegie Mellon University becomes the first Internet-connected appliance, able to report its inventory and whether newly loaded drinks were cold or not.



Source: CMU

# History

- 1991: Mark Weiser's paper on ubiquitous computing, "The Computer of the 21st Century", as well as academic venues such as UbiComp and PerCom produce the contemporary vision of the IoT
- 1993-1997: several companies proposed solutions like Microsoft's at Work or Novell's NEST.
- 1999: the field gains momentum when Bill Joy envisions device-to-device communication as a part of his "Six Webs" framework, presented at the World Economic Forum at Davos.

# History

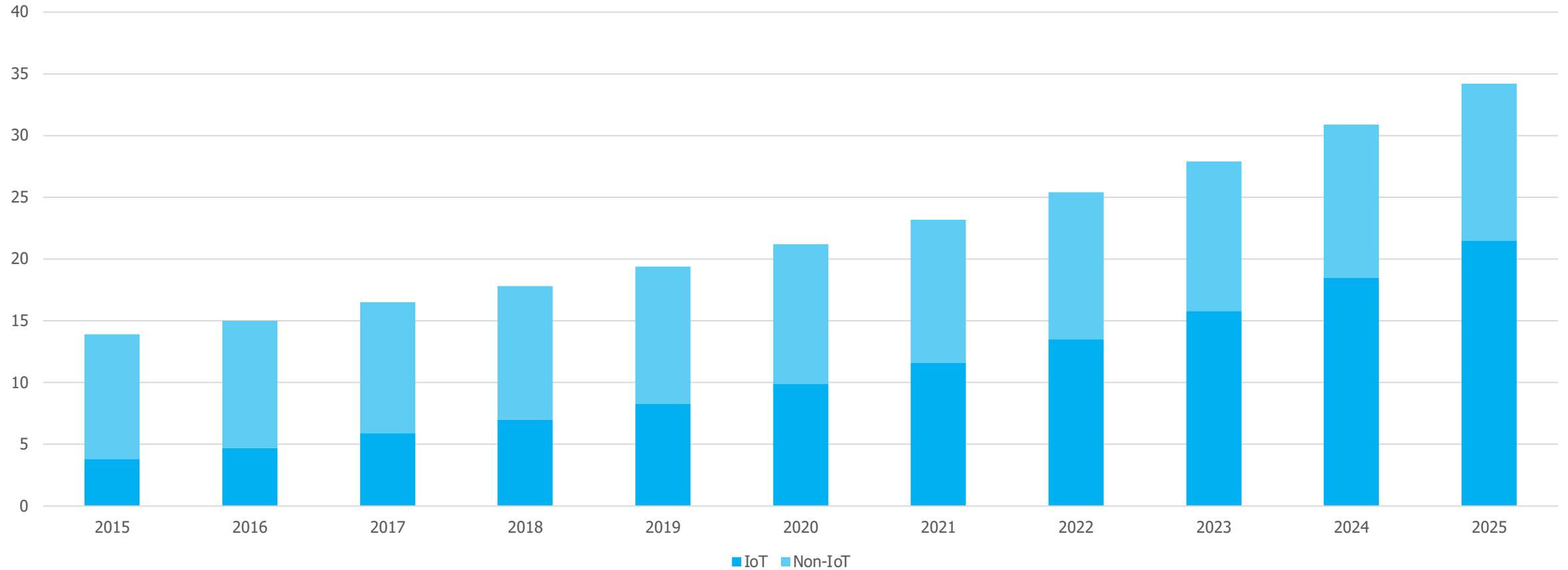
- 1999: the term "Internet of things" was likely coined by Kevin Ashton of Procter & Gamble, later MIT's Auto-ID Center, though he prefers the phrase "Internet for things".
- Radio-frequency identification (RFID) is considered essential to the Internet of things, which would allow computers to manage all individual things.
- Defining the Internet of things as "simply the point in time when more 'things or objects' were connected to the Internet than people", Cisco Systems estimated that the IoT was "born" between 2008 and 2009, with the things/people ratio growing from 0.08 in 2003 to 1.84 in 2010.

# IoT Facts

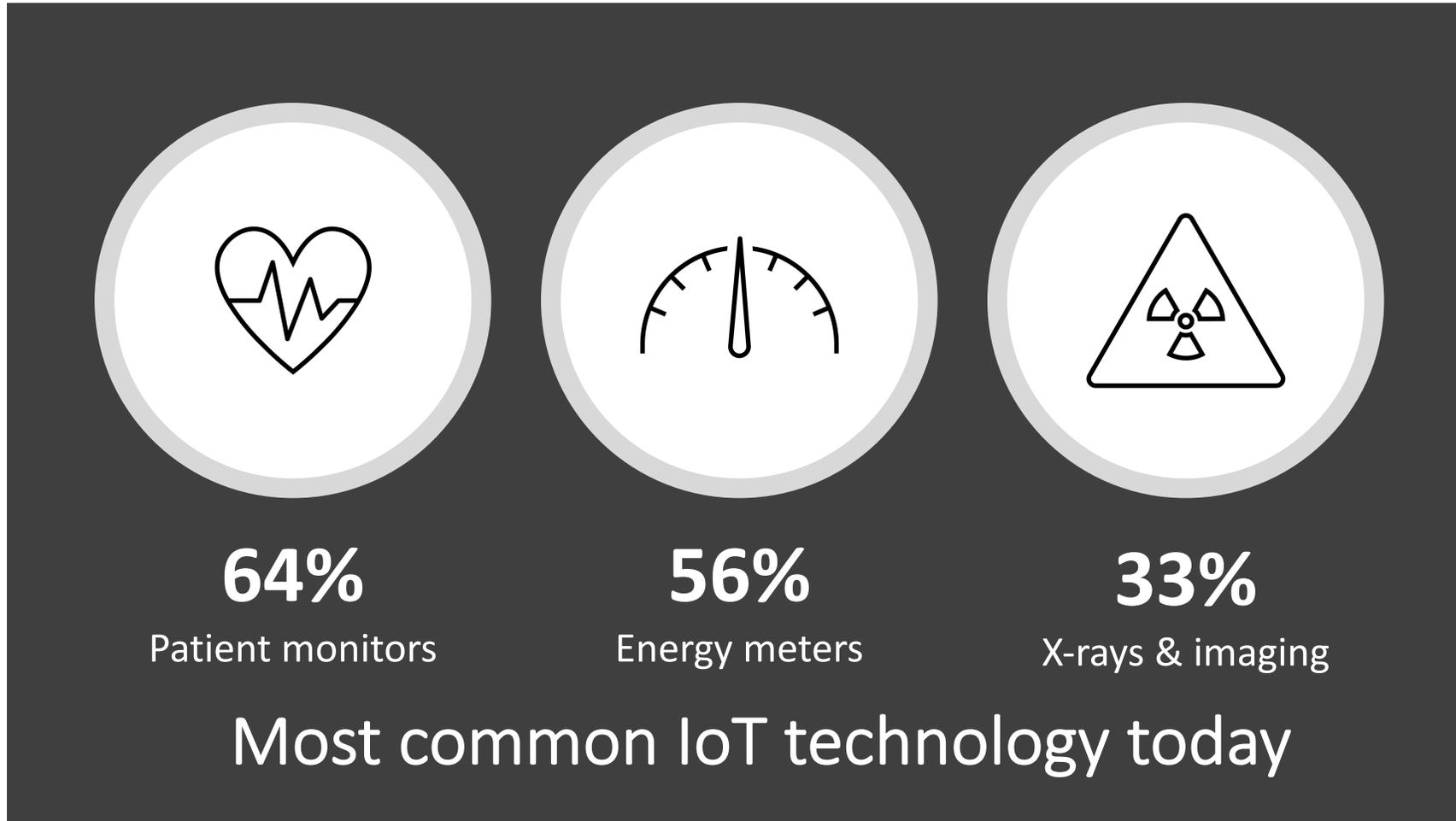
- 7 billion devices were connected to the internet as of 2018, with the number of connected devices expected to approach 10 billion by the end of 2020. (Source: IOT Analytics)
- Gartner predicts a larger amount of “connected things” by 2020. According to Gartner, there will be over 14 billion connected devices by the end of 2019, and over 25 billion by the end of 2021. (Source: Gartner)
- A large majority of IoT is made up of smartphones. According to Newzoo, the total number of smartphone users topped 3 billion in 2018. (Source: Newzoo)
- Most IoT devices are those we’re using at home, or at work. As of 2018, over half of all IoT devices were connected to Wireless Personal Area Networks, such as Bluetooth, Zigbee, and Z-Wave. (Source: IOT Analytics)
- The global IoT market was worth over \$150 billion in 2018 and is expected to exceed \$1.5 trillion by 2025. (Source: IOT Analytics)

# IoT Penetration

Total Number of Active Device Connections Worldwide (in Bn)

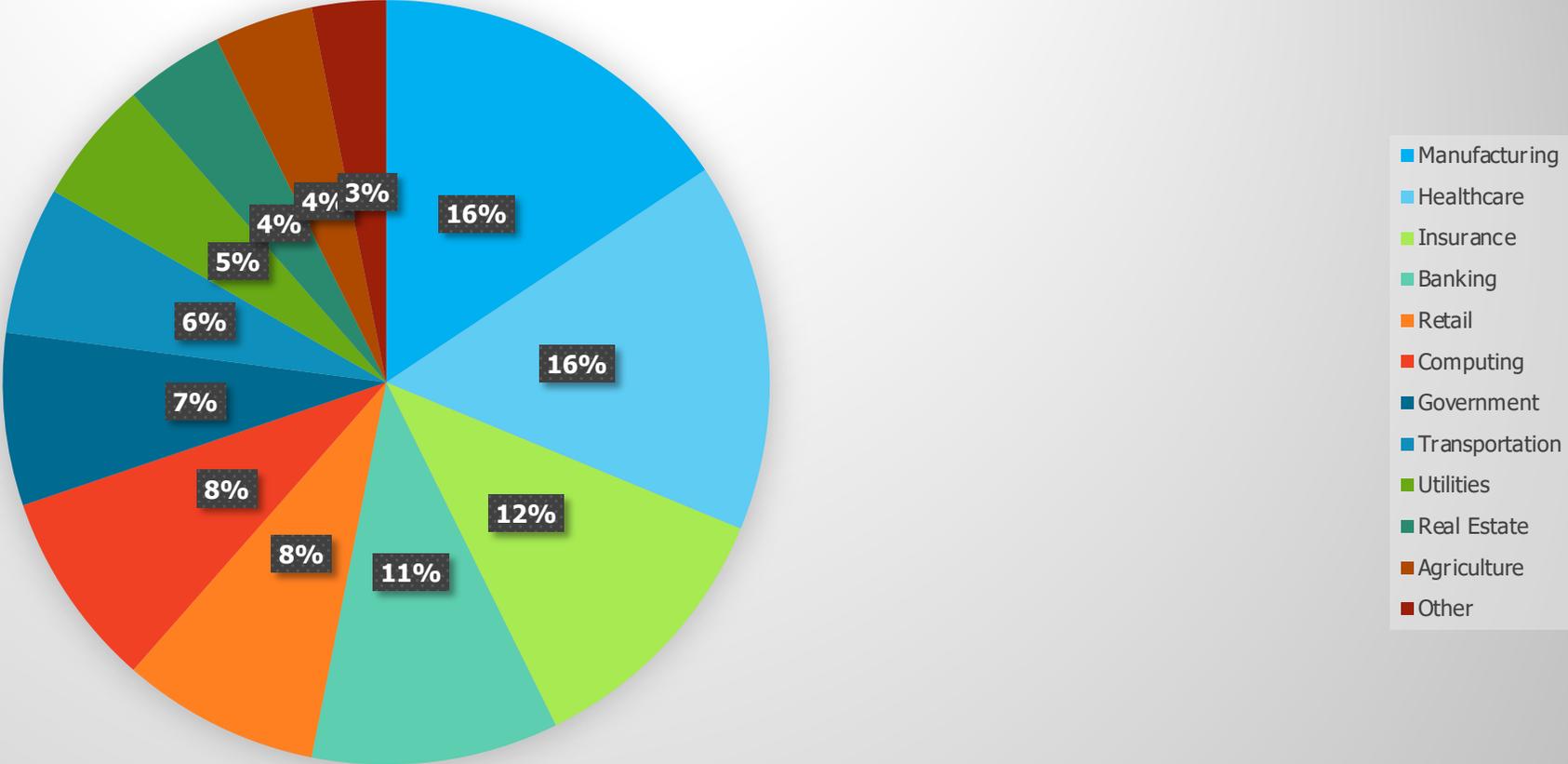


# What happens today?



# Applications

IoT Value Add by 2020 (\$1.9 Trillion)





# Enabling Technologies

## Foreword

- There are many technologies that enable the IoT.
- Crucial to the field is the network used to communicate between devices of an IoT installation, a role that several wireless or wired technologies may fulfil.
- The next slides summarize the major functionalities and technologies required by IoT.

# Enabling Technologies

## Addressability and Application Layer

### ○ Addressability

- The original idea of the Auto-ID Center is based on RFID-tags and distinct identification through the Electronic Product Code.
- This has evolved into objects having an IP address or URI (Universal Resource Identifier).

### ○ Application Layer

- An application layer protocol and supporting framework for implementing IoT applications is required.
- E.g. ADRC.

# Enabling Technologies

## Wireless Technologies

### ○ Short-range wireless

- Bluetooth mesh networking – Specification providing a mesh networking variant to Bluetooth low energy (BLE) with increased number of nodes and standardized application layer (Models).
- Light-Fidelity (Li-Fi) – Wireless communication technology similar to the Wi-Fi standard, but using visible light communication for increased bandwidth.
- Near-field communication (NFC) – Communication protocols enabling two electronic devices to communicate within a 4 cm range.
- Radio-frequency identification (RFID) – Technology using electromagnetic fields to read data stored in tags embedded in other items.
- Wi-Fi – Technology for local area networking based on the IEEE 802.11 standard, where devices may communicate through a shared access point or directly between individual devices.
- ZigBee – Communication protocols for personal area networking based on the IEEE 802.15.4 standard, providing low power consumption, low data rate, low cost, and high throughput.
- Z-Wave – Wireless communications protocol used primarily for home automation and security applications

# Enabling Technologies

## Wireless Technologies

### ○ Medium-range wireless

- LTE-Advanced – High-speed communication specification for mobile networks. Provides enhancements to the LTE standard with extended coverage, higher throughput, and lower latency.
- 5G - 5G wireless networks can be used to achieve the high communication requirements of the IoT and connect a large number of IoT devices, even when they are on the move.

### ○ Long-range wireless

- Low-power wide-area networking (LPWAN) – Wireless networks designed to allow long-range communication at a low data rate, reducing power and cost for transmission. Available LPWAN technologies and protocols: LoRaWan, Sigfox, NB-IoT, Weightless, RPMA.
- Very small aperture terminal (VSAT) – Satellite communication technology using small dish antennas for narrowband and broadband data.

# Enabling Technologies

## Wired Technologies

- Ethernet – General purpose networking standard using twisted pair and fiber optic links in conjunction with hubs or switches.
- Power-line communication (PLC) – Communication technology using electrical wiring to carry power and data. Specifications such as HomePlug or G.hn utilize PLC for networking IoT devices.

# Enabling Technologies

## Standards and Standards Organizations

Short name	Long name	Standards under development
<a href="#">Auto-ID Labs</a>	Auto Identification Center	Networked <a href="#">RFID</a> (radiofrequency identification) and emerging <a href="#">sensing</a> technologies
<a href="#">EPCglobal</a>	Electronic Product code Technology	Standards for adoption of <a href="#">EPC</a> (Electronic Product Code) technology
<a href="#">FDA</a>	U.S. Food and Drug Administration	<a href="#">UDI</a> (Unique Device Identification) system for distinct identifiers for <a href="#">medical devices</a>
<a href="#">GS1</a>	Global Standards One	Standards for <a href="#">UIDs</a> ("unique" identifiers) and RFID of <a href="#">fast-moving consumer goods</a> (consumer packaged goods), health care supplies, and other thingsThe GS1 digital link standard, <a href="#">U471</a> first released in August 2018, allows the use QR Codes, GS1 Datamatrix, RFID and NFC to enable various types of business-to-business, as well as business-to-consumers interactions.
<a href="#">IEEE</a>	Institute of Electrical and Electronics Engineers	Underlying communication technology standards such as <a href="#">IEEE 802.15.4</a> , IEEE P1451-99 (IoT Harmonization), and IEEE P1931.1 (ROOF Computing).
<a href="#">IETF</a>	Internet Engineering Task Force	Standards that comprise <a href="#">TCP/IP</a> (the Internet protocol suite)
MTConnect Institute	—	<a href="#">MTConnect</a> is a manufacturing industry standard for data exchange with <a href="#">machine tools</a> and related industrial equipment. It is important to the IIoT subset of the IoT.
<a href="#">O-DF</a>	Open Data Format	O-DF is a standard published by the Internet of Things Work Group of The Open Group in 2014, which specifies a generic information model structure that is meant to be applicable for describing any "Thing", as well as for publishing, updating and querying information when used together with <a href="#">O-MI</a> (Open Messaging Interface).
<a href="#">O-MI</a>	Open Messaging Interface	O-MI is a standard published by the Internet of Things Work Group of The Open Group in 2014, which specifies a limited set of key operations needed in IoT systems, notably different kinds of subscription mechanisms based on the <a href="#">Observer pattern</a> .
<a href="#">OCF</a>	Open Connectivity Foundation	Standards for simple devices using <a href="#">CoAP</a> (Constrained Application Protocol)
<a href="#">OMA</a>	Open Mobile Alliance	<a href="#">OMA DM</a> and <a href="#">OMA LWM2M</a> for IoT device management, as well as GotAPI, which provides a secure framework for IoT applications
<a href="#">XSF</a>	XMPP Standards Foundation	Protocol extensions of <a href="#">XMPP</a> (Extensible Messaging and Presence Protocol), the open standard of <a href="#">instant messaging</a>

## Section Outline

- IoT Vertical Applications
  - *Industrial*
  - *Commercial Medical/Healthcare*
  - *Automotive*
  - *Energy/Utilities*
  - *Financial*
  - *Open Source applications.*



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

## Section 3

### IoT Vertical Applications

---

# IoT Applications

## Consumer Applications

- A growing portion of IoT devices are created for consumer use, including connected vehicles, home automation, wearable technology, connected health, and appliances with remote monitoring capabilities.
- Smart Home
  - IoT devices are a part of the larger concept of home automation, which can include lighting, heating and air conditioning, media and security systems. Long-term benefits could include energy savings by automatically ensuring lights and electronics are turned off.
- Elder care
  - One key application of a smart home is to provide assistance for those with disabilities and elderly individuals. These home systems use assistive technology to accommodate an owner's specific disabilities.

# IoT Applications

## Organizational Applications

- Medical and healthcare
  - The Internet of Medical Things (IoMT) is an application of the IoT for medical and health related purposes, data collection and analysis for research, and monitoring
- Transportation
  - The IoT can assist in the integration of communications, control, and information processing across various transportation systems.
- V2X communications
  - In vehicular communication systems, vehicle-to-everything communication (V2X), consists of three main components: vehicle to vehicle communication (V2V), vehicle to infrastructure communication (V2I) and vehicle to pedestrian communications (V2P). V2X is the first step to autonomous driving and connected road infrastructure
- Building and home automation
  - IoT devices can be used to monitor and control the mechanical, electrical and electronic systems used in various types of buildings (e.g., public and private, industrial, institutions, or residential)

# IoT Applications

## Industrial Applications

- Also known as IIoT, industrial IoT devices acquire and analyze data from connected equipment, operational technology (OT), locations and people. Combined with operational technology (OT) monitoring devices, IIoT helps regulate and monitor industrial systems.
- Manufacturing
  - The IoT can realize the seamless integration of various manufacturing devices equipped with sensing, identification, processing, communication, actuation, and networking capabilities. Based on such a highly integrated smart cyber-physical space, it opens the door to create whole new business and market opportunities for manufacturing.
- Agriculture
  - There are numerous IoT applications in farming, such as collecting data on temperature, rainfall, humidity, wind speed, pest infestation, and soil content. This data can be used to automate farming techniques, take informed decisions to improve quality and quantity, minimize risk and waste, and reduce effort required to manage crops. For example, farmers can now monitor soil temperature and moisture from afar, and even apply IoT-acquired data to precision fertilization programs

# IoT Applications

## Infrastructure Applications

- Monitoring and controlling operations of sustainable urban and rural infrastructures like bridges, railway tracks and on- and offshore wind-farms is a key application of the IoT. The IoT infrastructure can be used for monitoring any events or changes in structural conditions that can compromise safety and increase risk. Usage of IoT devices for monitoring and operating infrastructure is likely to improve incident management and emergency response coordination, and quality of service, up-times and reduce costs of operation in all infrastructure related areas.
- Metropolitan scale deployments / Smart City
  - There are several planned or ongoing large-scale deployments of the IoT, to enable better management of cities and systems. For example, Songdo, South Korea, the first of its kind fully equipped and wired smart city, is gradually being built, with approximately 70 percent of the business district completed as of June 2018. Much of the city is planned to be wired and automated, with little or no human intervention

# IoT Applications

## Infrastructure Applications

### ○ Energy management

- Significant numbers of energy-consuming devices (e.g. lamps, household appliances, motors, pumps, etc.) already integrate Internet connectivity, which can allow them to communicate with utilities not only to balance power generation but also helps optimize the energy consumption as a whole
- The smart grid is a utility-side IoT application; systems gather and act on energy and power-related information to improve the efficiency of the production and distribution of electricity

### ○ Environmental monitoring

- Environmental monitoring applications of the IoT typically use sensors to assist in environmental protection by monitoring air or water quality, atmospheric or soil conditions, and can even include areas like monitoring the movements of wildlife and their habitats

### ○ Living Lab

- Another example of integrating the IoT is Living Lab which integrates and combines research and innovation process, establishing within a public-private-people-partnership

# IoT Applications

## Military Applications

- The Internet of Military Things (IoMT) is the application of IoT technologies in the military domain for the purposes of reconnaissance, surveillance, and other combat-related objectives.
- It is heavily influenced by the future prospects of warfare in an urban environment and involves the use of sensors, munitions, vehicles, robots, human-wearable biometrics, and other smart technology that is relevant on the battlefield.
- Internet of Battlefield Things
  - The Internet of Battlefield Things (IoBT) is a project initiated and executed by the U.S. Army Research Laboratory (ARL) that focuses on the basic science related to IoT that enhance the capabilities of Army soldiers.
- Ocean of Things
  - The Ocean of Things project is a DARPA-led program designed to establish an Internet of Things across large ocean areas for the purposes of collecting, monitoring, and analyzing environmental and vessel activity data.

# Section Outline

- Trends and characteristics
  - *Intelligence*
  - *Architecture*
  - *Network architecture*
  - *Size considerations*
  - *Space considerations*



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

## Section 4

Identification of key research directions and connections

# Trends and Characteristics

## Foreword

- The IoT's major significant trend in recent years is the explosive growth of devices connected and controlled by the Internet.
- The wide range of applications for IoT technology mean that the specifics can be very different from one device to the next but there are basic characteristics shared by most.

# Trends and Characteristics

## Intelligence

- There is a shift in research to integrate the concepts of the IoT and autonomous control, with initial outcomes towards this direction considering objects as the driving force for autonomous IoT
- IoT intelligence can be offered at three levels: IoT devices, Edge/Fog nodes, and Cloud computing.
- The need for intelligent control and decision at each level depends on the time sensitiveness of the IoT application.
  - For example, an autonomous vehicle's camera needs to make real-time obstacle detection to avoid an accident. This fast decision making would not be possible through transferring data from the vehicle to cloud instances and return the predictions back to the vehicle. Instead, all the operation should be performed locally in the vehicle.

# Trends and Characteristics

## Architecture

- IoT system architecture mainly consists of three tiers:
  - Tier 1: Devices
  - Tier 2: the Edge Gateway
  - Tier 3: the Cloud.
- Devices include networked things, such as the sensors and actuators found in IoT equipment
- The Edge Gateway consists of sensor data aggregation systems called Edge Gateways that provide functionality, such as pre-processing of the data, securing connectivity to cloud, using systems such as WebSockets, the event hub, and, even in some cases, edge analytics or fog computing
- The final tier includes the cloud environment built for supporting IoT using a microservices architecture

# Trends and Characteristics

## Architecture

Cloud

- Cellular, WiFi, Mobile, DSL, Fibre

Gateway

- Thread, Bluetooth, Zigbee, NFC, etc.

Things

- Sensors, actuators, IoT enabled devices, mesh networks

# Trends and Characteristics

## Network Architecture

- The Internet of things requires huge scalability in the network space to handle the surge of devices.
  - IETF 6LoWPAN could be used to connect devices to IP networks.
  - With billions of devices being added to the Internet space, IPv6 will play a major role in handling the network layer scalability.
- IETF's Constrained Application Protocol, ZeroMQ, and MQTT could play a relevant role in providing lightweight data transport.
- Fog computing is a viable alternative to prevent large bursts of data flow through Internet

# Trends and Characteristics

## Size Considerations

- The Internet of things could include 50 to 100 trillion objects, and be able to follow the movement of those objects.
- Human beings in surveyed urban environments are each surrounded by 1000 to 5000 trackable objects.
- In 2015 there were already 83 million smart devices in people's homes, expected to grow to 193 million devices by 2020.
- Online capable devices grew 31% from 2016 to 2017 to reach 8.4 billion.

# Trends and Characteristics

## Space Considerations

- In the Internet of Things, the precise geographic location of a thing will be critical.
  - Some things in the Internet of Things will be sensors, and sensor location is usually important.
- The GeoWeb<sup>[1]</sup> and Digital Earth<sup>[2]</sup> are promising applications that become possible when things can become organized and connected by location.

[1] <http://geowebforum.com/>

[2] <http://www.digitalearth-isde.org/>

# Section Outline

- Challenges
  - *Security*
  - *Regulation*
  - *Compatibility*
  - *Bandwidth*
  - *Energy Consumption*
  - *Customers' expectations*



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

## Section 4

### IoT Challenges

---

# IoT Challenges

## Security

- Security is the biggest concern in adopting Internet of things technology, with concerns that rapid development is happening without appropriate consideration of the profound security challenges involved and the regulatory changes that might be necessary.
- Internet of Things devices also have access to new areas of data, and can often control physical devices, so that even by 2014 it was possible to say that many Internet-connected appliances could already "spy on people in their own homes" including televisions, kitchen appliances, cameras, and thermostats.
- The Internet of Things Security Foundation (IoTSF) was launched on 23 September 2015 with a mission to secure the Internet of things by promoting knowledge and best practice. Its founding board is made from technology providers and telecommunications companies. In addition, large IT companies are continually developing innovative solutions to ensure the security of IoT devices.
- In 2017, Mozilla launched Project Things, which allows to route IoT devices through a safe Web of Things gateway.

# IoT Challenges

## Regulation

- Another common characteristic of technological innovations is that government regulation often takes a long time to catch up with the current state of technology.
- With the rapid evolution that's happening every day in IoT, governments are taking time in catching up and businesses are often left without crucial information they need to make decisions.
- The lack of strong IoT regulations is a big part of why the IoT remains a severe security risk, and the problem is likely to get worse as the potential attack surface expands to include ever more crucial devices.

# IoT Challenges

## Compatibility

- Bluetooth has long been the compatibility standard for IoT devices.
- When it comes to home automation using mesh networking, several competitors have sprung up to challenge Bluetooth's mesh network offerings, including protocols such as Zigbee and Z-Wave.
- It could be years before the market settles enough to crown a single universal standard or unifying architecture for home IoT.
- Continued compatibility for IoT devices also depends upon users keeping their devices updated and patched, which, as we've just discussed, can be pretty difficult. When IoT devices that have to talk to each other are running different software versions, all kinds of performance issues and security vulnerabilities can result.

# IoT Challenges

## Bandwidth

- Connectivity is a bigger challenge to the IoT than you might expect.
- As the size of the IoT market grows exponentially, bandwidth-intensive IoT applications such as video streaming will soon struggle for space on the IoT's current server-client model.
- However, limitations might come also from massive number of connections (especially in the network control plane).
- The server-client model uses a centralised server to authenticate and direct traffic on IoT networks. However, as more and more devices begin to connect to these networks, they often struggle to bear the load.
- Features like intelligent switching between mobile network operators (MNOs) are particularly useful for creating a more reliable and user-friendly IoT product for your customers.

# IoT Challenges

## Energy Consumption

- Most IoT devices (and sometimes even gateways) are battery powered.
- In several cases, applications require IoT sensors to run for months or years.
- For this reason, energy management represents a challenge, that requires optimization in operation, idle/sleep time management, and in general energy-aware operation.
- In the recent years, energy harvesting is playing a relevant role in this area.

# IoT Challenges

## Customers' expectations

- Businesses looking to enter this competitive and innovative sector should be prepared for a market that never sits still and customers who always want a smoother and more advanced experience.
- IoT is an exciting sector with a lot of potential to change the way we live, work and play. But the tech industry, government and consumers alike must get on the same page about issues of security and performance to ensure that the IoT remains safe and productive to use.





# Thank You

-  Prof. Fabrizio Granelli
-  Skype: fabrizio.granelli
-  [fabrizio.granelli@unitn.it](mailto:fabrizio.granelli@unitn.it)
-  <http://disi.unitn.it/>

**PROCEED**  
Introducing Recent Electrical Engineering  
Developments Into Undergraduate Curriculum



# This week's topics...

- Introduction to Programming for IoT
- Programming Fundamentals
- Procedural Programming
- Variables, Expressions and Simple Statements
- Integrated Development Environment (IDE)
- Practice and Conclusions

- Variables
- Data Types
- Comments
- Expressions
- Blocks
- Statements
- Conditionals
- Loops
- Arrays
- Functions and Function Calls

# Introduction to Programming for IoT

## Background

- IoT and the Maker Movement
- Programming languages
  - 1<sup>st</sup> generation (machine)
  - 2<sup>nd</sup> generation (assembly)
  - 3<sup>rd</sup> generation (general-purpose languages, e.g. C/C++, Java, etc.)
  - 4<sup>th</sup> generation (specialized languages, e.g. Python, SQL, Matlab)
  - 5<sup>th</sup> generation (more specialized languages, e.g. Prolog)

# Introduction to Programming for IoT

Why programming?

- Microcontrollers

- Small computers, typically on a single chip
- Used for embedded applications—e.g. an automatic irrigation system
- This is a significant difference from microprocessors—the latter general-purpose and typically used in computers, smartphones, etc.

# Introduction to Programming for IoT

## Hardware

- What is hardware used for?
  - Running actual computations (processor)
  - Remembering things (memory)
  - Talking to other computers (networking)
  - Interfacing with the real world (via sensors and actuators)

# Introduction to Programming for IoT

## Software

- What is software needed for then?
  - You can specify how all these work together to achieve a certain goal:
    - *The processor executes your program*
    - *The memory is where your data—including the program itself—is kept*
    - *The sensors and actuators are what make microcontrollers so exciting: You can combine arbitrary controllers and actuators to implement your ideas*
- The need for software is probably better understood by examining a couple of scenarios

# Introduction to Programming for IoT

## Scenarios

- Think of what kind of code you would need to realize these scenarios
  - A smart irrigation system
    - *Multiple sensors are used to monitor relevant properties—such as temperature and humidity—in a field*
    - *Past measurements are also stored in memory so they can be used when deciding the irrigation time and duration*
    - *Multiple actuators are used for starting and stopping the irrigation system, possibly handling each actuator independently*
    - *Networking functionality enables remote monitoring and controlling of the system*

# Introduction to Programming for IoT

## Scenarios

- Think of what kind of code you would need to realize these scenarios
  - A smart parking system
    - *Multiple sensors are used to monitor the occupancy in roads around a city*
    - *Past measurements are also stored in memory so they can be used to anticipate demand for certain areas*
    - *Multiple actuators are used for signalling—e.g. via lights or large matrix displays—the availability of parking in certain areas*
    - *Networking functionality enables remote monitoring and controlling of the system—e.g. for observing the situation or for posting emergency messages on the displays*

# Classroom Exercise

Work in small groups of 3-4 persons each

- Think of an IoT scenario:
- Express it in a few sentences, answering the following:
  - What is it?
  - Who needs it? Why?
  - How is it used?
- Identify the main components needed:
  - Sensors, actuators needed?
  - Is networking needed?
  - Discuss its costs in terms of hardware (you can just guess) and software (express it as # of person hours)

# Programming Fundamentals

- A quick - but not exhaustive - intro to Programming
- Assumes some familiarity with programming
- Can help as a quick fresh-up

# Prerequisites

Some familiarity with...

- Binary Numbers (and Hexadecimals)
  - What is the Binary equivalent of the current year?
  - What is the Decimal equivalent of '1001101'?
  - What is the Hexadecimal equivalent of the above numbers?
- Logical (aka Boolean) Operations
  - You are required to pass the 'coursework AND exam'
  - You must attend the meeting on 'Monday OR Thursday'
- Using an Integrated Development Environment (IDE)
  - Discussed later...

# Procedural Programming

- Statements
  - Simplest possible commands which can be executed independently
- Line-by-line execution
  - Sequential execution of statements
  - Frequently used statements can be grouped in “functions”
  - Sequence of execution can be modified using “conditional statements” and “loop statements”

# Data Types

- Data Types prescribe the 'meaning' of the stored (Binary) data
  - It can be used to represent numbers, text, logical values, etc.
- C is a strictly typed language
  - All values must be declared to have a 'specific type'

# Data Types

## Basic Data Types of C

- In C, the basic data types are the following:
  - int – used to encode integer numbers (i.e. with no fractional part).
  - double – used to encode real, floating point numbers (i.e. with a fractional part).
  - char – used to encode a character in an alphanumeric value (i.e. a letter in a text).
- C does not explicitly define Boolean/logical value types
  - Typically `int` values are used in its place, where the convention is that zero (0) corresponds to FALSE and other values (typically 1) to TRUE

# Examples of Data Types

- Variables must be 'declared'
- Variables *can be* initialized at declaration

```
x = 2; // cannot succeed unless variable 'x' is declared  
  
char a; // variable declared with type 'char', name 'a'  
  
double z = 121.35; // variable declared with type  
                  // 'double', name 'z', initialize  
                  // to value '121.35'
```

# Comments

What, why, and how?

- **What** - Informal text which does not affect program flow
- **Why** - Helps programmers by serving as extra information reminding why/how code works
- **How** - In C, comments can be added in two main ways:
  - Using the double slash (//) characters to signify that the rest of the line is a comment
  - Using the double markers of slash and star to indicate the start (/\*) and the end (\*/) of a comment

# Examples of Comments

```
// this is a single line comment

int x = 1; // single line comment at end of line

/* Sometimes a comment
can take multiple lines.
In this case the multi-line markers
are more suitable. */
```

# Expressions

What, why, and how?

- Expressions are either part of, or a full statement
- Commonly take the form of:
  - Mathematical expressions
  - Logical expressions
- Expressions consist of:
  - Variables
  - Literals (values expressed in the source code, e.g. 7, or "hello")
  - Operators

# Operators

## Mathematical Operators

- Common arithmetic operators for addition (+), subtraction (-), multiplication (\*), division (/) and remainder (%)
- Commonly take the form of:
  - Addition:  $x+2$
  - Subtraction:  $10-y$
  - Multiplication:  $10*24$
  - Division:  $10/4$  (answer is 2 because it is executed as integer division)
  - Remainder:  $10/4$  (answer is 2 because that's the remainder of the division)

# Examples of Mathematical Operators

```
int x = 10;    // declare and init 'x' with value 10
int y = 1 + 2; // declare 'y' and init with expr. 1+2
y = 10 * 2;   // modify 'y' so its value is replaced
              // with expr. 10*2, i.e. 'y' will be 20
```

# Operators

## Special Mathematical Operators

- Quite frequently, in C/C++ special operators are used
- Unary operators (single operand):
  - ++      Increment by 1. For example, 'x++;' is equivalent to 'x = x+1;'
  - --      Decrement by 1. For example, 'y--;' is equivalent to 'y = y-1;'
- Additional special operators
  - +=      Increment by value. For example, 'x+=10;' is equivalent to 'x = x+10;'
  - -=      Decrement by value. For example, 'y-=4;' is equivalent to 'y = y-4;'
  - \*=      Multiplied by value. For example, 'z\*=2;' is equivalent to 'z = z\*2;'
  - /=      Divided by value. For example, 'w/=3;' is equivalent to 'w = w/3;'

# Examples of Special Mathematical Operators

```
int x = 10; // initialize 'x' with 10

int a = x++; // a gets the original value of x, 10,
             // and increases it by 1

int b = ++x; // x is first increased further to 12, then
             // assigned to b, which becomes also 12

x*=2;       // x is doubled, i.e. becomes 24

int c = x--; // c gets the original value of x, 24, and
             // decreases x by 1 to 23 (c is still 24)

int d = --x; // x is first decreased by 1 to 22, then
             // assigned to d which becomes also 22
```

# Operators

## Logical Operators

- Common logical operators for comparison of smaller than ( $<$ ), smaller or equal ( $<=$ ), greater than ( $>$ ), greater or equal ( $>=$ ) and equal ( $==$ )
  - Note assignment ( $=$ ) is different from equality check ( $==$ ) – this is a common source of bugs especially with programming beginners
- Note equality check with real numbers is less trivial than with integers
  - When you check if `'x==7'` you can be sure it will always work correctly
  - However when you check `'y==1.234'` this could be problematic because of how numbers are encoded in binary—see example in the following slide

# Examples of Comparison Operators

- Because of how real numbers are encoded in binary (with limited precision), you need to allow some flexibility in comparisons
  - For example instead of checking if two numbers are equal, you could check if their difference is within a threshold

```
double e = 0.000001; // threshold is 1 millionth
double x = 9.0-8.3-0.7; // assign x some value which you
                        // want to check if zero
int isZero = x>-e && x<e; // the 'isZero' variable is
                          // true if x within (-e, e)
                          // note that '&&' is the
                          // logical AND operator
                          // discussed next
```

# Operators

## Boolean Operators

- Boolean operators can be used to combine expressions or statements into more complex ones
- Main Boolean operators
  - ! - Logical NOT
  - && - Logical AND
  - || - Logical OR

# Examples of Boolean Operators

## ○ Boolean Operators

- ! - Logical NOT
- && - Logical AND
- || - Logical OR

```
int day = 4; // where Sun is 0, Mon 1, ..., Sat 6

double temperature = 32.7; // assume in Celsius

// equality check has precedence over assignment, so
// that value of 'weekend' is false because (day == 0)
// is false and (day == 6) is also false
int weekend = day == 0 || day == 6;

// the value of 'mustGoToBeach' is set to zero/false as
// the value of 'weekend' is false—even though the value
// of (temperature > 30) is true
int mustGoToBeach = weekend && temperature > 30;
```

# Precedence

Ordering of operations – precedence dictates order of evaluation

- ++ and -- // Postfix increment and decrement
  - ++ and -- // Prefix increment and decrement
  - ! Logical NOT
  - \* and / and % // Multiplication, division and remainder
  - + and - // Addition and subtraction
  - < and <= and > and >= // arithmetic comparison operators
  - == and != // Equality and inequality operators
  - && // Logical AND
  - || // Logical OR
  - = and += and -= and \*= and /= // simple assignment and assignment by addition/subtraction/multiplication/division
- Parentheses can be used to override the order, just like in mathematical formulas
  - Full list at: [https://en.cppreference.com/w/c/language/operator\\_precedence](https://en.cppreference.com/w/c/language/operator_precedence)

# Code Blocks

- Code blocks are groups of statements at the same level
- They define a naming space, i.e. a space where the declared variables are recognized
- Defined using the curly brackets '{' and '}'
- With code blocks, you can limit variable visibility to the space using it, thus making it harder to introduce bugs
- Also with code blocks you can increase the 'locality' of your code (avoiding mixing up variables) and thus increasing code readability

# Examples of Code Blocks

- Code blocks are defined with curly brackets `{` and `}`
- Used properly, they minimize risk of bugs and increase readability

```
{  
    int x = 1;  
    // x is visible here  
    {  
        // x is visible also here, as this is a block nested  
        // in the block where x was declared  
    }  
}  
  
// x is not recognizable here, as it was declared in  
// another code block
```

# Statements

What, how?

- **What** - The building blocks of any program
- **How** – In C there are three main types of statements:
  - Assignments
  - Conditionals
  - Loops
- Statements can also comprise of *functions* and *function calls*
- Terminated with a semicolon `;`

# Hello World!

- The `#include <stdio.h>` statement is used to import a library
  - In this case, containing functions like `printf`
- Statements
  - `printf("Hello, World!\n");`
  - `return 0;`

```
#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    return 0;
}

// produces output
Hello World!!
```

# Conditionals

- Enable the program to respond to a condition (commonly the value of a variable) by following one or another path
- Two types of conditionals:
  1. `if <condition> { one or more statements ... }`
  2. `if <condition> { one or more statements ... }`  
`else { one or more statements ... }`

# Conditionals (IF)

```
#include <stdio.h>

int main() {
    int day = 5; // assume Sat=0, Sun=1, Mon=2, ..., Fri=6
    if(day==0 || day==1) { // if 'day' equals 0 or 1 ...
        printf("Weekend! Hooray!\n");
    }

    return 0;
}

// produces output
Weekend! Hooray!
```

# Conditionals (IF-ELSE)

```
#include <stdio.h>

int main() {
    int day = 5; // assume Sat=0, Sun=1, Mon=2, ..., Fri=6
    if(day==0 || day==1) { // if 'day' equals 0 or 1 ...
        printf("Weekend! Hooray!\n");
    } else {
        printf("Weekday... :-(\n");
    }

    return 0;
}

// produces output
Weekday... :- (
```

# Loops

- Computers can execute the same commands again and again, tirelessly and super-fast
- Looping using the WHILE and FOR keywords
- Two types of WHILE loops:
  1. `while <condition> { one or more statements ... }`
  2. `do { one or more statements ... } while <condition>`
- One type of FOR loops:
  1. `for (<init>; <check>; <step>) { one or more statements ... }`

# Loops (WHILE)

- Condition is checked first
- Statements are executed *while* condition is true

```
#include <stdio.h>

int main() {
    int x = 1;        // define 'x' and initialize to 1
    while(x < 24) {  // repeat while x smaller than 24
        x = x * 2;   // in each loop, duplicate 'x' ...
        printf("x: %d\n", x); // ...and print its value
    }
    return 0;
}

// produces output
x: 2
x: 4
x: 8
x: 16
x: 32
```

# Loops (FOR)

- The *initializer* executes first, exactly once `'int i = 1;'`
- The *conditional check* is evaluated in every iteration *before* the execution of the statements `'i < 6;'`
- The *step increment* executes in every iteration *after* the execution of the statements `'i++'`

```
#include <stdio.h>

int main() {
    for(int i = 1; i < 6; i++) { // repeat for i in 1..5
        printf("i: %d\n", i); // in each loop, print 'x' ...
    }
    return 0;
}

// produces output
i: 1
i: 2
```

# Arrays

- Arrays are *sequences of values*
- Commonly stored in *consecutive addresses* in the computer memory
- Arrays are fixed (once declared, they have a fixed size)
  - There are methods for realizing dynamically-sized arrays but they are beyond the scope of this chapter
- They are accessed directly using an index—defined using square brackets '[' and ']'
  - Arrays have a size (i.e. num of items they contain)
  - First index is 0
  - Last index is size-1

# Array Declarations

- Statically declared
- Cannot change size at runtime
- Values are accessed via the index specified using the square brackets
  - Same for reading and writing values

```
#include <stdio.h>

int main() {
    int a[3]; // create an array of int named 'a' size 4
    a[0] = 10; // initialize the first element to 10
    a[1] = 20; // initialize the second element to 20
    a[2] = 30; // initialize the third element to 30
    a[3] = 40; // initialize the fourth element to 40
    printf("a[0]: %d\n", a[0]); // print first element
    printf("a[3]: %d\n", a[3]); // print last element
    return 0;
}

// produces output
a[0]: 10
a[3]: 40
```

# Array Iterations

- Commonly, arrays are iterated
  - Each value is visited once
  - Usually from first (index: 0) to last (index: size-1)

```
#include <stdio.h>

int main() {
    const int SIZE = 3; // constants in capitals by convention
    int a[SIZE]; // create array of int with given size
    for(int i = 0; i < SIZE; i++) { // for i in 0..SIZE-1
        a[i] = i+1; // make each value equal to its index plus 1
    }
    for(int i = 0; i < SIZE; i++) {
        printf("a[%d]: %d\n", i, a[i]); // print index and value
    }
    return 0;
}

// produces output
```

```
a[0]: 1
a[1]: 2
a[2]: 3
```

# Functions and Function Calls

- Functions enable *modularity*
  - Frequently used code can be *modularized*
  - Increases readability, maintainability, and comprehensibility of code
- Functions interface with the caller
  - They can return a value, like mathematical functions
  - They can take zero, one, or more arguments
- General form

```
<return_type> <function_name>( [optional parameter list] ) {  
    <one or more statements>  
}
```

# Function definition and function call example

- Commonly, arrays are iterated
  - Each value is visited once
  - Usually from first (index: 0) to last (index: size-1)

```
#include <stdio.h>

void square(int side) {
    if(side < 1 || side >> 10) { // side must be in 1..10
        printf("Side must be 1..10!\n");
        return;
    } else { // side is 1..10
        for(int x=0; x<side; x++) {
            for(int y=0; y<side; y++) { printf("*"); }
            printf("\n"); // move to next line
        }
    }
}

int main() {
    square(4);
    return 0;
}

// produces output
```

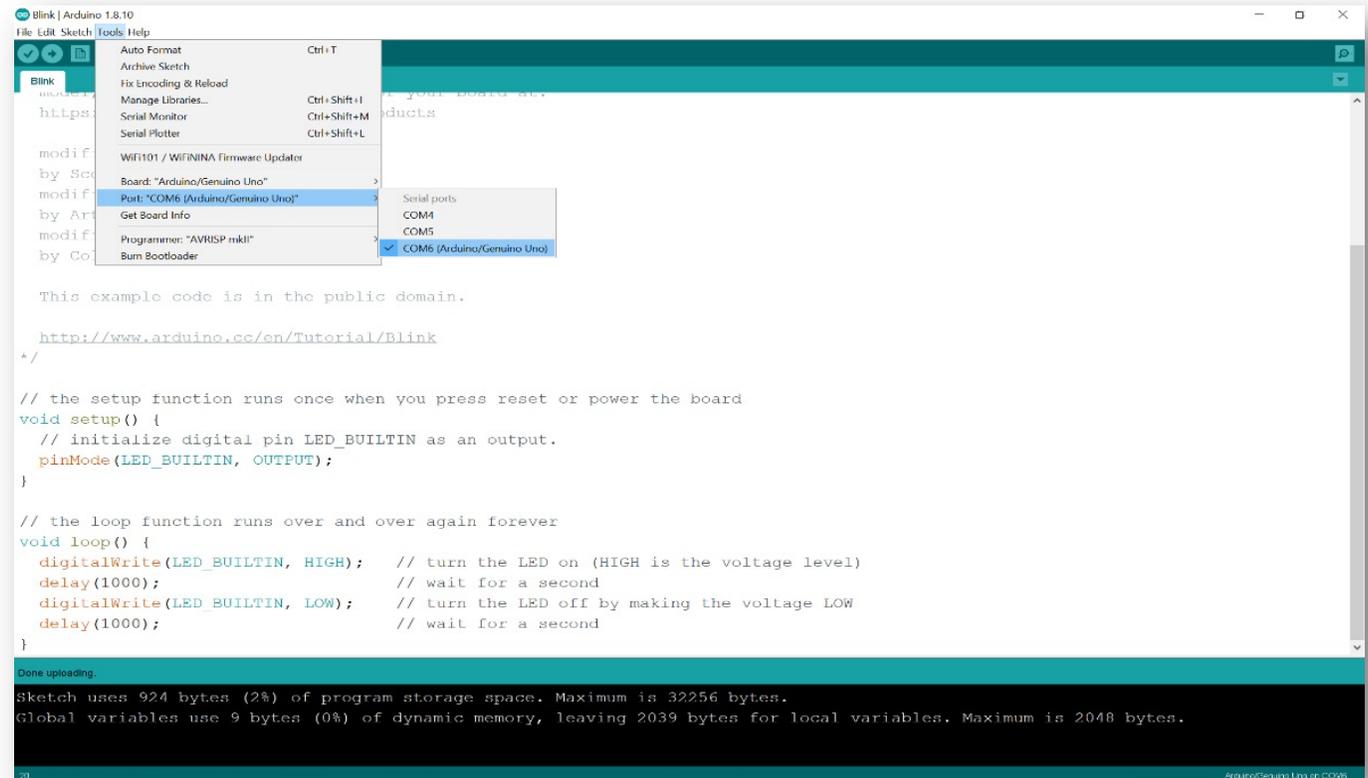
```
****
****
****
****
```

# IDEs

## Integrated Development Environments

○ Specialized software which provides various tools to software developers to:

- Specify
- Manage
- Test
- Debug
- Manage, and
- Deploy code



Arduino IDE

- The basics of programming
- Data Types
- Statements and expressions
- Variables
- Conditionals
- Loops
- Arrays
- Functions and Function Calls
- IDEs (Integrated Development Environments)



[This Photo](#) by Unknown Author is licensed under [CC BY-NC](#)

## Summary

Programming Fundamentals

---



# Thank You

 Dr Nearchos Paspallis

 +357 24 694091

 [npaspallis@uclan.ac.uk](mailto:npaspallis@uclan.ac.uk)

 <http://www.uclancyprus.ac.uk/>

**PROUDER**  
Introducing Recent Electrical Engineering  
Developments Into Undergraduate Curriculum



# This week's topics...

- Introduction
- Development Environment
- Examples
- Additional Resources

# Introduction

## IoT Components

### ○ Endpoints

- Microcomputers (e.g. Raspberry Pi)
- Microcontrollers (e.g. Arduino Uno)
- Sensors (temperature, humidity, accelerometer, compass, light, etc.)
- Actuators (LED lights, relays, servo motors, speakers, etc.)

### ○ Infrastructure

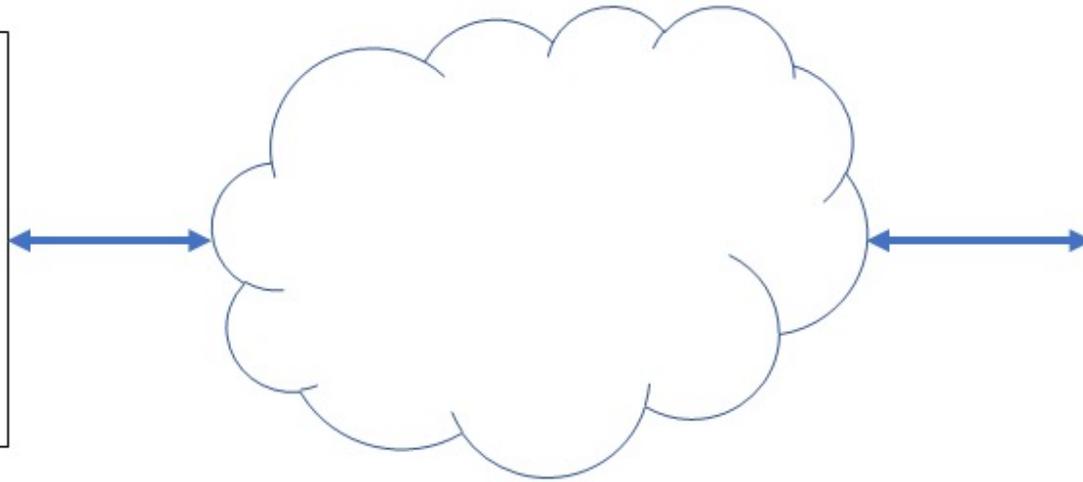
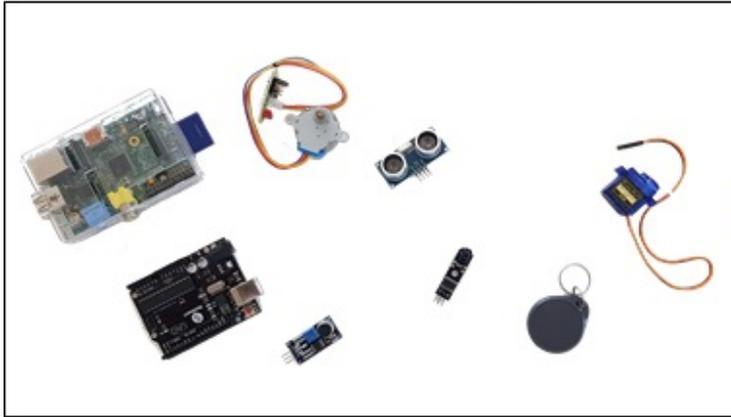
- Networking infrastructure
- Backends (processing, persistence, etc.)

# Introduction

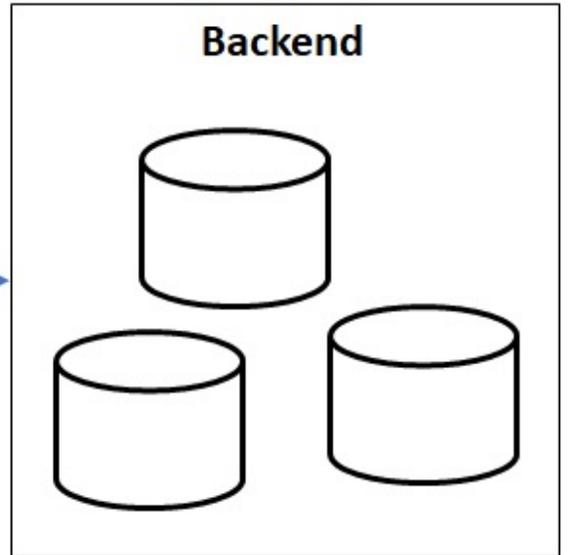
## A Reference IoT Architecture

### Endpoints

Microprocessors, microcontrollers,  
Sensors, actuators, etc.



### Backend



# Introduction

## Why Arduino Uno?

- Many options are available for a microcontroller
- Arduino Uno advantages
  - Very widely available
  - Inexpensive
  - A platform-independent IDE is also available for free
  - Ability to interface with very wide set of hardware components (sensors and actuators)
  - Many online resources (tutorials & examples)
  - Ability to operate independently from a computer (e.g. with a 9V battery)

# Development Environment

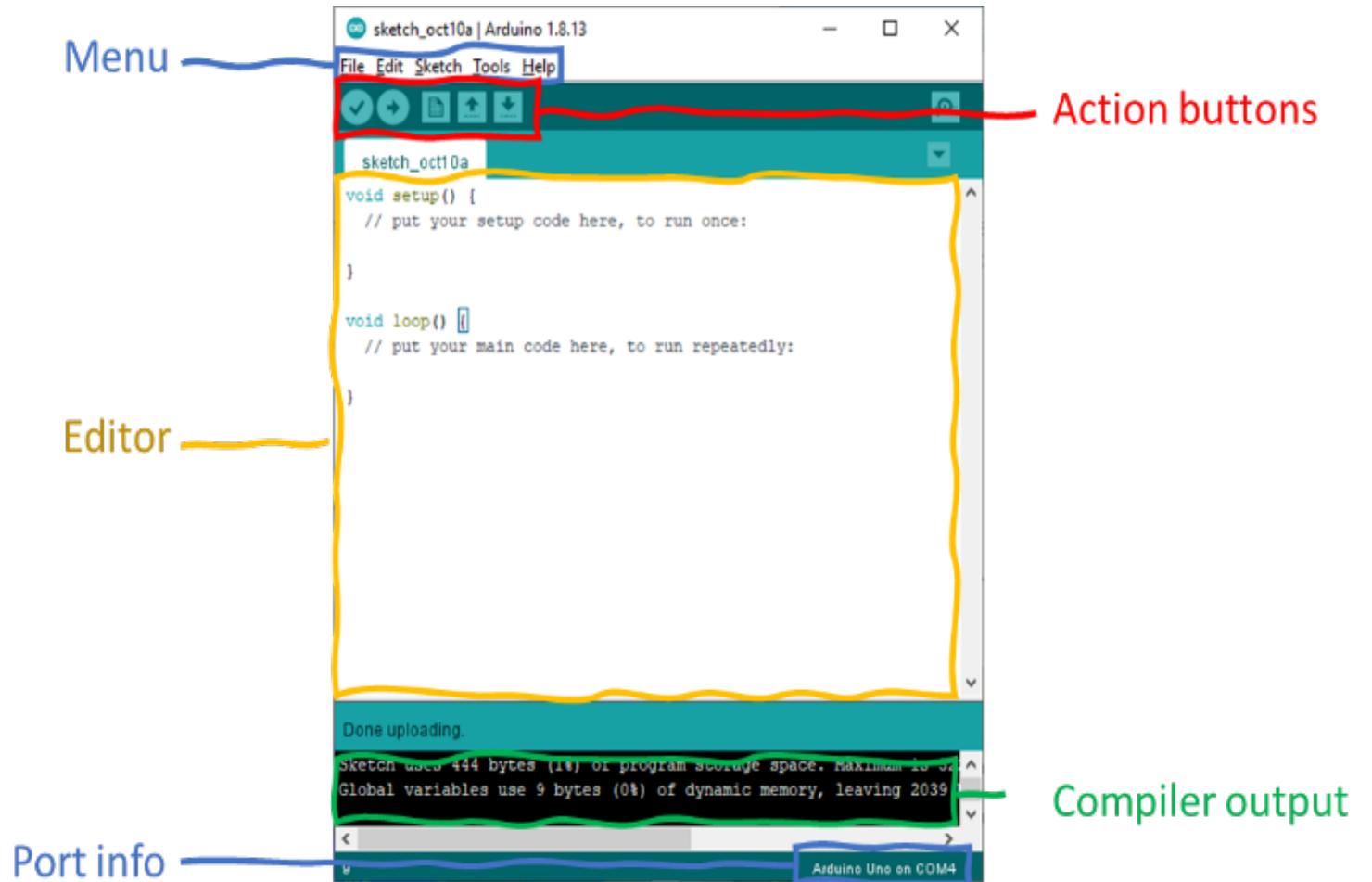
## Arduino IDE

- Integrated Development Environment (IDE)
  - Supports the Development of Software
  - Edit, Compile, Debug code
  - Connect to test Boards, Deploy code, Monitor execution
- Arduino IDE (aka Arduino Software)
  - Available for Windows, Mac OS X, Linux
  - Supports multiple boards (even besides Arduino, e.g. ESP8266)
  - Comes with many Examples preinstalled

# Development Environment

## Arduino IDE Tour

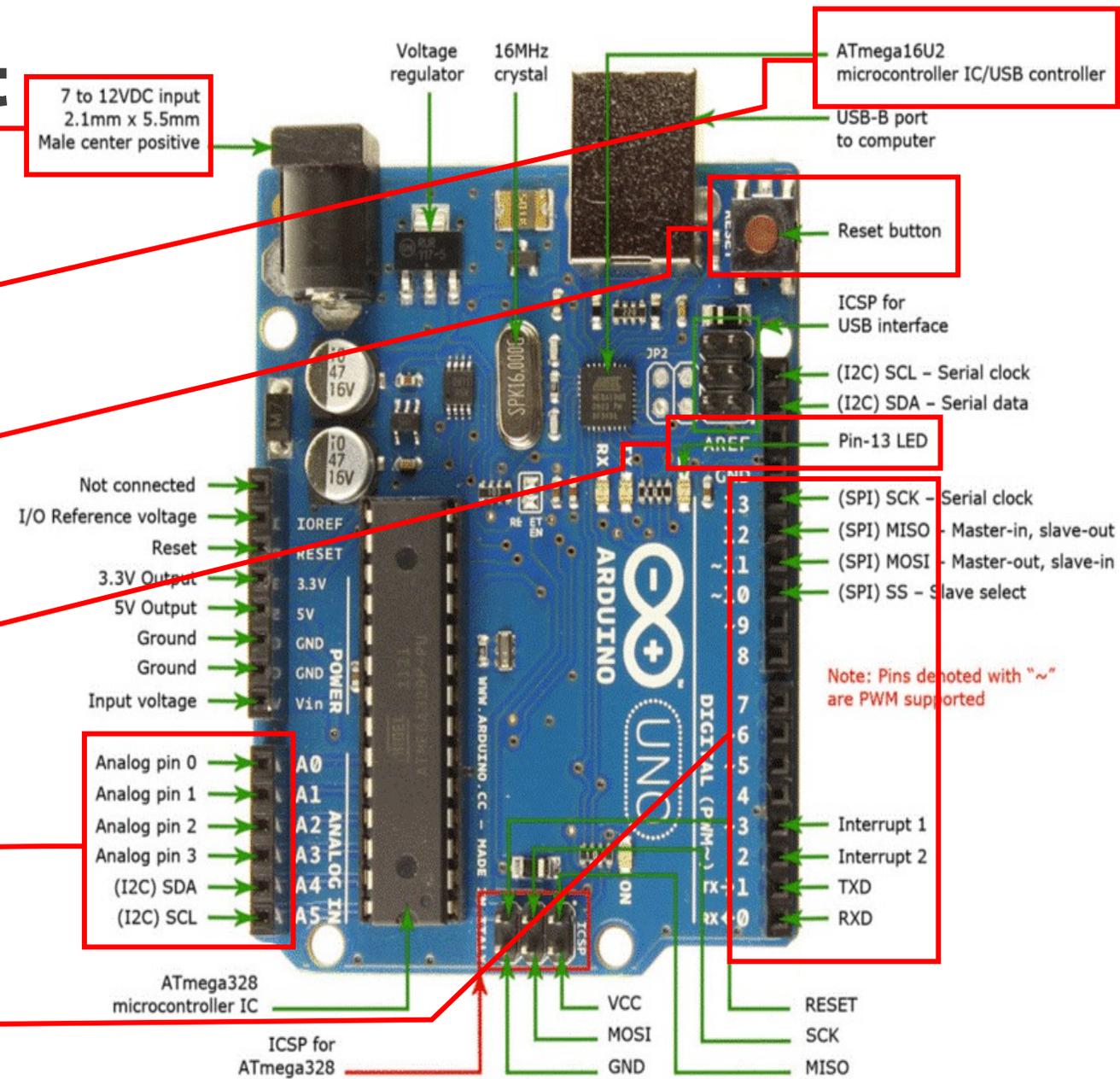
- Menu
  - Access IDE options
- Action buttons
  - Frequent actions (verify, upload, new, open, save)
- Editor
  - Where you type code
- Compiler output
  - Progress of compilation (and error messages if any)
- Port info
  - Info about connected board



# Development Environment

## Arduino Uno Board

- Battery Input
- USB-B Port
- Reset Button
- Embedded LED (Pin 13)
- Analog Ports (Pins)
- Digital Ports (Pins)



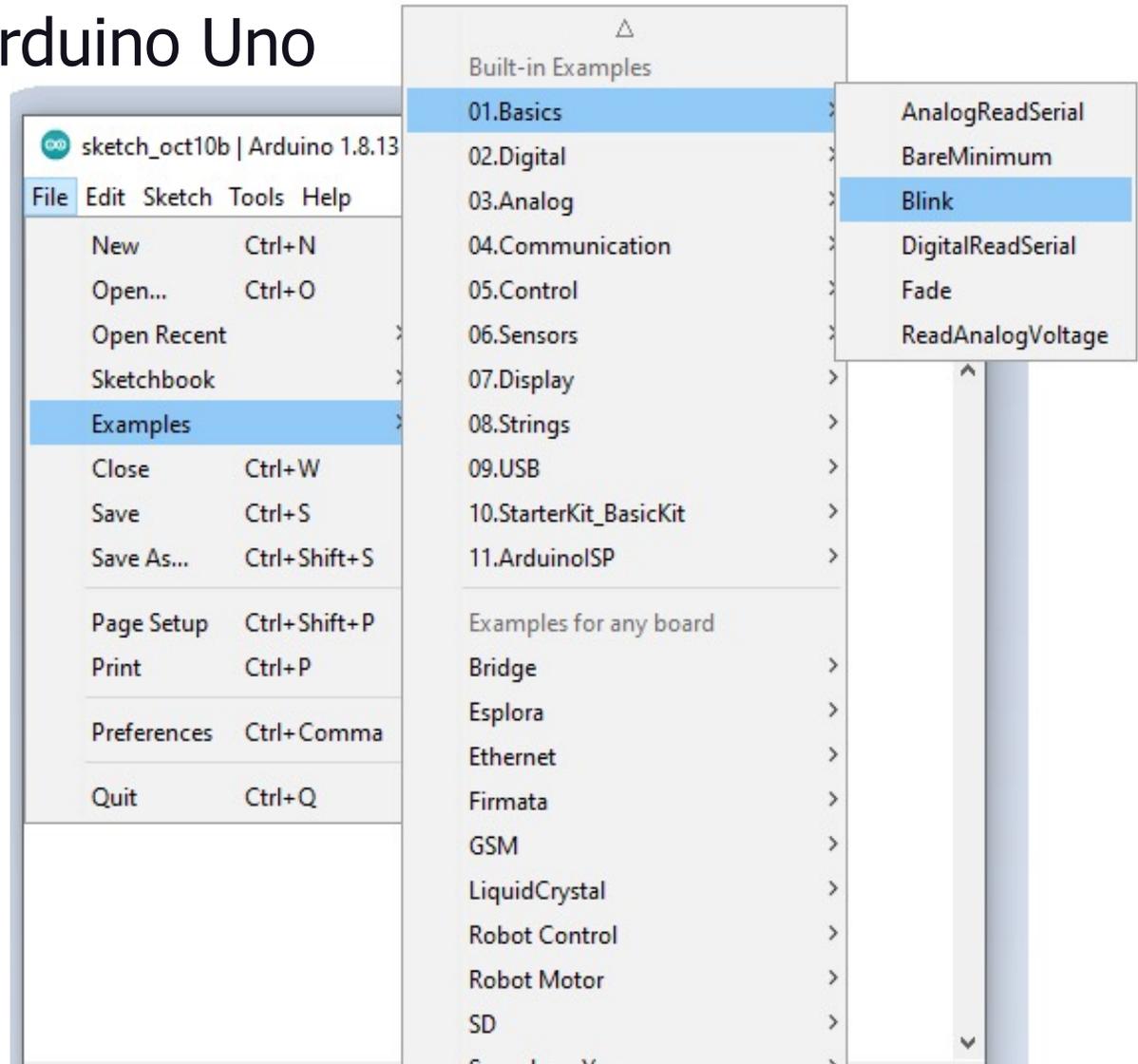
Schematic of a typical Arduino UNO (Jameco.com, 2015)

# Hello “Blinking” World – Minimum Working Example

# Hello 'Blinking' World

A minimum working example for Arduino Uno

- No external components (circuitry) needed
- Utilizes built-in LED light (at Pin 13)
- You can type the code...
- ...or simply load it from Arduino IDE's built-in examples



# Hello 'Blinking' World

## ○ void setup()

- Preparatory steps
- Initializing variables & libraries
- Configuring hardware (e.g. pins as IN or OUT)

## ○ void loop()

- Main programme logic
- Repeats indefinitely
- No wait between loops

```
// the setup function runs once - on reset or power on the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn LED on (HIGH voltage)
  delay(1000);                      // wait for a second = 1000ms
  digitalWrite(LED_BUILTIN, LOW);  // turn LED off (LOW voltage)
  delay(1000);                      // wait for a second = 1000ms
}
```

# Hello 'Blinking' World

## ○ void setup()

- pinMode() configures a given pin number to INPUT or OUTPUT mode
- *Note:* LED\_BUILTIN is a constant—for Uno, it's 13

## ○ void loop()

- digitalWrite() takes 2 arguments
  1. Output pin number
  2. Either HIGH (5 Volt) or LOW (0 Volt = GND)

```
// the setup function runs once - on reset or power on the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

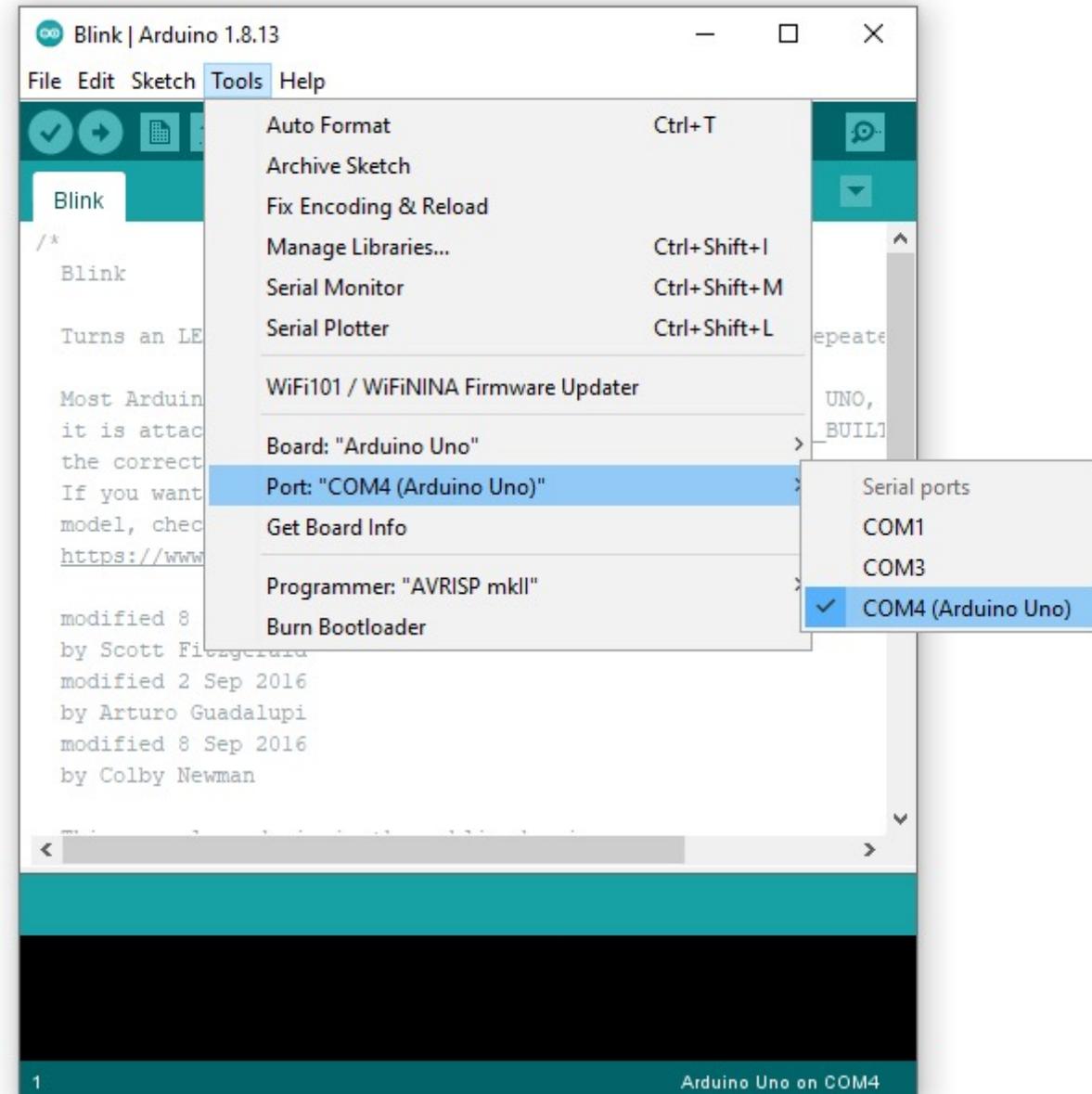
// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn LED on (HIGH voltage)
  delay(1000);                      // wait for a second = 1000ms
  digitalWrite(LED_BUILTIN, LOW);  // turn LED off (LOW voltage)
  delay(1000);                      // wait for a second = 1000ms
}
```

# Hello 'Blinking' World

A minimum working example for Arduino

- Deploying a project to a board

- Connect the board to the computer via an appropriate USB cable
- Make sure the selected port matches the one in the "Port info" panel (bottom right)
- Select the "Upload" button (second option in the "Action buttons" panel)
- Observe the "Compiler output" (black window at the bottom) for possible errors



# Hello 'Blinking' World

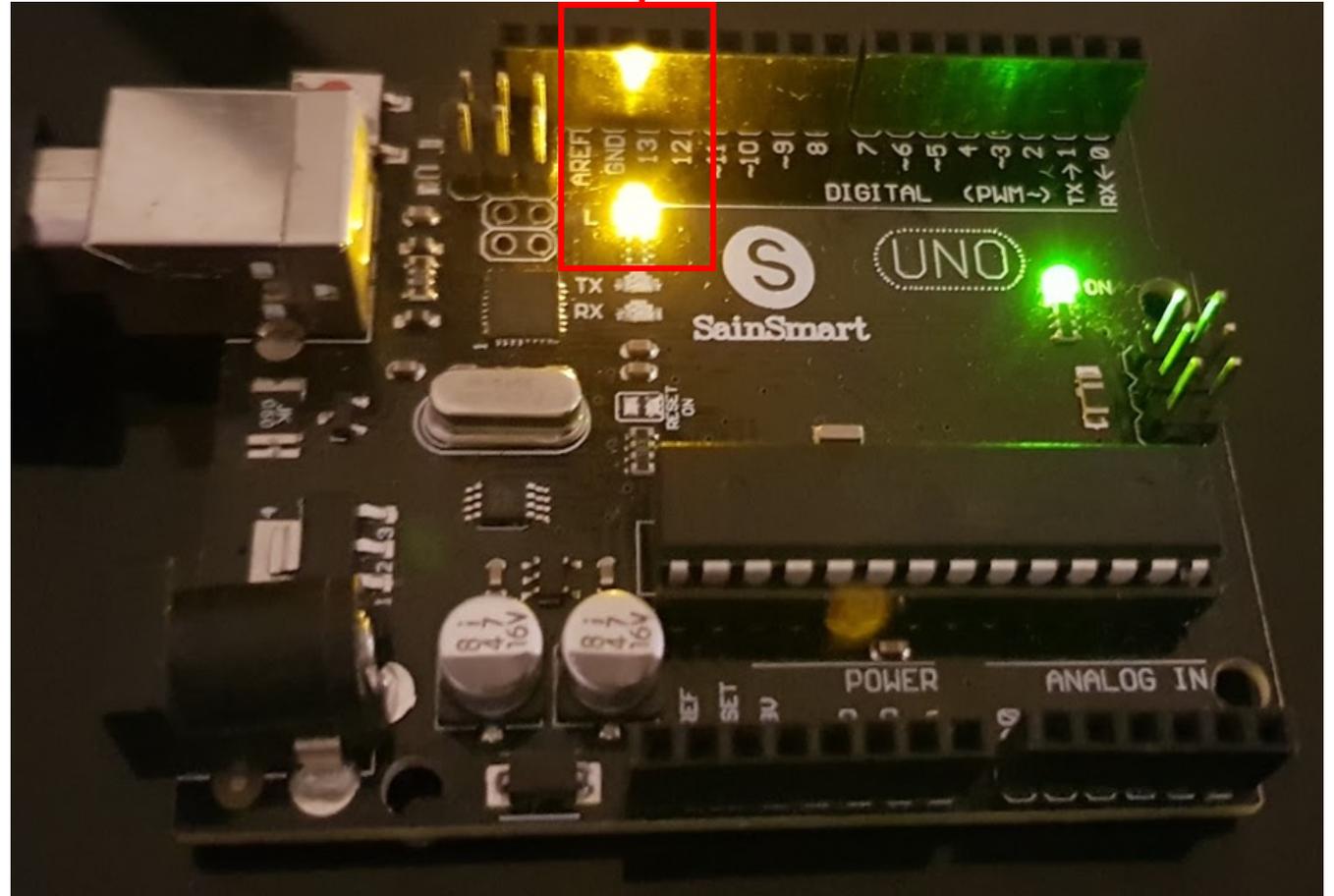
A minimum working example for Arduino Uno

- On successful "Upload"...

- LED blinks!

- Turns on
- Waits 1 second...
- Turns off
- Waits 1 second ...
- Repeats indefinitely

Built-in LED light (Pin 13)

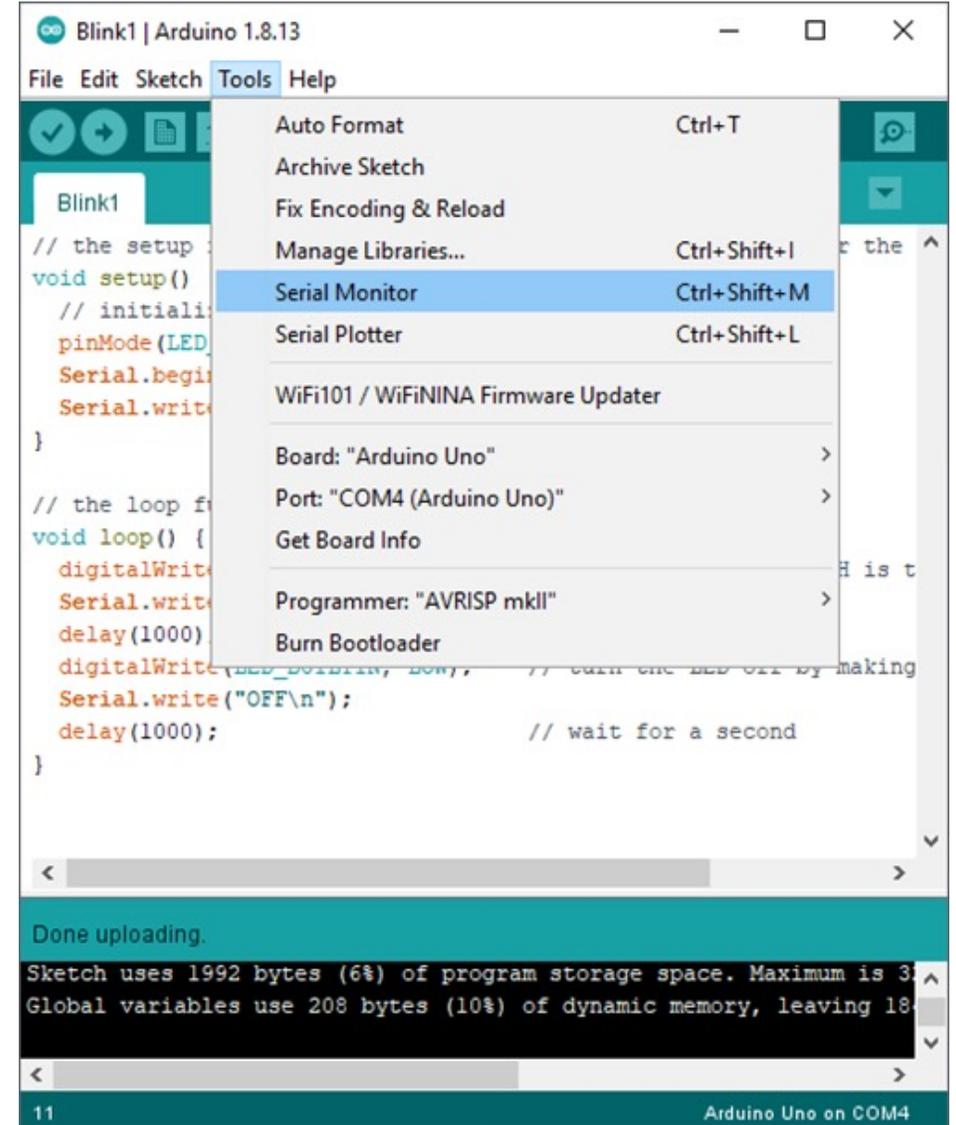


# Monitoring code execution and debugging

# Monitoring code execution and debugging

Using the Serial Monitor tool

- Microcontroller boards typically have no displays
- How can you monitor code execution?
- Arduino IDE offers a **Serial Monitor** tool to runtime output on your connected computer
  - Uses a “serial” connection (thus the name)
  - You can launch it from the “Tools” menu option...
  - ...or via the Ctrl+Shift+M shortcut in Windows



# Monitoring code execution and debugging

Using the Serial Monitor tool

- In the setup function, you need to initialize the Serial Monitor connection
  - `Serial.begin(9600);`
- In your code, you can print messages in the Serial Monitor using the "write" function
  - `Serial.write("Hello World!\n");`
- The '\n' is the special character used to move to the next line (Enter/Return)

# Monitoring code execution and debugging

- Update the 'Blink' code to demonstrate the **Serial Monitor** tool
  - `setup()`: Initialize tool with a predefined rate for communication
  - `loop()`: Write "ON" right after turning the LED on, and "OFF" right after turning it off

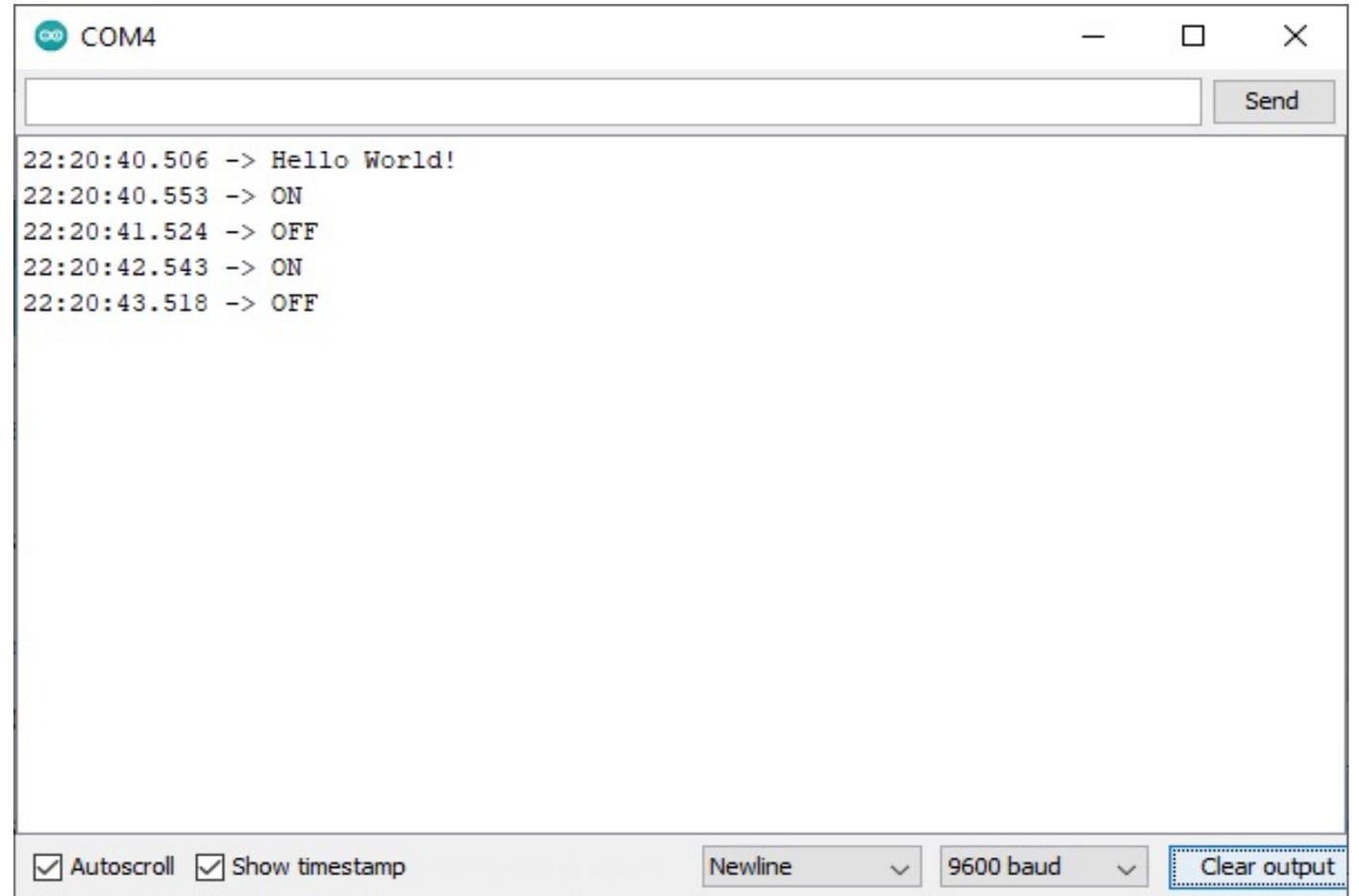
```
// the setup function runs once - on reset or power on the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(9600); // initialize Serial monitor @ 9600 bytes/m
  Serial.write("Hello World!\n");
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn LED on (HIGH voltage)
  Serial.write("ON\n");
  delay(1000); // wait for a second = 1000ms
  digitalWrite(LED_BUILTIN, LOW); // turn LED off (LOW voltage)
  Serial.write("OFF\n");
  delay(1000); // wait for a second = 1000ms
}
```

# Monitoring code execution and debugging

## Using the Serial Monitor tool

- The Serial Monitor showing messages printed by the **setup** and **loop** functions
  - The `setup()` function prints the message "Hello World!" once
  - The `loop()` function prints interchangeably the messages "ON" and "OFF" every 1 second
  - Note how the "ON" and "OFF" lines differ by nearly—but not exactly—1000 milliseconds
  - There are no guarantees as per the exact timing of execution—e.g. at a millisecond precision



```
COM4  
22:20:40.506 -> Hello World!  
22:20:40.553 -> ON  
22:20:41.524 -> OFF  
22:20:42.543 -> ON  
22:20:43.518 -> OFF
```

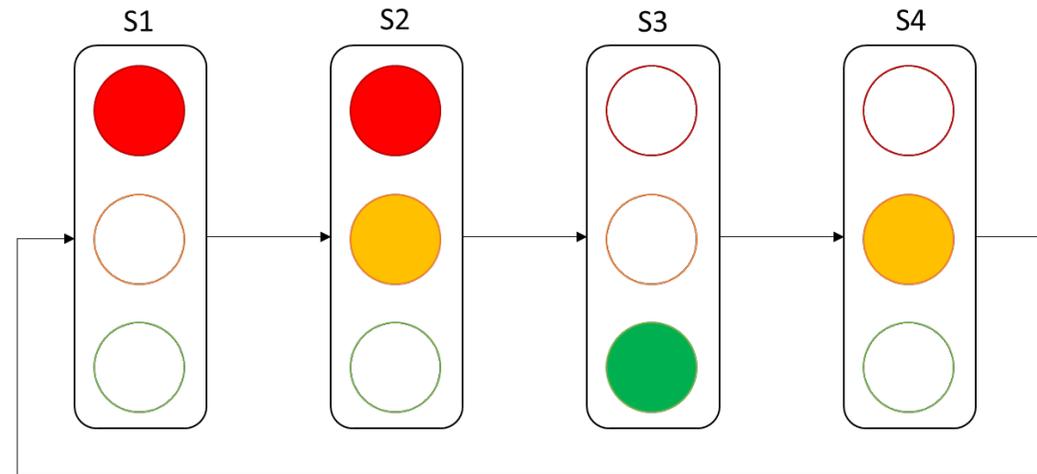
Autoscroll  Show timestamp    Newline    9600 baud    Clear output

# Example: Simple Traffic Lights System

# Example: Simple Traffic Lights System

Implementing simple logic to coordinate 3 LED lights

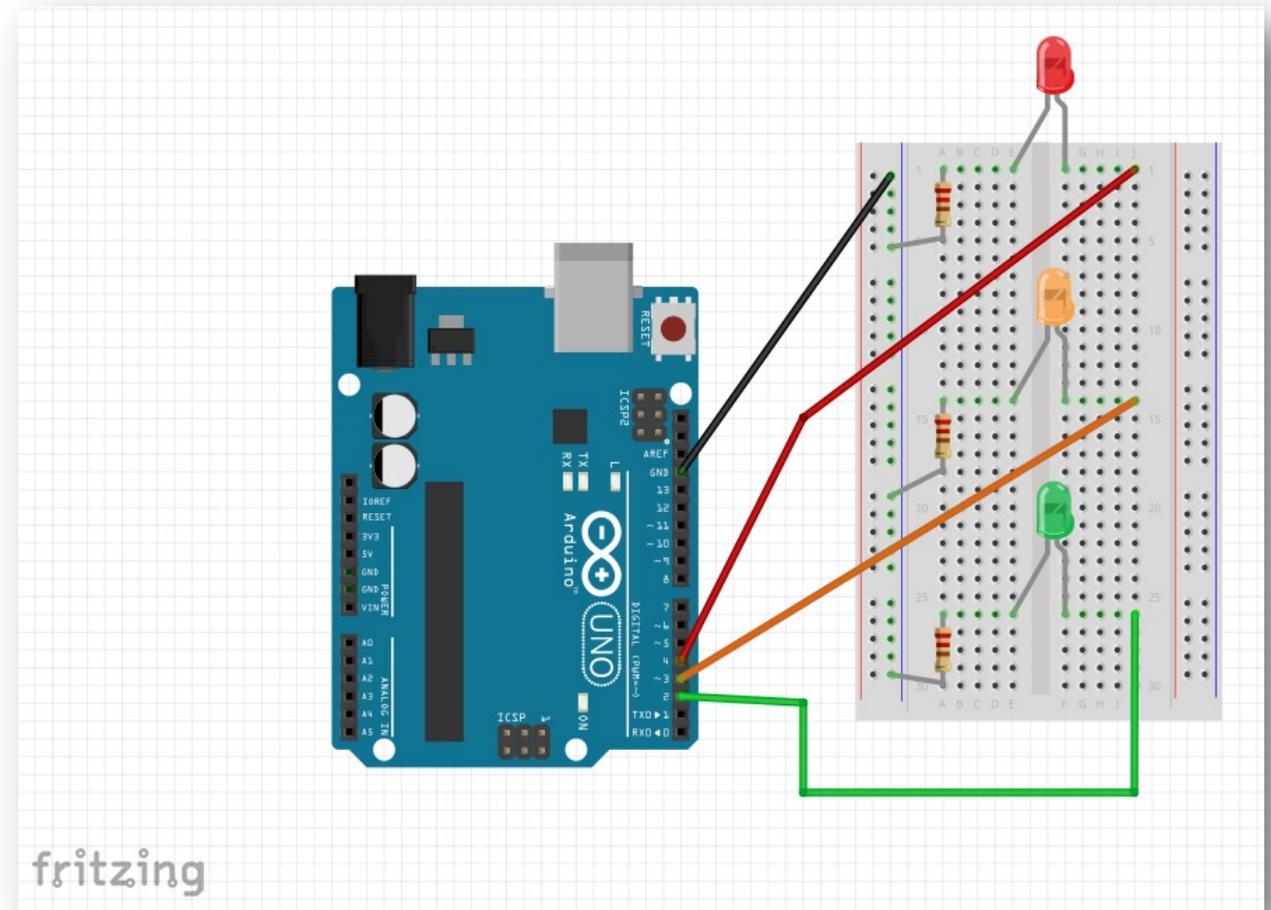
- Traffic lights realize the following state machine
  - Red
  - Red + Yellow
  - Green
  - Yellow
  - ... repeat
- Delay between states varies...
- For simplicity assume it's the same
  - 1 second = 1000 milliseconds



# Example: Simple Traffic Lights System

Implementing simple logic to coordinate 3 LED lights

- Required components
  - Microcontroller board (e.g. Arduino Uno)
  - A breadboard (to help form the circuit)
  - Colored LEDs (red, yellow, green)
  - Resistors (why?)
  - Wires
- Form circuitry
  - Connect anodes of green, yellow, red LEDs to Uno's pins 2, 3, 4 respectively
  - Connect cathodes of LEDs to GND via in-line resistors (typically 220 Ohm)



# Example: Simple Traffic Lights System (1 of 4)

- Define constants
  - For each color, the matching pin number
  - Set all ports as OUTPUT
- Define a variable state
  - Corresponds to the states shown in the *state machine* (see previous slides)
  - Valid states: 1, 2, 3, 4
  - State 0: transitory, right before restarting cycle

```
// Pin numbers for corresponding colored LEDs as constants
const int GREEN = 2;
const int YELLOW = 3;
const int RED = 4;

// holds the current state (must be 1..4, 0 is a transitory one)
int state = 0;

// prepare by setting all 3 pin ports as OUTPUT
void setup() {
  pinMode(RED, OUTPUT);
  pinMode(YELLOW, OUTPUT);
  pinMode(GREEN, OUTPUT);
}

// more code on next slide
```

# Example: Simple Traffic Lights System (2 of 4)

- The loop implements the logic
  - State always increases by 1 (resulting to 1..4)
  - Check value of state and call matching function
  - If already last state (4), then reset to zero
  - At the end of each loop, wait for 1 second

```
void loop() {
  state++; // move to next state
  if (state == 1) {
    state1();
  } else if (state == 2) {
    state2();
  } else if (state == 3) {
    state3();
  } else { // assume state is 4
    state4();
    state = 0; // reset state - will become 1 again when it loops
  }
  delay(1000); // wait for 1 second = 1000 milliseconds
}
// more code on next slide
```

# Example: Simple Traffic Lights System (3 of 4)

- Implement functions for setting state
  - state1(): Set lights to RED
  - state2(): Set lights to RED & YELLOW
  - ...

```
// implement 1st state - RED
void state1() {
    digitalWrite(RED, HIGH);
    digitalWrite(YELLOW, LOW);
    digitalWrite(GREEN, LOW);
}

// implement 2nd state - RED & YELLOW
void state2() {
    digitalWrite(RED, HIGH);
    digitalWrite(YELLOW, HIGH);
    digitalWrite(GREEN, LOW);
}

// more code on next slide
```

# Example: Simple Traffic Lights System (4 of 4)

- Implement functions for setting state
  - ...
  - `state3()`: Set lights to GREEN
  - `state4()`: Set lights to YELLOW

```
// implement 3rd state - GREEN
void state3() {
    digitalWrite(RED, LOW);
    digitalWrite(YELLOW, LOW);
    digitalWrite(GREEN, HIGH);
}

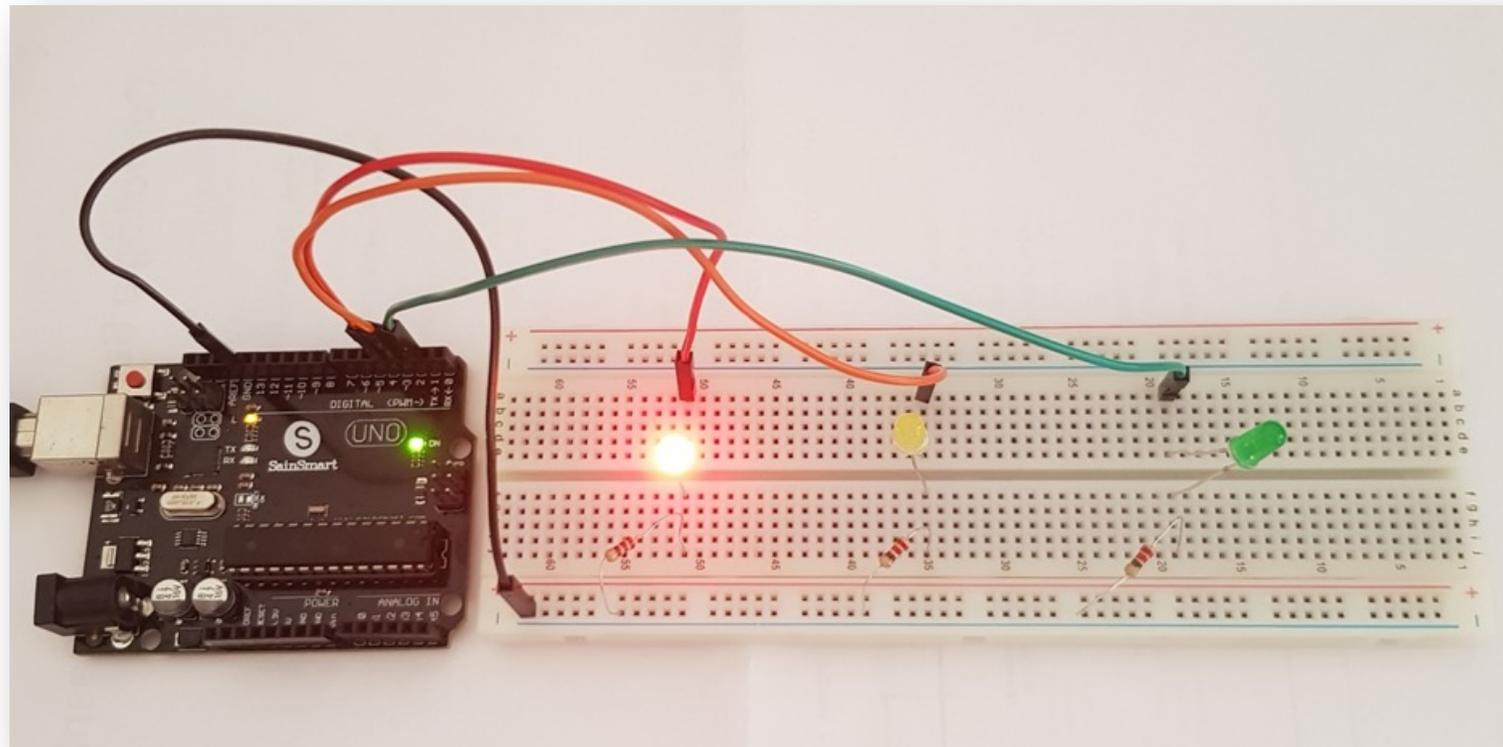
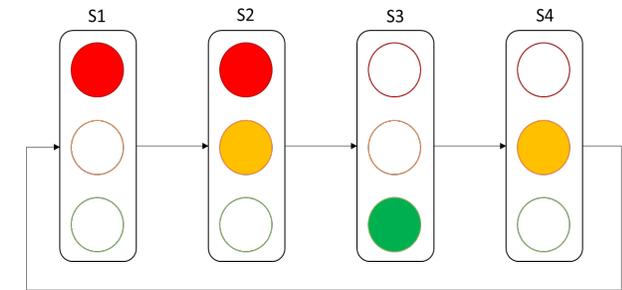
// implement 4th state - YELLOW
void state4() {
    digitalWrite(RED, LOW);
    digitalWrite(YELLOW, HIGH);
    digitalWrite(GREEN, LOW);
}

// end of code listing
```

# Example: Simple Traffic Lights System

Implementing simple logic to coordinate 3 LED lights

- Upload the code
  - Observe the lights transition between the 4 states, once each second



# Example: Simple Traffic Lights System

Implementing simple logic to coordinate 3 LED lights

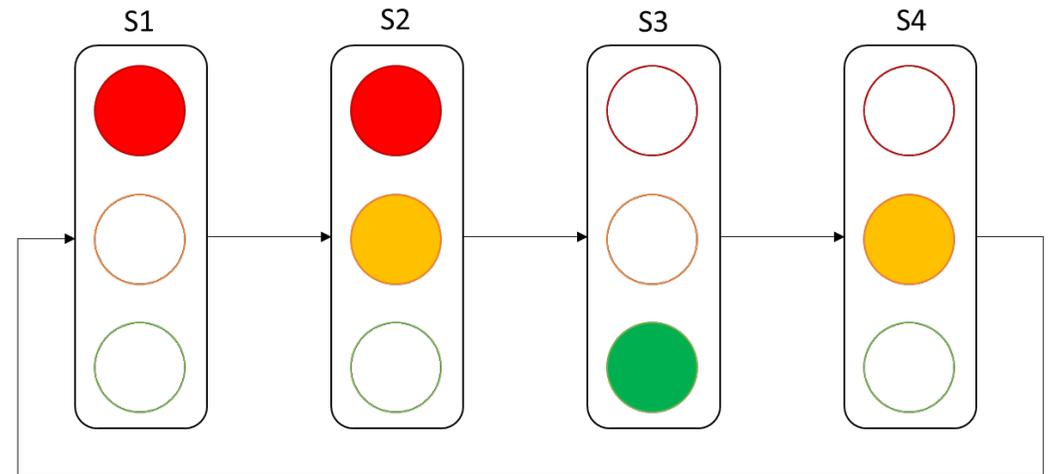
- How about handling Analogue I/O?
- Also, how about handling Input (besides Output)?

# Example: Adaptive Traffic Lights System

# Example: Adaptive Traffic Lights System

Implementing additional logic to coordinate 3 LED lights with adaptive timing

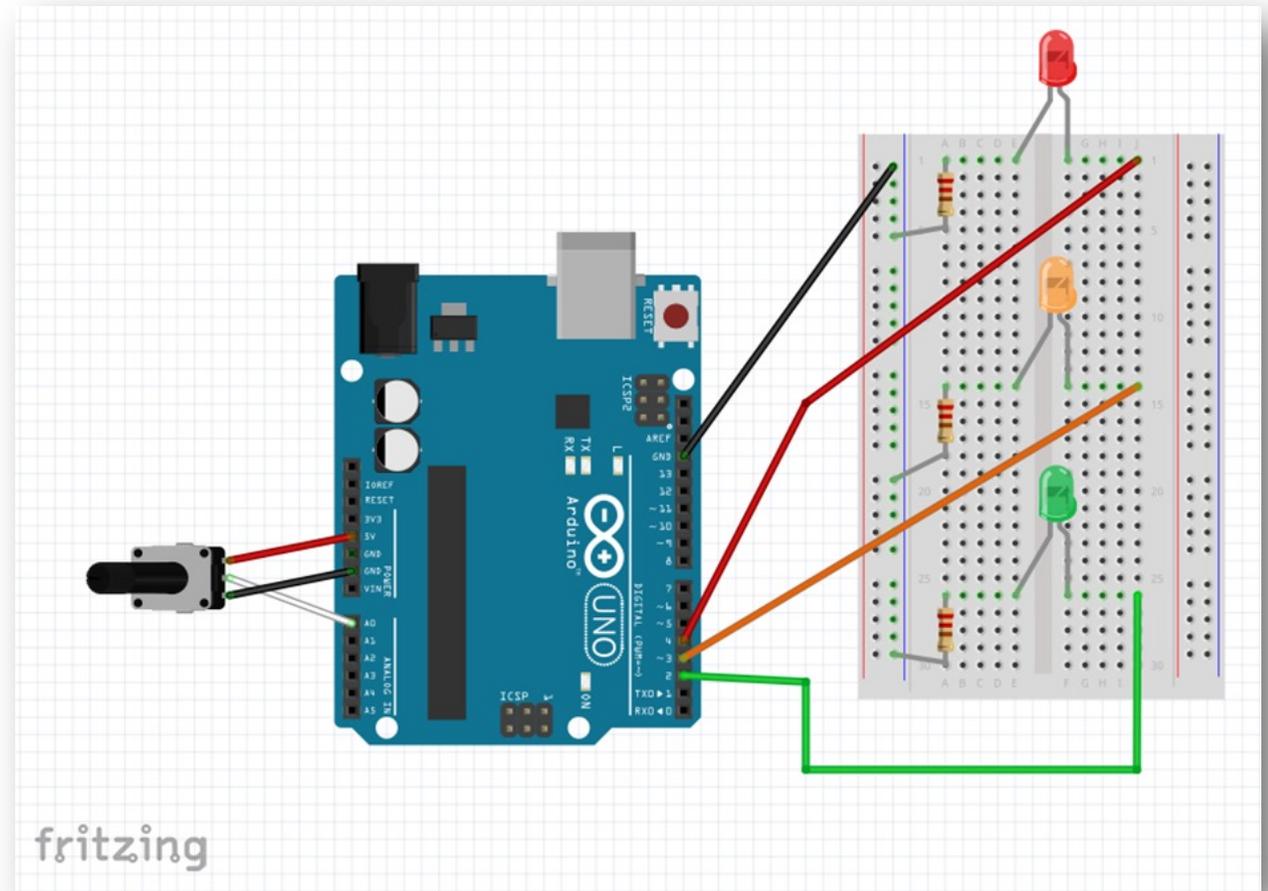
- Traffic lights realize the same state machine as before
  - Red
  - Red + Yellow
  - Green
  - Yellow
  - ... repeat
- But delay between states is controlled
  - Use a potentiometer (i.e. voltage divider)
  - Controls delay – possible values are between  $\sim 0.1$  and  $\sim 4.0$  seconds



# Example: Adaptive Traffic Lights System

Implementing additional logic to coordinate 3 LED lights with adaptive timing

- Additional required components
  - Potentiometer
  - Extra wires
- Form additional circuitry
  - Connect the two end (edge) terminals to the 5 Volt (red) and the GND (black)
  - Connect wiper (middle) terminal to an input analogue Pin, e.g. A0 (white)
  - Analog input port A0 will then be able to sense the selected voltage (range 0-5V) and translate it to an integer (0-1023)



# Example: Adaptive Traffic Lights System

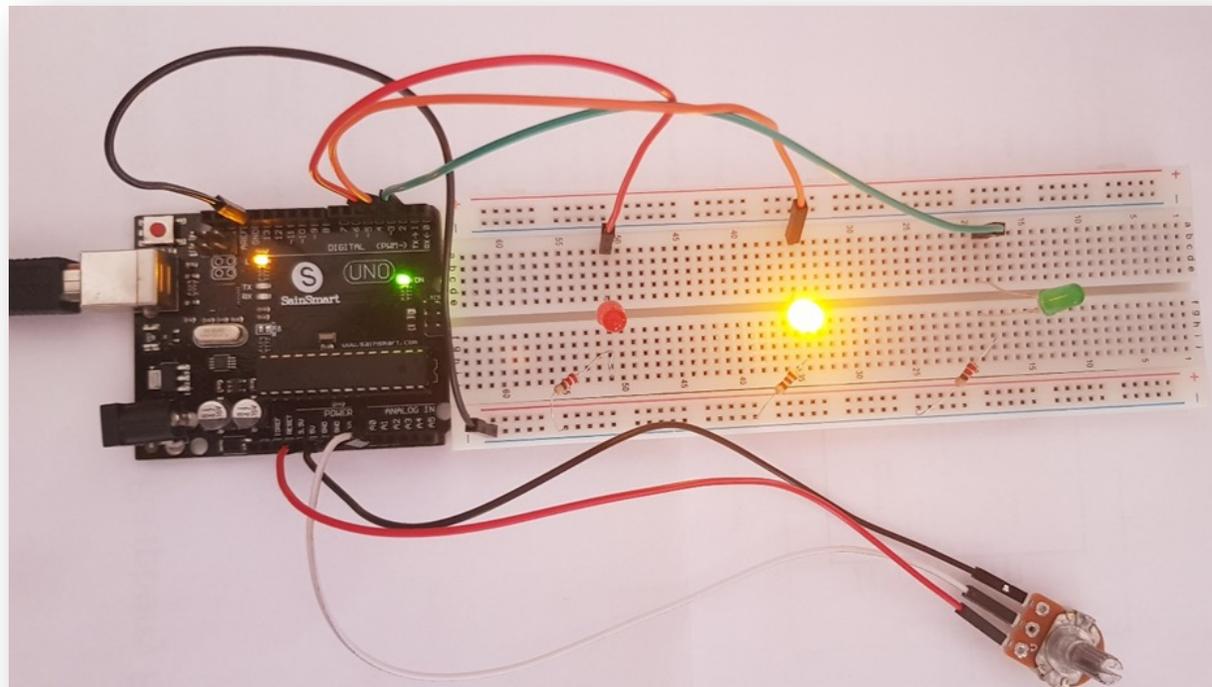
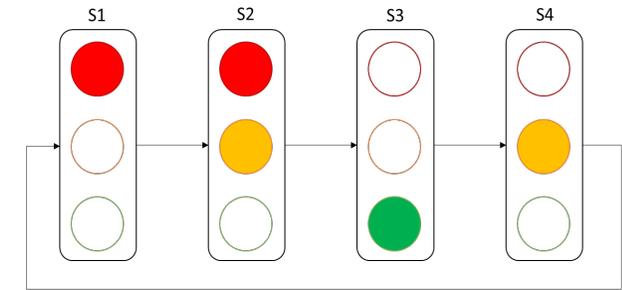
- Simply edit the `loop()` function to realize the adaptive behaviour
  - Define `const` for analogue input pin number (0)
  - In each loop, read the selected value (as indicated by the potentiometer setting)
  - Use the read value (0-1023) to set the delay (from 100 ms to ~4.2 sec)
- Remaining code unchanged

```
const int POT_IN = 0; // port for analogue input
void loop() {
    state++; // move to next state
    if (state == 1) {
        state1();
    } else if (state == 2) {
        state2();
    } else if (state == 3) {
        state3();
    } else { // assume state is 4
        state4();
    }
    state = 0; // reset state - will become 1 again when it loops
}
int val = analogRead(POT_IN); // read int from 0 to 1023
delay(100 + val*4); // set delay from 100 ms to ~4.2 seconds
}
```

# Example: Adaptive Traffic Lights System

Implementing additional logic to coordinate 3 LED lights with adaptive timing

- Upload the code
  - Observe the lights transition between the 4 states
  - Delay between states can be controlled using the potentiometer

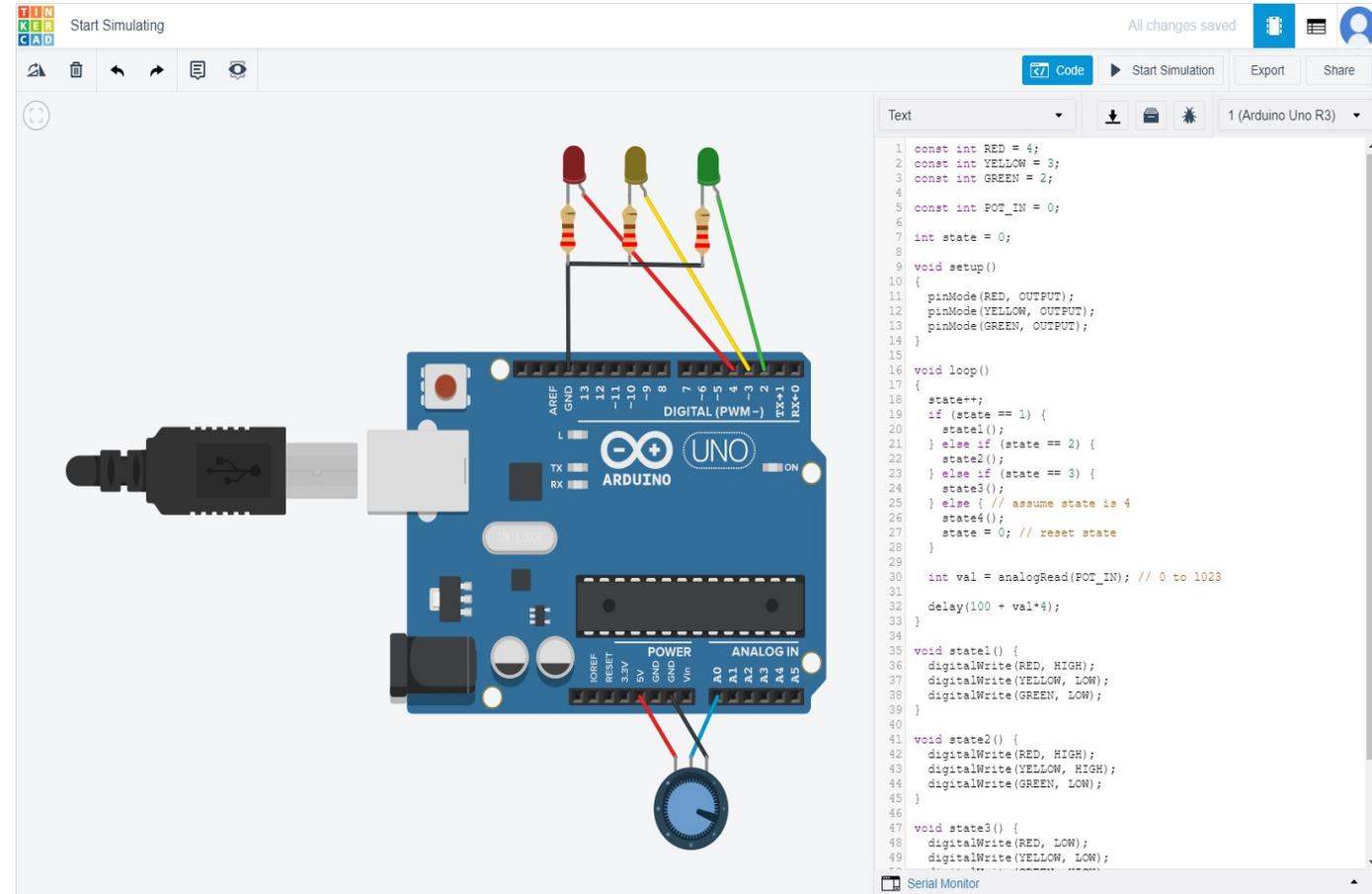


# Additional Resources

# Arduino Simulator

## Additional Resources

- Using a software-based simulator can accelerate prototyping
- AutoDesk's **TinkerCad** is a web-based simulation tool which has support for Arduino Uno
  - <https://www.tinkercad.com>
- This is a fully functional model, capable of running the specified code just like a real setup of a board with Arduino IDE



Implementation of Adaptive Traffic Lights project on TinkerCad

# Online tutorials and examples

## Additional Resources

- About Arduino IDE
  - Online reference guide (from the Arduino IDE menu, select "Help", then "Environment")
  - Or, <https://www.arduino.cc/en/Guide/Environment>
- Language Reference
  - Covers functions, values and structures of C language used in Arduino (from the Arduino IDE menu, select "Help", then "Reference")
  - Or, <https://www.arduino.cc/reference/en/>
- Arduino built-in examples
  - From the Arduino IDE menu, select "File", "Examples", then select one relevant example
  - Or, <https://www.arduino.cc/en/Tutorial/BuiltInExamples>





# Thank You

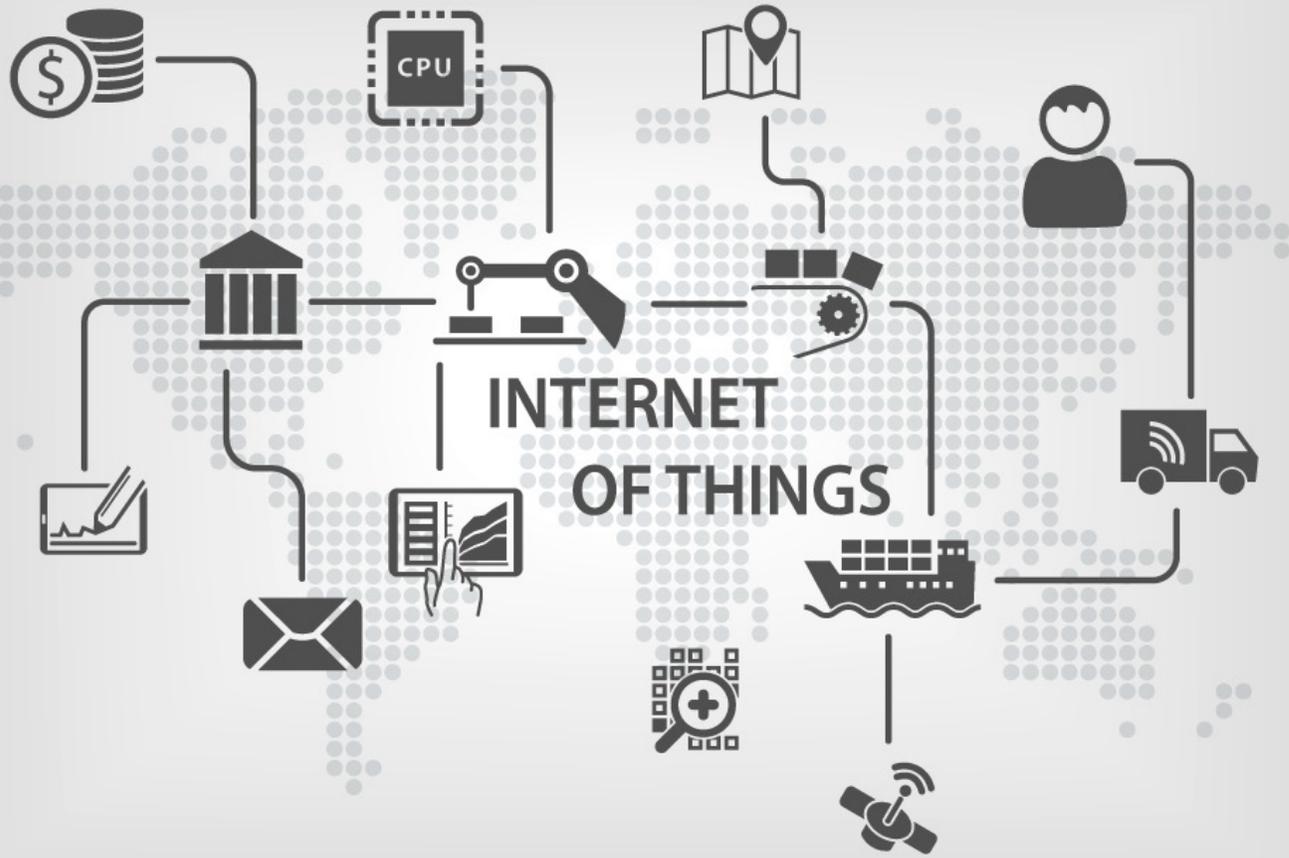
 Dr Nearchos Paspallis

 +357 24 694091

 [npaspallis@uclan.ac.uk](mailto:npaspallis@uclan.ac.uk)

 <http://www.uclancyprus.ac.uk/>

**PROUDER**  
Introducing Recent Electrical Engineering  
Developments Into Undergraduate Curriculum



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

*This publication was produced with the financial support of the European Union. Its contents are the sole responsibility of the partners of IREEDER project and do not necessarily reflect the views of the European Union*



Co-funded by the  
Erasmus+ Programme  
of the European Union

Dr Marios Raspopoulos

Dr Stelios Ioannou

# Introduction to the Internet of Things

Lecture 4: IoT  
architecture and  
components (1 of 2)

Introducing Recent Electrical Engineering  
Developments into undergraduate curriculum

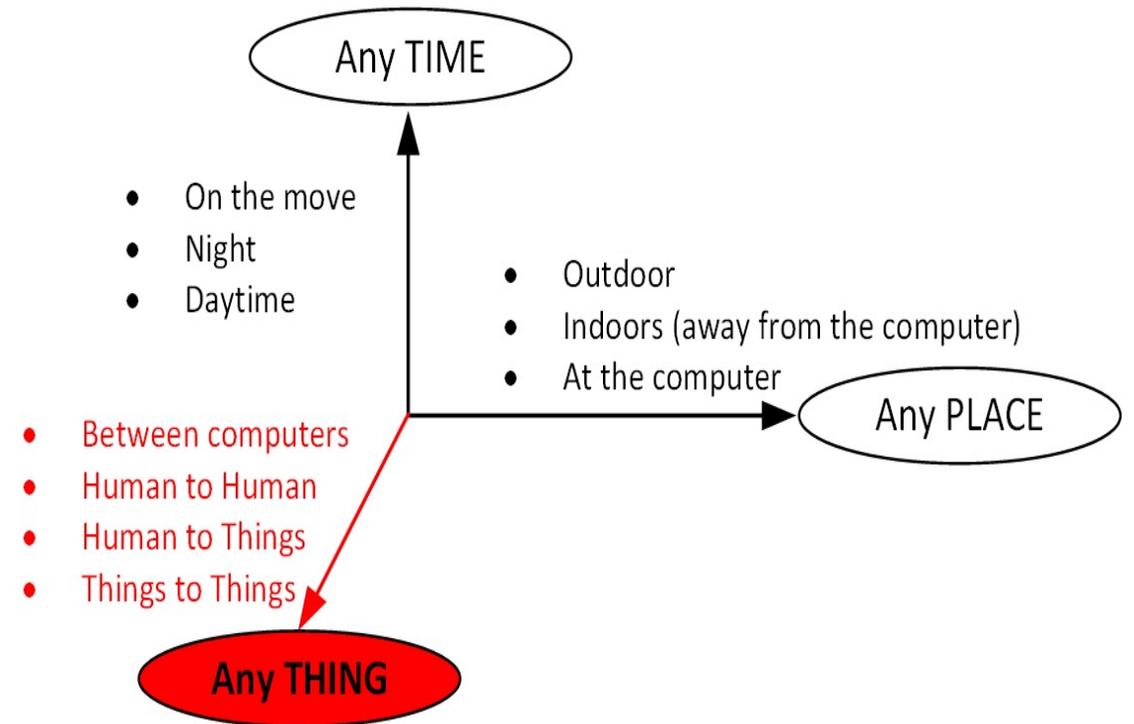
# IREEDER

# This week's topics...

- Characteristics and Requirements of IoT
- IoT Architecture.
  - Introduction to proposed reference architectures
  - 3-Layer and 5-layer Architectures
  - Cloud and Fog-Based Architectures
  - Social IoT Architectures
  - ITU-T Reference Model for IoT
- Major components of IoT (Hardware & Software).
  - Sensors/Actuators and Embedded Technology
  - IoT Connectivity
  - Data Storage and IoT Analytics
  - IoT Cloud
  - User Interface

# Introduction

- IoT is a very popular topic of R&D mainly due to the ubiquitous transformation of computing
- Physical devices have become “smart” being able to sense, communicate in a pervasive way and interact with their environment offering useful applications and solutions to the humankind in a range of activities
  - e.g. health, transport, agriculture etc.
- The 2005 ITU Internet Report [2] adds a 3<sup>rd</sup> dimension to the legacy “ANY PLACE” and “ANY TIME” communication; the “ANY THING” communication



# Introduction

- To ensure connectivity and interoperability it is important that there should exist a reference IoT architecture upon which all IoT applications would be based upon.
- Nevertheless, there is not a consensus on a single IoT architecture, globally agreed.
- Literature mainly reports two architectural models for IoT;
  - a 3-layer architecture
  - a 5-layer architecture
  - some specific purpose architectures.
- In Parallel with the research efforts reported in literature the International Telecommunications Union (ITU) has started in 2012 an effort to standardize the functional architecture model for IoT.

# Section Outline

- Useful Definitions
- ITU-T Technical Overview of the IoT
- Types of Devices
- Fundamental Characteristics of the IoT
- IoT Requirements



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

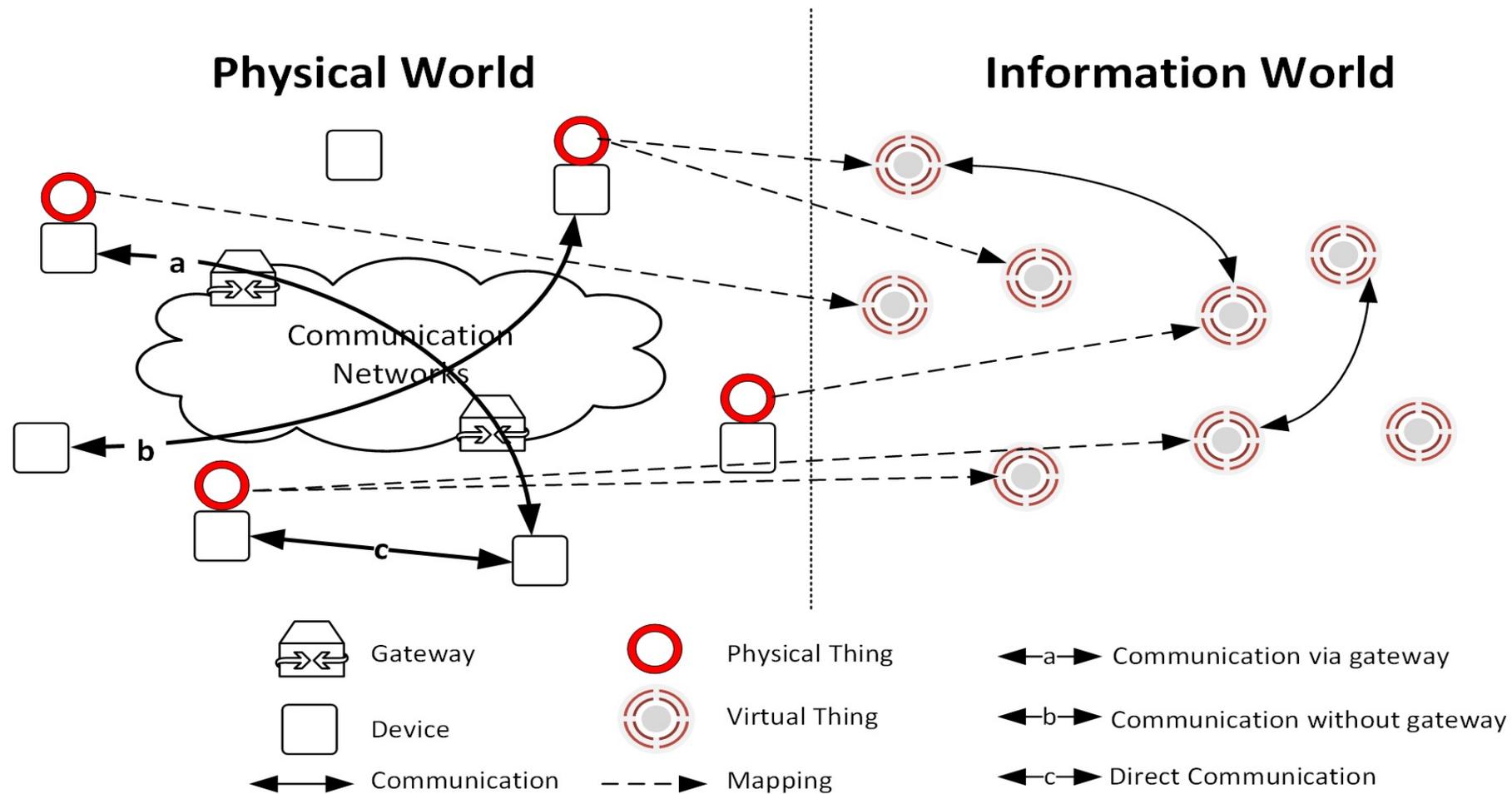
# Section 1

## Characteristics and Requirements of the IoT

# Useful Definitions

- **Device:** In the IoT context, this is a piece of equipment must be able to communicate and could optionally sense, act, capture data, store data or process data. Its only mandatory capability is the communication.
- **Thing:** An object inside the IoT system which is capable of being identified and integrated into communication system.
- **Physical Thing:** An object of the physical world which is able of being sensed, actuated and connected is known as a physical thing (e.g. industrial robots, electrical equipment etc.)
- **Virtual Thing:** An object in the information world capable of being stored, processed and accessed is known as virtual thing. For example, multimedia content, application software etc.
- **Internet of Things:** A global information infrastructure which enables advanced services by interconnecting Things (Physical and/or virtual) based on existing and/or evolving interoperable technologies. The IoT includes functions for identification, data capture, processing, and communication to offer different kinds of applications whilst ensuring security and privacy.

# ITU-T Technical Overview of the IoT

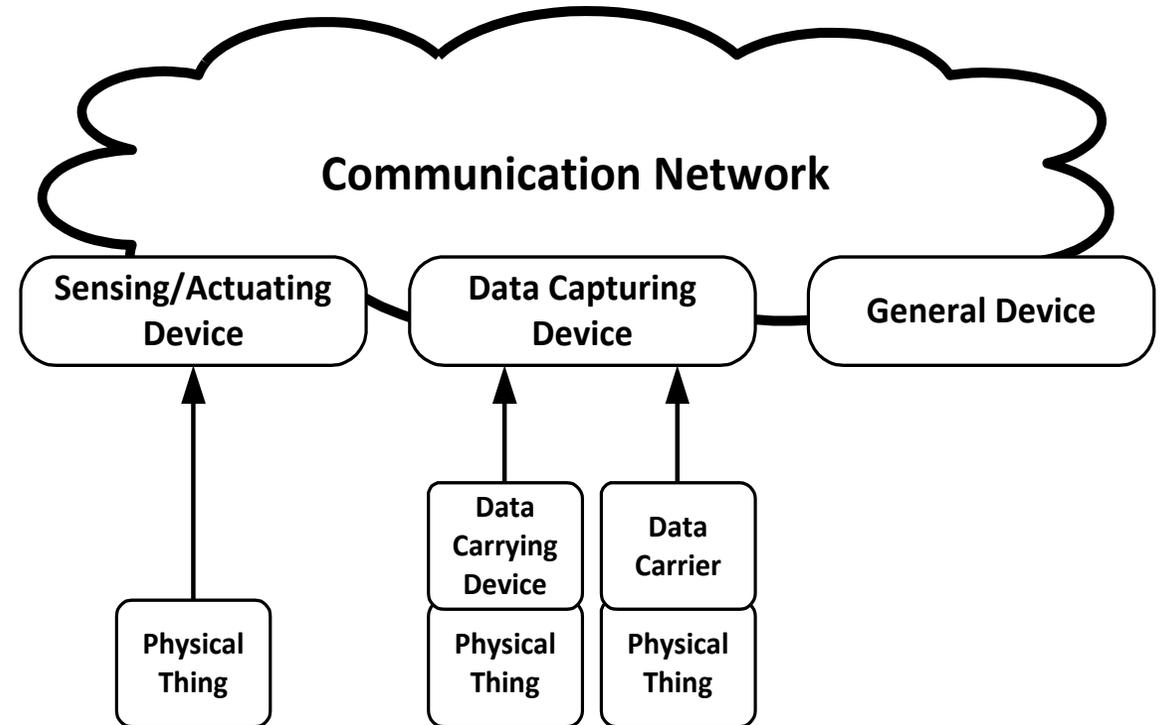


# ITU-T Technical Overview of the IoT

- A physical thing can be mapped (or represented) by one or more virtual things in the Information domain.
- Information is being collected by physical devices (or things) in the physical world and is Communication Networks and the Information domain for further processing.
- Devices may communication with each other either via the communication network (with or without a gateway) or directly without using the communication network or combinations of these communication links.
- Exchange of information not only happens between physical things in the Physical world but also between virtual things in the Information World.
- The communication networks provide capabilities for reliable and efficient data transfer.
  - The network infrastructure may be implemented or realized via existing networking technologies (e.g. TCP-IP networks) or evolving networks following the current telecommunication trends.

# Types of Devices

- **Data-Carrying Device:** A device which is directly attached to a physical thing to indirectly connect it the communication network.
- **Data-Capturing Device:** A device with reading/writing functionalities capable of interacting with the physical things either directly via data carriers attached to the physical thing or indirectly via a data-carrying device.
- **Sensing and Actuating Device:** A device capable of detecting and measuring data within its environment and digitize it. Inversely, it can convert electronic signals from the communication network into actions/operations.
  - Typically, this kind of devices communicate with each other either wirelessly or through wires on a local network and use gateways to connect between different networks.



# Fundamental Characteristics of the IoT

## ITU-T fundamental characteristics of IoT systems

- **Interconnectivity.** Any IoT device can be interconnected with the global Information and Communication Infrastructure.
- **Things-related services.** The IoT provisions services which concern the connected “things” within their constraints such as privacy protections and semantic consistency between physical things and their associated virtual things.
- **Heterogeneity.** Heterogeneous IoT devices with different hardware and networking characteristics get connected and interact with other devices or platforms on various types networks.
- **Dynamic Changes.** While roaming and interacting in an IoT system, devices change their state dynamically.
  - For example, sleeping and waking up, get connected or disconnected while changing their location and speed.
- **Enormous scale.** Usually the number of devices that need to be managed and that of the devices that communicate with each other is significantly larger than the ones that connected to the Internet.
  - This practically means that the communication initialized by devices is much higher than the one that is initialized by humans. Even more important is the management and the analysis of the data generated.

# IoT Requirements

- **Identification-based connectivity:** There needs to be a supported for the “Things” to be connected to the IoT based on their identifiers. This includes a unified processing of identifiers which might be heterogeneous.
- **Interoperability:** Interoperability between heterogeneous and distributed systems needs to be ensured so that a variety of information and services is supported.
- **Automatic Networking:** The IoT network infrastructure should provide control functions for automatic networking including self-management, self-configuration, self-healing, self-optimization and self-protection, to be able to support and facilitate adaptation in different application domains, different communication environments and larger number and types of devices.
- **Autonomic services provisioning:** Services need to be provided by automatically capturing, communicating and processing of the data of the “Things” according to the rules configured by the operators and/or configured by the subscribers. This autonomic service provisioning needs to be based on data fusion and data mining techniques.

# IoT Requirements

- ***Location-based capabilities.*** Things should be able to track their position to facilitate the provision of services which depend on their location.
- ***Security.*** There is an important requirement to integrate different security policy and measures related to the things and their communication in an IoT framework to secure against CIA (Confidentiality, Integrity and Authenticity) for both data and services
- ***Privacy protection:*** Data acquired by “Things” may contain private information of their owners and/or their users. Therefore, it is important that privacy protection is supported during transmission, aggregation, storage, mining and processing of this data while not setting a barrier to data source authentication.

# IoT Requirements

- **High quality and highly secure human body related services:** Services which are based on the capturing, communicating, and processing of data related to human behaviour (e.g. exercise, health, location etc.) automatically or through human intervention should be offered while guaranteeing high quality, accuracy and security.
- **Plug and Play:** It is important for IoT systems to support plug and play capability in order to enable or facilitate on-the-fly generation, composition and acquisition of semantic-based configurations to seamlessly integrate an internetwork of things with the respective applications and efficiently respond to these applications' requirements.
- **Manageability:** Applications in an IoT systems usually need to work automatically with out the intervention or participation of people and therefore the whole operation process needs to be manageable by the relevant entities in order to ensure normal network operations.
- **Scalability:** Any IoT architecture should be highly scalable and be able to support a very large and progressively increasing number of devices that are constantly sending, receiving, and acting on data

## Section Outline

- 3-Layer Architecture
- 5-Layer Architecture
- Cloud and Fog-Based Architectures
- Social IoT
- The ITU-T IoT Reference Model



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

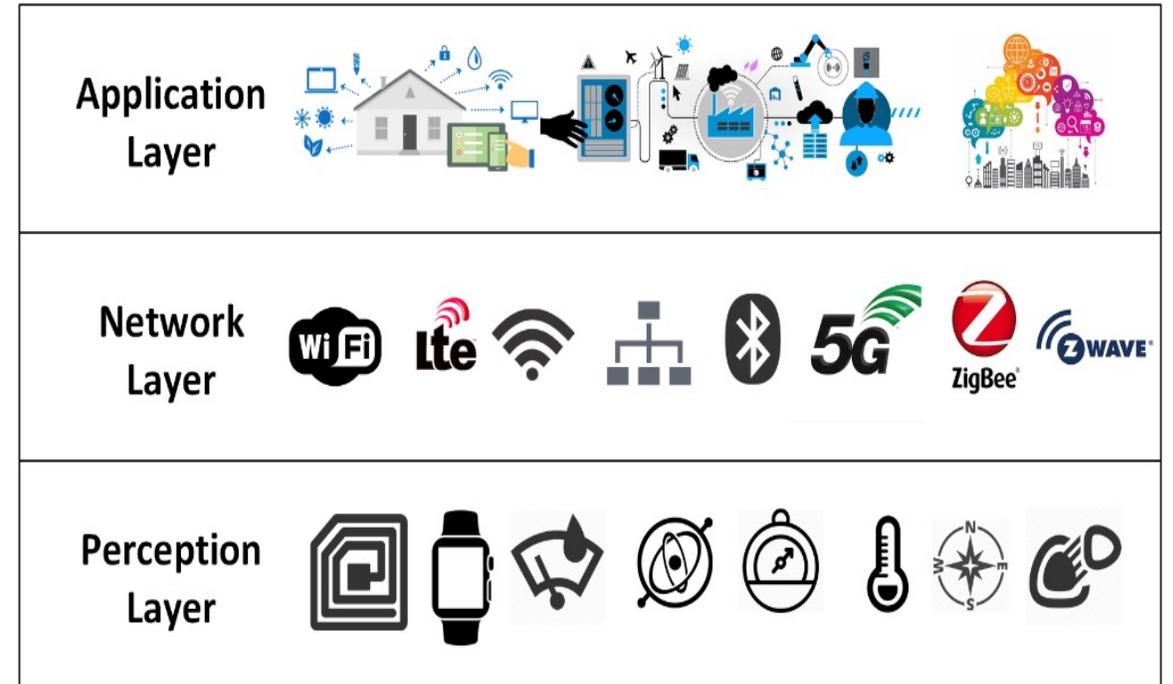
## Section 2

### IoT Architectures

---

# 3-Layer Architecture

- The most basic IoT architecture.
- It was introduced for the first time in 2009
- Consists of
  - Perception Layer
  - Network Layer
  - Application Layer
- Simple and defines the main idea about IoT but it is not sufficient for research and innovation purposes as research focuses on finer and more detailed aspects of IoT.

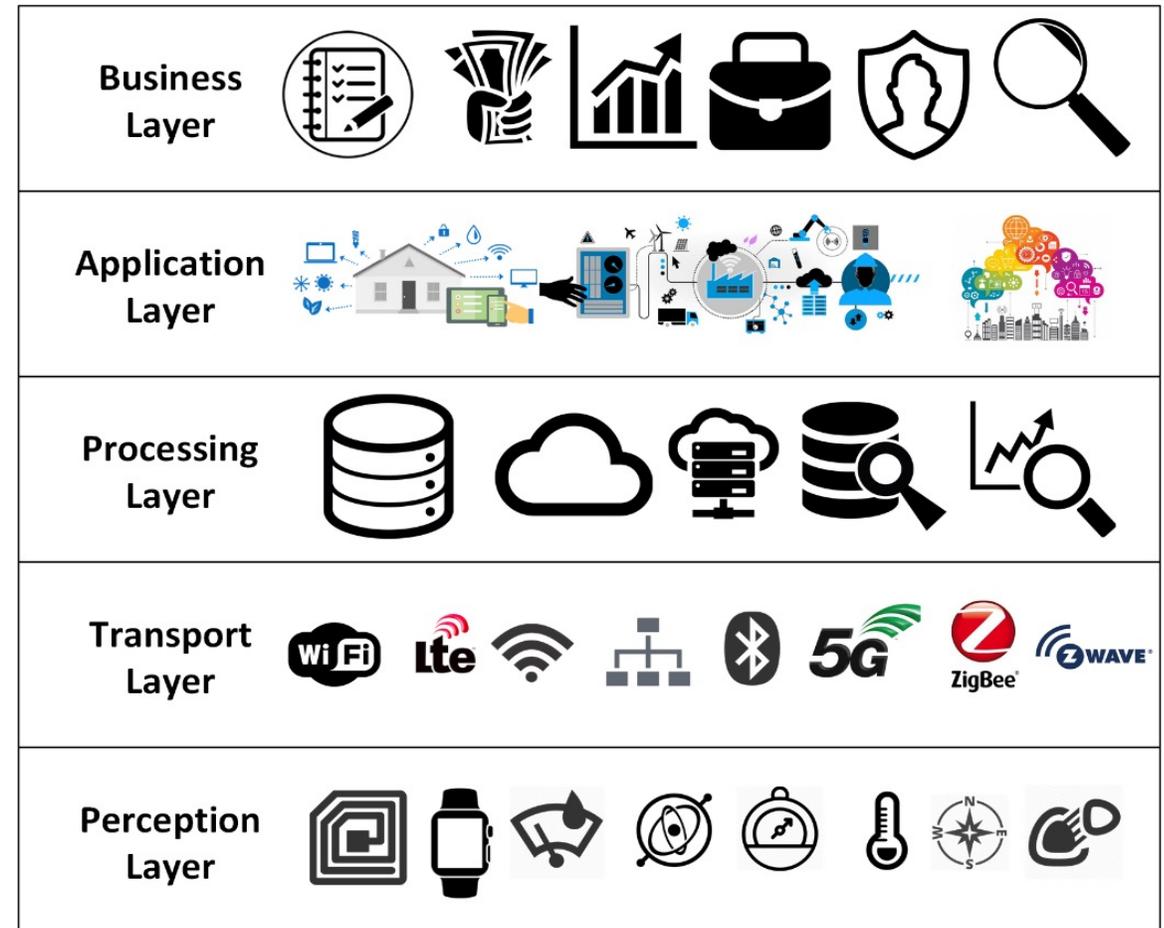


# 3-Layer Architecture

- **Perception Layer.** Provides the mechanisms (sensors) through which the Things perceive their environment.
  - It includes sensing devices that measure different parameters or conditions in their surrounding environment (e.g. thermometers, humidity sensors, inertial sensors etc.) and functions to find and identify objects.
- **Network Layer.** This layer is responsible for the connectivity of Things to other Things, to network devices (e.g. routers, access points etc.), to servers and to the Internet.
  - Includes functions to connect, associate, authenticate to the attached node, transmit the collected information, and/or receive actions to be performed by the Thing from the attached network. Network Layer is implemented using the current but also the evolving network and mobile technologies (e.g. IEEE802.11 standards, 4G, 5G, Zigbee, Bluetooth etc.) but also different types of networking and data collection protocols (e.g. TCP/IP, MQTT, etc).
- **Application Layer.** This layer is responsible for the delivery of the application services to the users/subscribers. It is responsible of utilizing the collected context from the layers below to deliver intelligent applications to the end users (e.g. smart-home, e-health, smart-transport etc.).
  - It is the final goal of the IoT system which consolidates the input from the underlying technologies to offer useful and user-friendly applications to the users. It therefore mostly includes intelligent software development functions.

# 5-Layer Architecture

- The Perception and the Application Layers are the same as in the 3-layer architecture while the Network Layer is renamed to Transport Layer
- Two new layers are added:
  - the processing layer
  - the business layer



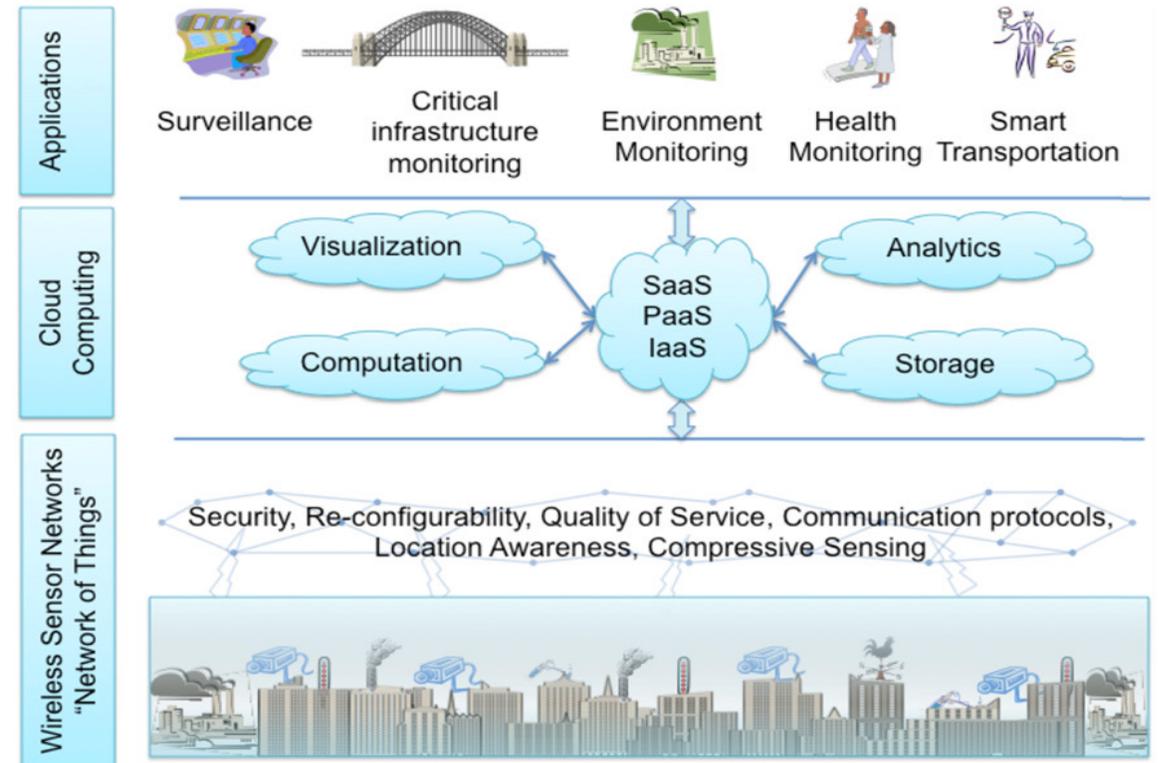
# 5-Layer Architecture

Additional layers compared to the 3-Layer Architecture

- ***Business Layer.*** It is responsible for the management of the whole IoT system, including the business and profit models, the charging, and the privacy of the users. This layer is also concerned with the research and development in the IoT domain.
- ***Processing Layer.*** Also known as the middleware layer, the processing layer is responsible for the storage and the analysis of the data collected at the perception layer and communicated over the transport layer. It includes databases, cloud storage and computing capabilities, data analysis modules etc.

# Cloud-Based Architectures

- Processing is done centrally at cloud computing servers.
- This is a cloud-centric approach where all the applications are built around using the communication network to convey the data back and forth.
- This kind of approach offers the benefits of flexibility and scalability.
- IoT development can be done using storage tools, data mining and machine learning tools, visualization tools and others that are available on the cloud.

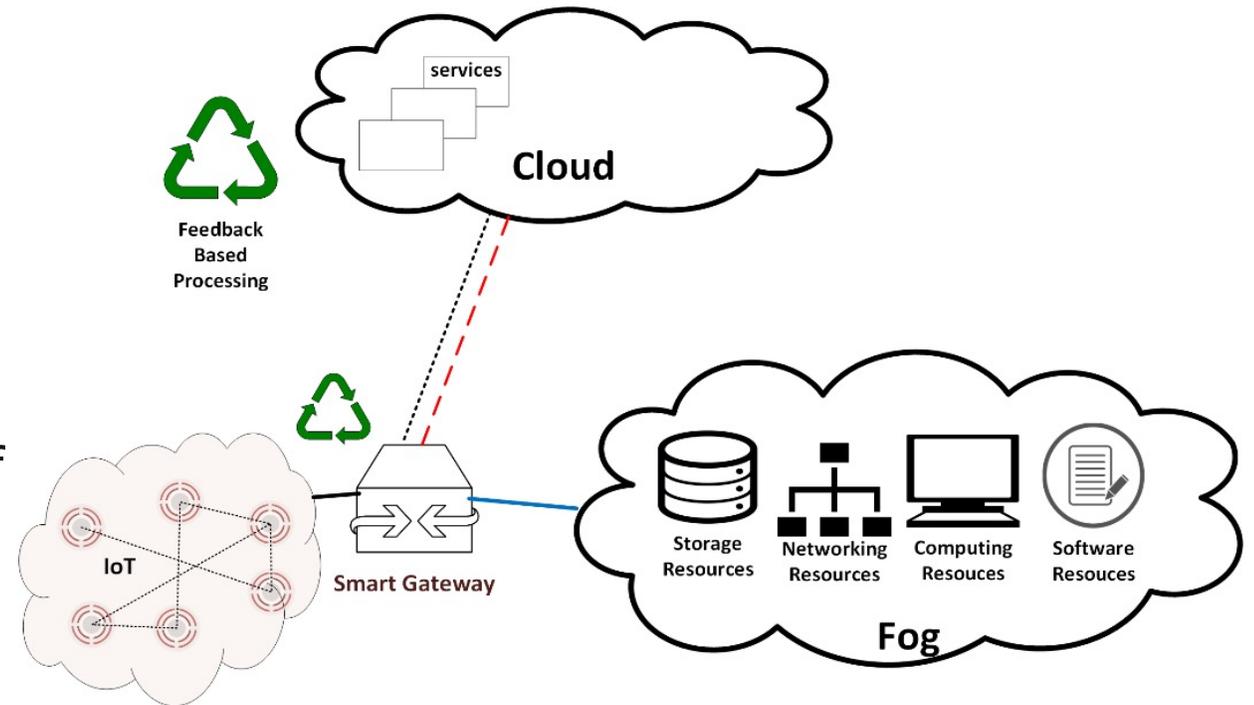


**Conceptual IoT framework with Cloud Computing at the Centre**

J. Gubbi, R. Buyya, S. Marusic and a. M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," Future Generation Computer Systems, vol. 29, no. 7, pp. 1645-1660, 2013.

# Fog-Based Architectures

- The sensors as well as the network gateways do part of the processing and the analysis of the data.
- The capabilities of the cloud computing are extended to the edge of the network which due to the localization of the data the latency is significantly reduced allowing the fast delivery of real-time data and the provision of low-latency and delay-sensitive applications (e.g. real time streaming, e-health applications etc.).
- As some pre-processing is done at the sensors or the smart gateways before reaching the central cloud there might be interoperability and transcoding problems to solve.

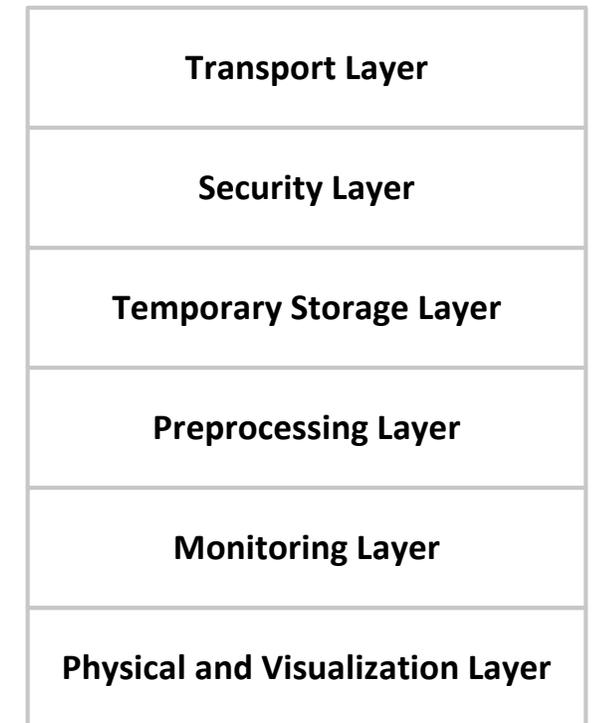


## Smart Gateway with Fog Computing/Smart Network

M. a. H. E.-N. Aazam, "Fog Computing and Smart Gateway Based Communication for Cloud of Things," in *International Conference on Future Internet of Things and Cloud*, Barcelona, 2014.

# Fog-Based Architecture

- The *Physical Layer* includes all the physical and virtual Things as well as the physical and virtual networks that interconnect them.
- The *Monitoring Layer* is responsible for the monitoring of the activities of the nodes and networks in the physical layer.
- The *Pre-processing* layer is responsible of the tasks related to data management such as analysis of the collected data, data filtering, reconstruction and trimming in order to generate more meaningful and useful data for further processing (typical example is the analysis of inertial data from accelerometers, magnetometers and gyroscope in order to extract navigation information like direction of movement, speed, orientation, acceleration etc.)
- The *Temporary Storage Layer* temporarily stores the data generated by the Pre-Processing Layer on the Fog resources. This data is kept on the Fogs only until is uploaded on the cloud and then it is deleted.
- Since there might be generation of private and sensitive data at the underlying layers (e.g. in healthcare, location, military IoTs) there should be functionality to provision security. This is the role of the security layer which includes encryption/decryption, privacy, authentication and integrity measures.
- The *Transport Layer* is responsible for the uploading of the pre-processed and secured data to the cloud.



# Social IoT

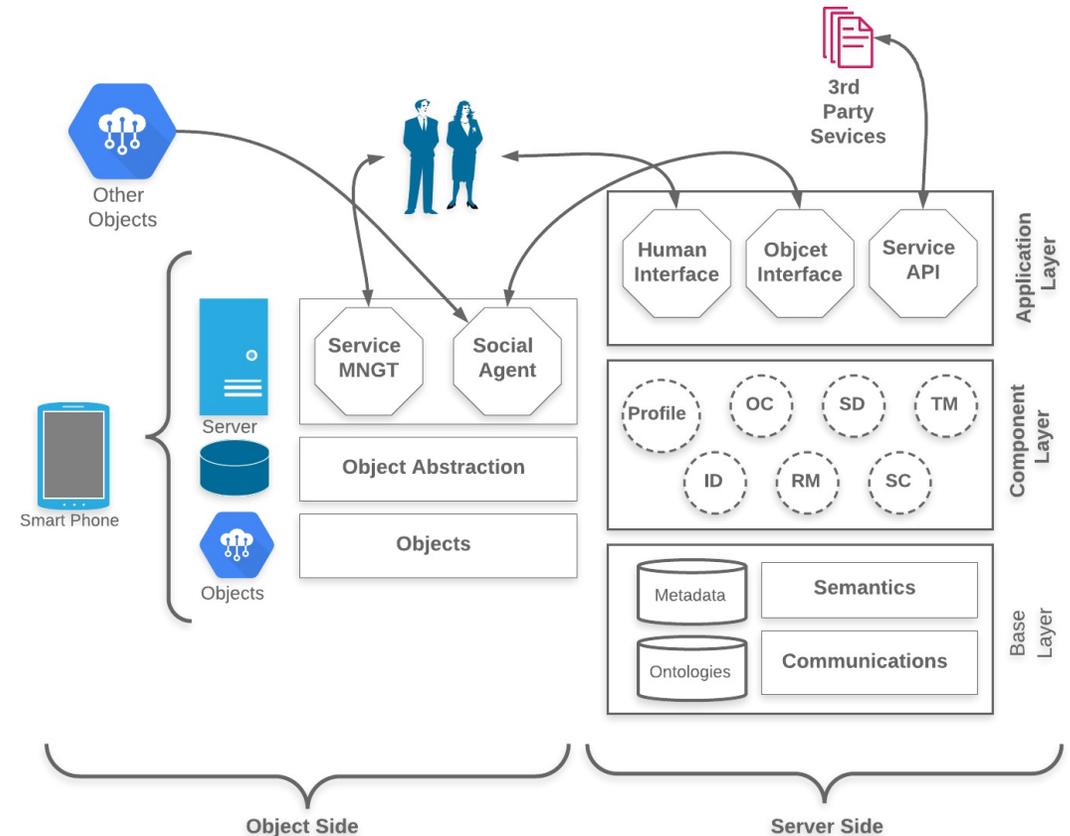
- Based on the notion of social relationships amongst the objects of the IoT systems.
  - analogous to the way that people establish social relationships with one another.
- This approach has 3 main benefits:
  - **Navigability:** Objects can easily discover other objects and establish connections between them easily and in a very scalable way.
  - **Trustworthiness between friendly objects:** Object connected to each other in a friendly relationship can establish a level of trustworthiness between them.
  - **Reusability:** Social Networks already in place for humans can be re-used and stented to apply for IoT related solutions and applications

# Social IoT – A social Networking approach

- The SIIoT model adopts the human social networking approach but it extends it in order to become applicable in the IIoT world. In this context the most important components of a SIIoT architecture are:
  - **ID Management (ID):** Objects need to be identifiable therefore an ID is assigned to each object based on typical parameters like MAC address, IPv6 address, the product code etc.
  - **Object Profiling (OP):** The profile of each object is comprised of information about that object which allows the organization of objects into classes based on their features.
  - **Owner Control (OC):** the owner defines specific policies regarding the operations that can be performed by the objects. This includes security and access control policies as well as the control of the Relationship Management (RM) component.
  - **Relationship Management (RM):** Functionality for the creation, termination and updating of object relationships based on human-controlled settings and relationship rules (e.g. what are the conditions for an object to establish a relationship with another one – for instance a temperature sensor with an air-conditioning unit).
  - **Service Discovery (SD):** To enable objects or services to discover other objects/services. This is analogous to the human world where people search for friends to establish relationships with. Service discovery is performed by each object by querying its social relationship network.
  - **Service Composition (SC):** The objective of this module is to provide better integrated services to the users based on their preferences and needs. Based on the composition and usage of the services included in this component the service discovery can discover the best service for the users. For this reason, this component should include functions for crowd information processing so that information is gathered from main objects and the best response to a service query is obtained.
  - **Trustworthiness Management (TM):** Information is collected regarding the behaviour of objects in order to define their reliability and estimate their trustworthiness.

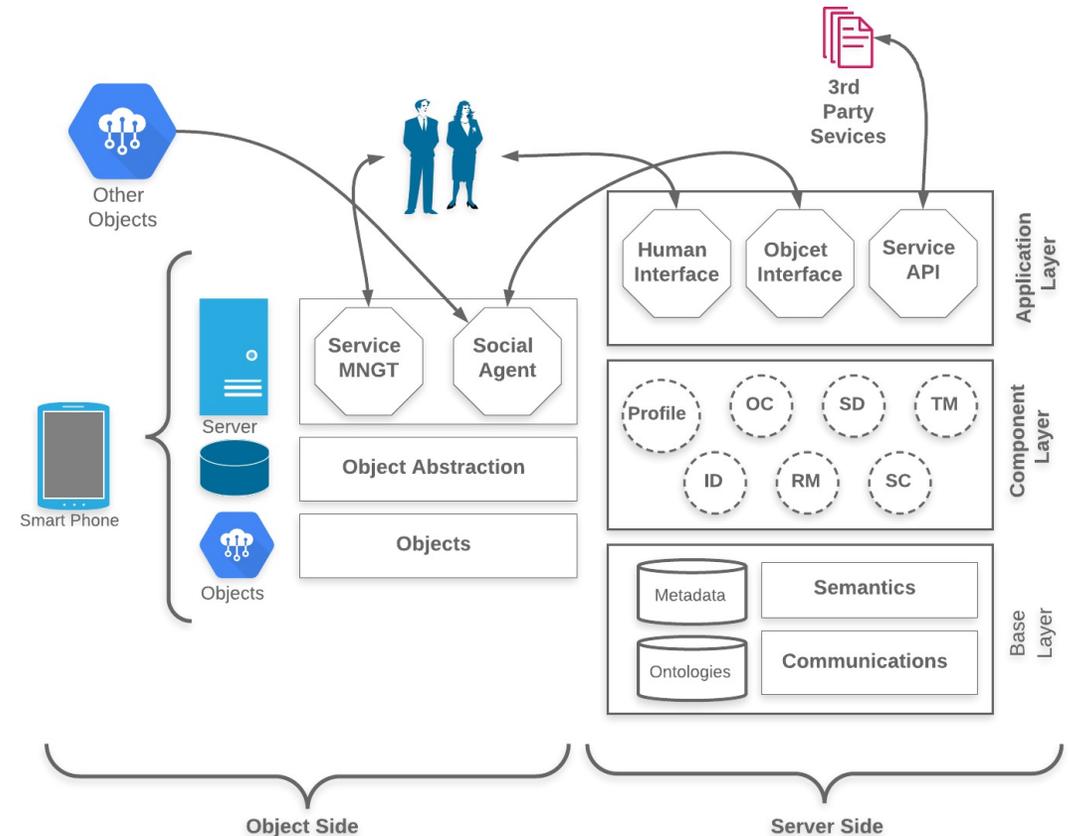
# Social IoT Architecture

- The server side consists of 3 main layers:
  - **Base Layer.** It includes a database that stores information about all the objects (attributes, metadata, relationships, semantic engines and communications between them).
  - **Component Layer.** Functionality for interaction between the objects
  - **Application Layer.** to interface and deliver services to the users.



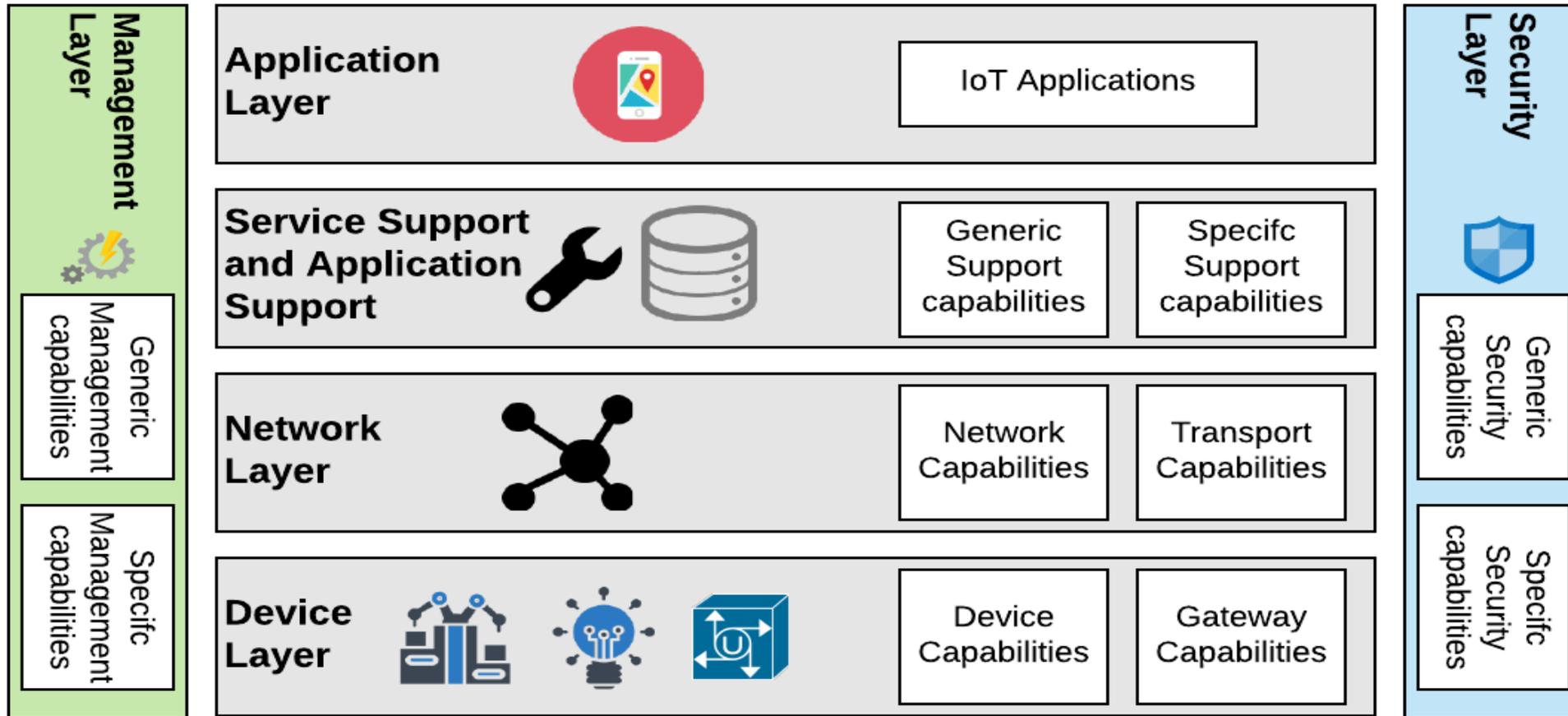
# Social IoT Architecture

- The object side there are again 3 layers:
  - **Object Layer.** Includes all the physical objects that can be discovered and reached through the communication interfaces:
  - **Object Abstraction Layer.** Common languages and procedures are used to harmonize the communication between different objects.
  - **Social Agent and Social Management Layer.** The social agent handles the communication between objects with regards to updating of the profiles and relationships and to discover or request services from the social network. The Service Manager provides the interface for humans to control the objects' behaviour.



# ITU-T Reference Model

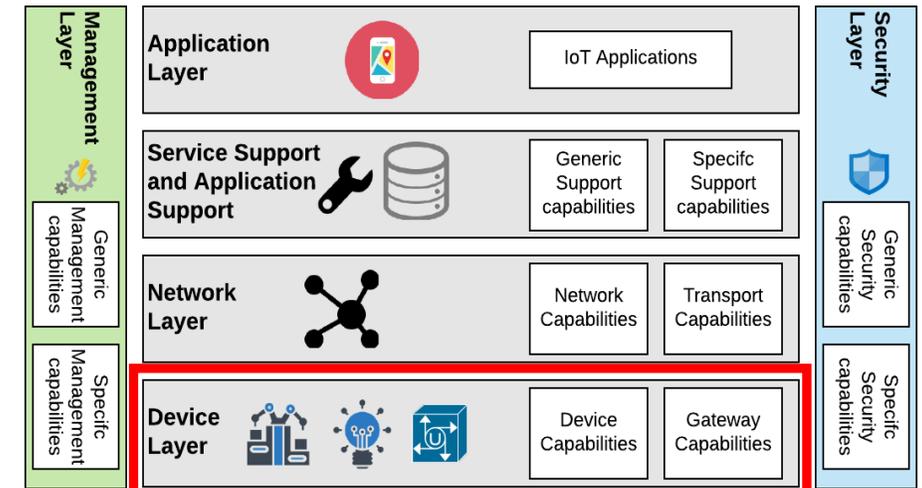
Consists of 4 Horizontal and 2 vertical Layers



# ITU-T Reference Model

## Device Layer

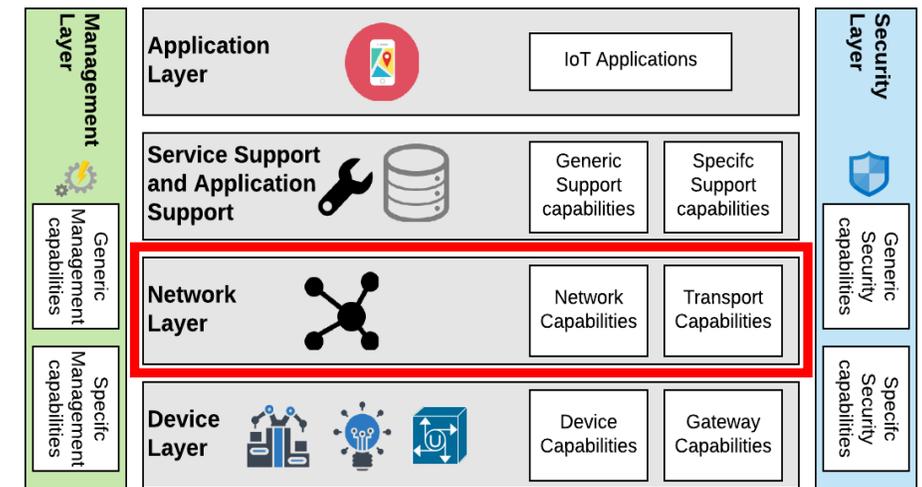
- Includes the Devices and the Gateway that provides connectivity to the WAN. For this reason, this Layer capabilities are grouped in two main categories:
- **Device Capabilities.** The most important capabilities related to the devices are:
  - Direct and Indirect interaction of the devices with the communication Network without using the Gateway in order to collect and upload or to receive information from the communication network.
  - Ad-hoc network capabilities so that devices can form ad-hoc networks in situations where there is increased need for scalability and quick deployment.
  - Sleeping and waking up: These mechanisms are provided to save energy.
- **Gateway Capabilities.** The most important capabilities of the gateway are:
  - Multi-Interface Support in order to support communication of devices using any wired or wireless technology (e.g. Zigbee, Bluetooth, WiFi etc.). At the network layer the gateway gets connectivity to the WAN (e.g. the Internet) using mobile technologies (2G, 3G, LTE, 5G), PSTN, DSL etc.
  - Protocol Conversion: To support communication at the device layer when the connected devices use different communication protocols (e.g. Zigbee devices connected with Bluetooth devices) and at the network layer when the connected devices use different technology to connect to the WAN (e.g. DSL and 5G).



# ITU-T Reference Model

## Network Layer

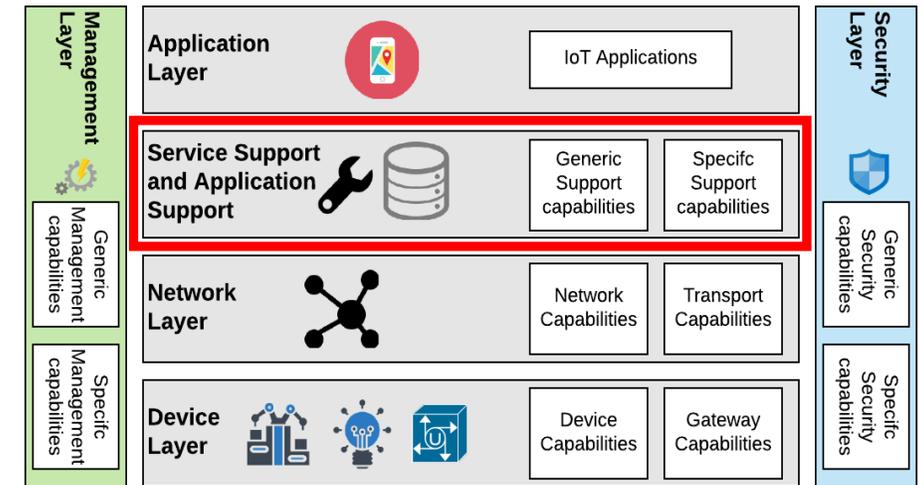
- Includes two sets of capabilities:
  - **Networking capabilities** to support connectivity to the network such as access and transport resource control functions, mobility management or authentication, authorization and accounting
  - **Transport capabilities** to provide connectivity for the transport of the IoT Service and application-specific data and the related control and management information.



# ITU-T Reference Model

## Service Support and Application Support Layer

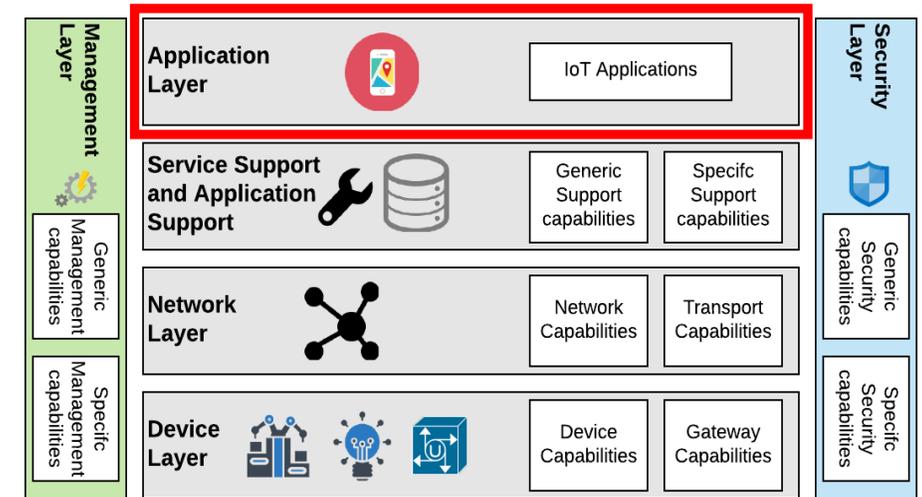
- Includes two sets of capabilities:
  - **Generic Support Capabilities:** Common capabilities usable by different IoT applications (e.g. data processing, data storage).
  - **Specific Support Capabilities** to support the requirements of diversified applications. In simple words they provide different support functions for different types of IoT applications.
  - *Generic Support Capabilities can be re-used in order to build specific Support Capabilities.*



# ITU-T Reference Model

## Application Layer

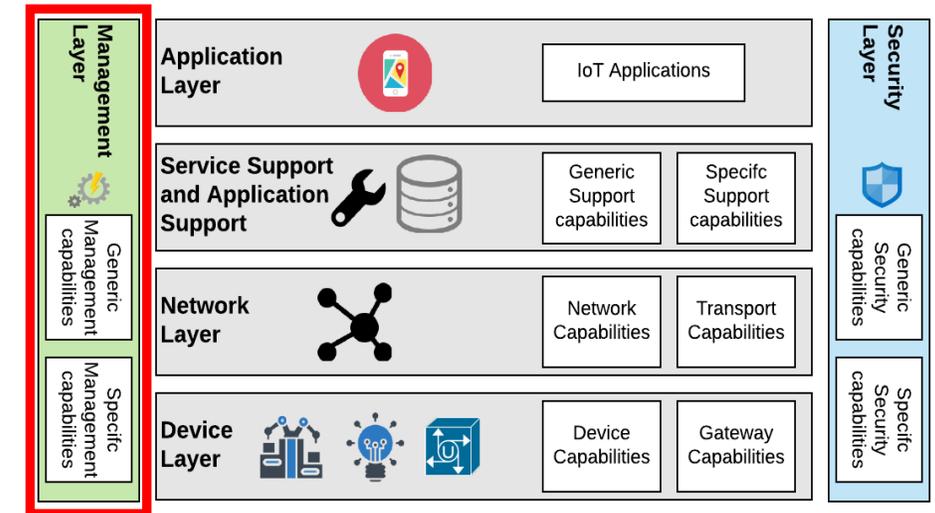
- Contains the IoT Applications.



# ITU-T Reference Model

## Management Layer

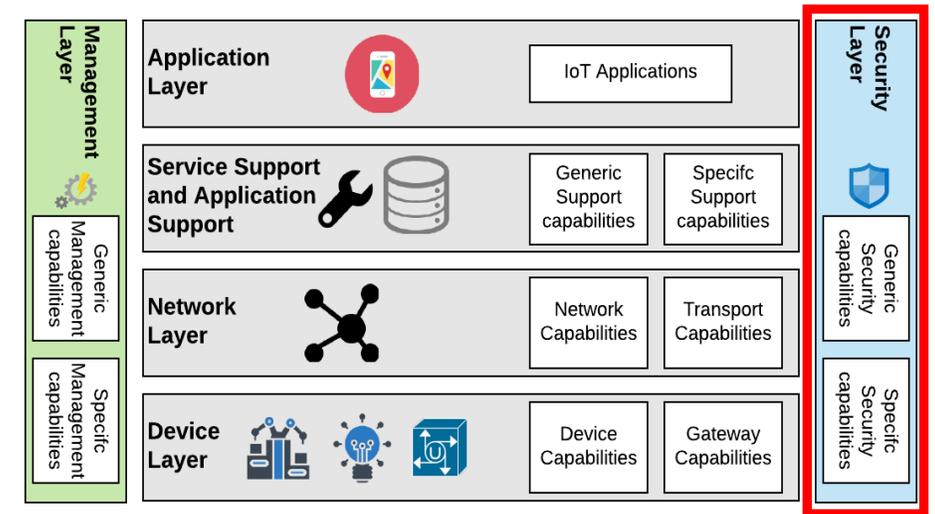
- Includes capabilities to support traditional networking management functions such as fault, configuration, accounting, performance and security (FCAPS) management. The management capabilities can be grouped in:
  - **Generic Management capabilities** like:
    - Device Management (e.g. remote device activation and deactivation, diagnostics, software and firmware updating, device working status management etc)
    - Local network topology management
    - Traffic and congestion management
  - **Specific Management capabilities** which depend on the application requirements



# ITU-T Reference Model

## Security Layer

- It includes two sets of capabilities:
- **Generic security capabilities** that do not depend on the application used:
  - At the Application Layer: authorization, authentication, confidentiality of application data, data integrity protection, privacy protection, security audit and anti-virus.
  - At the network Layer: authorization, authentication, confidentiality of use data and signalling data, signalling integrity protection
  - At the Device Layer: Authentication, Authorization, device integrity validation, access control, data confidentiality and integrity protection.
- **Specific security Capabilities** which depend on the application used (e.g. security of mobile payments, security of e-health data etc. )



## Section Outline

- Sensors/Actuators and Embedded Technology
- IoT Connectivity
- Data Management and IoT Analytics
- IoT Cloud
- User Interface



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

## Section 3

### IoT Devices and Components

---

# IoT Devices and Components

## Introduction

- An IoT system typically includes a large (and in some cases enormous) number of heterogenous devices with different capabilities and it becomes challenging how all these devices interoperate.
- Devices are categorized as data-carrying, data-capturing, sensing and actuating.
  - Data-capturing and data-carrying are responsible for the reading and/or writing of information from or to the physical things (e.g. temperature sensors, IR sensors, barcode readers etc.).
  - A general device on the other hand, has embedded processing and communication capabilities (e.g. a micro-controller) to perform more sophisticated functions or facilitate the development of stand-alone IoT systems without the need of connecting to the Wide Area network.
- Based on this categorizations, one can classify devices based on their processing power and their connectivity capabilities

# IoT Devices Classification

## Based on Processing Power

- **Devices with no processing capability:** In the context of IoT these are considered as passive devices, usually low-cost with no microcontrollers (e.g. RFID).
- **Devices with low processing capabilities:** Their processing capabilities are limited to the reading and writing data from or to sensors and actuators and sending this data to IoT applications, but they are not able to make decisions or run complex algorithm. They are typically low cost and usually embed a very low-power and low-cost microcontroller. (e.g. a smart light or a door sensor.)
- **Devices with high procession capabilities:** They have enough processing power to enable them making decisions and running complex algorithms. They are typically high cost as they employ a powerful microcontroller. (e.g. a smart cooling system, or a smart thermostat)

# IoT Devices Classification

## Based on Connectivity

- **Devices with low connectivity:** This kind of devices do not connect directly to the communication network to transfer the data but instead they rely on additional elements (e.g. gateway) to perform communications tasks (e.g. protocol translation or internet connectivity).
- **Devices with High connectivity:** They have the hardware and ability to directly connect to the network to transfer the data.

# IoT Devices Classification

- The ITU-T Recommendation Y.4460 defines the architecture models devices with different capabilities. Specifically, it proposes models for devices with:
  - Low Processing and Low Connectivity (LPLC)
  - Low Processing and High Connectivity (LPHC)
  - High Processing and High Connectivity (HPHC)

# The Main Components of an IoT System

- Sensors/Actuators and Embedded Technology
- Connectivity
- Data Management and IoT Analytics
- IoT Cloud
- User Interface

# Sensors/Actuators and Embedded Technology

- Considered the frontend of any IoT application or system.
- **Sensors** facilitate the concept of context awareness so that knowledge about the environment is collected and uploaded for further processing to the attached communication network.
- **Actuators**, receive instructions from the communication network and perform actions onto the environment they reside.

# Sensors

## General

- A sensor is a device that is used to measure a physical quantity by converting it into a signal that can be read by the system.
- In IoT physical quantities from the environment (e.g. temperature, humidity, inertia etc.) are measured then they are converted into electronic signals which are then digitized to be sent to the communication network.
- Sensors typically include transducers which, by definition can convert on form of energy to another.
- Based on the application there are many possible sensors that can be used in an IoT system (temperature sensors, RFID, light sensors, electromagnetic sensors etc.).



[This Photo](#) by Unknown Author is licensed under [CC BY-SA](#)

# Sensors

## Classification Criteria

- **Power supply requirements:** Passive Sensors or self-generating, directly generate an electrical signal in response to an external stimulus without the need for an external power supply (e.g. thermocouple or piezoelectric sensors). Active sensors require external power supply or an excitation signal for their operation and in this case the output signal power comes from the power supply (e.g. Infrared or Sonar sensors).
- **Nature of the Output signal:** Sensors can be either analogue or digital. Analogue sensors generate signals that are continuous in both their magnitude and temporal or spatial content (e.g. temperature, displacement, light etc.) Digital sensors are ones that generate signals that are discrete in time and amplitude (e.g. shaft encoders, switches etc).
- **Operational Mode:** Deflection mode sensors generate a response that is a deflection or a deviation from the initial condition of the instrument and this deflection is proportional to the measurand of interest (e.g. pressure sensor). A Null mode sensor exerts an influence on the measured system so as to oppose the effect of the measurand. The influence and measurand are balanced (typically through feedback) until they are equal but opposite in value, yielding a null measurement. Null mode sensors can produce very accurate measurements but are not as fast as deflection instruments. (e.g. Wheatstone bridge sensors).
- **Measurand:** Sensors depending on the quantity they measure (e.g. Mechanical, thermal, magnetic, radiant, etc.)
- **Physical Measurement Variable:** Depending on whether the sensors rely on the variation of resistance, capacitance or inductances they can be classified as resistive, capacitive or inductive.

# Sensors

## Selection Criteria

- According to the application as well as accuracy and precision requirements, sensors should be selected while considering the following aspects:
  - Accuracy of the input Readings
  - Reliability and Repeatability of input
  - The conditions of the environment the sensors will be placed in
  - Cost and power consumption

# Actuators

## General

- Actuators are devices that can take an effect on the environment they belong by converting electrical signals into different actions or in different forms of energy.
- Examples include lights, displays, motors, robotic arms, heating/cooling elements etc. Motion-based actuators are typically categorized into electrical, hydraulic or pneumatic actuators.
- Electrical actuators convert the electric signals into some form of rotation (e.g. motor) or motion, hydraulic ones facilitate mechanical motion using fluids whereas pneumatic actuators use the pressure of compressed air.
- In the typical example of a smart home automation system we can find actuators that lock/unlock doors, switch on/off the lights, heat up to increase the temperature, etc.

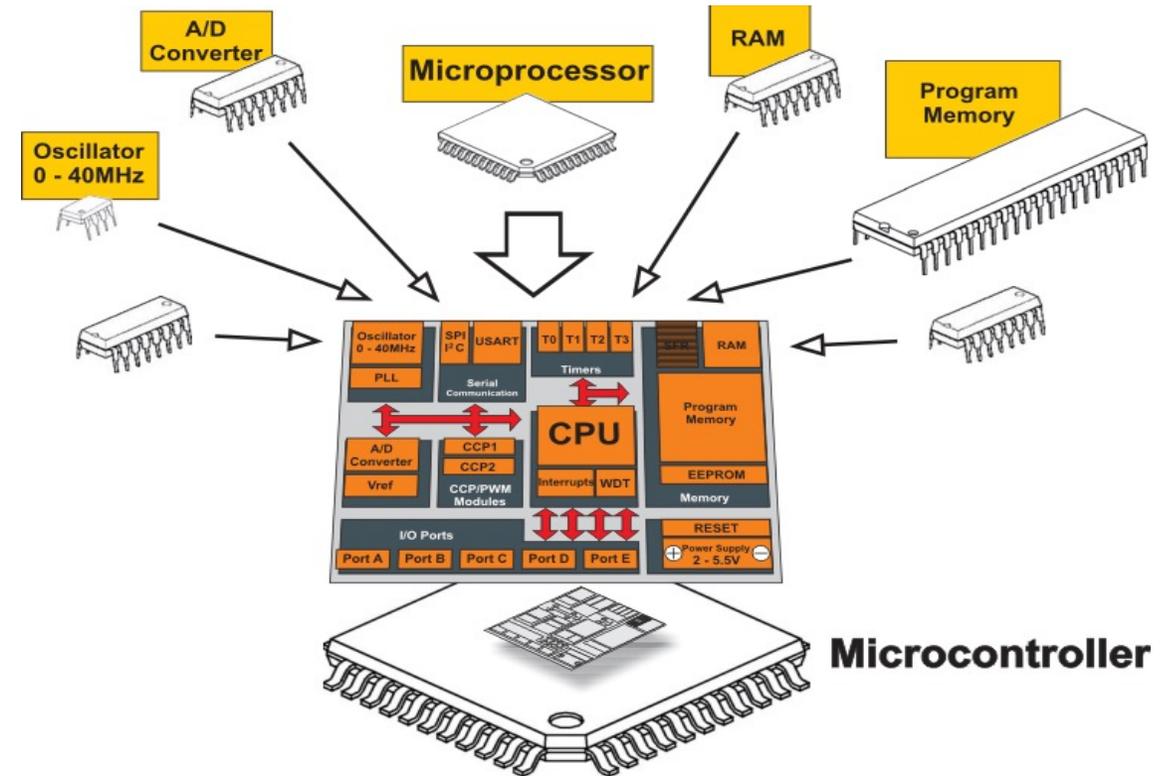


[This Photo](#) by Unknown Author is licensed under [CC BY-SA](#)

# Microcontrollers and Embedded Systems

## What is a microprocessor/microcontroller

- A sensor is a device that turns the received physical conditions or states into signals (analogue or digital)
- An actuator is the device that turns the digital signals into some sort of physical effect,
- the **microprocessor** is considered to be the computing systems which sits in the middle and processes and/or generates the digital signals.
- A microcontroller has a central processing unit (CPU), a fixed amount of memory (RAM and ROM) as well as other input/output ports and peripherals all embedded onto a single chip.



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

# Microcontroller Characteristics

## Choosing a microcontroller

- “one-size-fits-all” approach cannot be adopted. Various characteristics need to be taken into consideration when choosing the microcontroller:
  - **Bits:** Microcontrollers come with different capabilities with regards to the number of bits they can support. This affects their processing speed. Typical sizes are 8-bit, 16-bit, 32-bit and 64-bit.
  - **Memory:** Random Access Memory (RAM) is a fast-access memory that does not keep the data when there is no power on the device. Microcontrollers are embedded with this kind of memory in order to quickly perform various actions. They come in different sizes but increasing the size of the memory although it improves the processing capability it increases the cost.
  - **Flash or the ROM:** It is the microcontrollers memory which retains the data stored in it when power is off. It is not as big as the RAM but it is required in order to support offline storage.
  - **General-Purpose Input Output (GPIO) pins:** These are the points of connection for the sensors and the actuators. The number of pins can vary from a few tens up to hundreds, depending on the size and cost of the microcontroller.
  - **Connectivity:** The ability of the microcontroller to establish connections to the network or the Internet. This can be done via Wi-Fi, Bluetooth, Wired Ethernet or any other communication technology.
  - **Power consumption:** This is an important aspect as it will define how many active sensors and actuators the microcontroller will be able to power up and control especially when the microcontroller is powered but from alternative sources (e.g. solar). It is important IoT devices to be energy efficient so that they can perform tasks for a long time without the need of regularly powering them up.
  - **Development Tools and Community:** It very useful for microcontrollers to come with development tools and the related documentation to facilitate their integration onto IoT solution. Having a community or forums working on different types of microcontroller makes the integrators/developers job much easier in finding information related to their development.
- Some popular IoT Microcontrollers are Arduino, ARM, Raspberry Pi and many others.

# Connectivity

- Connectivity is a key ingredient of an IoT System since,
- Remember that the only mandatory capability of an IoT Device is communication. Any device attached to an IoT platform should be able to send or receive data from the attached network.
- There are various challenges that need to be considered and dealing with IoT communication:
  - **Identification and Addressing:** As there might be a very large number of IoT devices attached to the communication network there should exist efficient mechanisms and protocols to identify these devices through unique addresses. Due to running out of addresses in the IPv4 protocol, IPv6 becomes a necessity in IoT.
  - **Low Power Communication:** IoT devices are typically low power devices with many power restrictions. Therefore, it needs to be ensured that the communication technology does not consume much of the available power on these devices.
  - **Efficient Routing protocols** with low memory requirements
  - **High-Speed Communication**
  - **Mobility**

# Connectivity

## Smart Gateway

- Data from sensors to the communication network or data from the communication network to the actuators in many cases have to go through gateways especially in cases where multiple networks need to be traversed.
- Gateways are devices that make network protocol translations to ensure seamless communication between the many (typically heterogeneous) IoT devices.
- That said, gateways are an integral part and have central and crucial role in an IoT system being responsible for the easy management of data traffic.
- In some cases, gateways offer security by protecting the system from unauthorized access and malicious attacks.
- Moreover, act as pre-processors for the data collected from the sensors before forwarding them to the cloud.
- In this context, there exist "Smart" or "Intelligent" gateways that analyse the data to either infer new knowledge or compress the data by forward only the important and relevant information to the cloud.

# Connectivity

## Networks, Mobile Technologies and Protocols

- In IoT, connection to the Internet is typically and usually achieved using the Internet Protocol (IP) despite the fact the IP protocol stack is power- and memory-demanding for the connected devices.
- For this reason, it is also possible for devices to connect to the local network using non-IP technologies like RFID, Bluetooth, NFC, etc. however these technologies are limited in range.
  - These low-range technologies are used for personal area networking (PAN) and are quite popular in IoT applications such as wearables.
- For Local Area Networking (LAN) IP-compatible technologies should be used however the IP-protocol needs to be modified to support low power communications.
  - One of these protocols is 6LoWPAN which incorporates IPv6 with lower power requirements. Other networking technologies that can be used in IoT include IEEE802.15.4, RFID, LTE, 5G, 802.11 standards, Z-wave etc.
- More information about IoT connectivity in Lectures 7 and 8

# Data Management and IoT Analytics

## Data Management

- IoT is highly associated with a colossal number of data that gets communicated between the IoT components
  - there is raw sensed data which is being collected by the sensors, pushed to the gateways for preprocessing and maybe inferring of new data and then uploading of this data to the cloud for storage or further processing.
  - In the reverse direction analysed data leads to smart decisions which are forwarded back to the actuators for execution.
- This requires a system that is capable of storing, processing and analysing a very large amount of data; hence data management and data analytics are critical components of an IoT system.

# Data Management and IoT Analytics

## IoT Analytics

- IoT Analytics is used to make sense of the vast amounts of collected data.
  - For instance, to infer the walking patterns, or most-visited shops of shopping mall visitors whose position is anonymously tracked.
  - Another example could be the estimation of the likelihood of a car accident by monitoring the driving behaviour of cars in a smart transport environment.
  - Once such situations are identified, an immediate decision needs to be taken by the system and a specific action needs to be executed or a warning needs to be issued to prevent undesirable scenarios.
- In simple words, IoT analytics are concerned with the conversion of the raw sensor-collected data into useful insights (further knowledge inference) which are analysed to lead into smart and useful decisions.
- Data analysis has storage and computation requirements which in many cases cannot be accommodated in the sensors or even the smart gateways therefore they need to be provisioned in the cloud.
- More information about Data Management and Analytics in Lectures 9 and 10

# IoT Cloud

- It can be considered as a smart high-performance entity which combines the various IoT components together with high data-handling, storage and decision-making capabilities.
- Many IoT applications these days have very low latency requirements in the range of milliseconds (e.g. smart health, smart transport etc.) therefore the IoT Cloud should be able to process the colossal amount of data from multiple source extremely fast.
- The Cloud is the brain of the IoT ecosystem and typically responsible for processing, analysing, decision-making and storing the data.
- Nevertheless, the cloud is not a mandatory component of an IoT system since Fog or Computing allows processing and storage to happen in a distributed way. T
- he Cloud solution is preferred when massive scalability and decreased operational cost are required whereas the Edge computing solution is the preferred option when large amount of data processing and storage are required on-premises.

# User Interface

- The user interface is the physical and visible part of the IoT system to the user.
- It provides an interactive way through which the user can get access to the data, receive alarms, perform actions, give instructions, set preferences etc.
- It is of high importance to develop an interface which is friendly to the user and does not require extra effort to perform these interactions with the IoT application.
- The interface could be simply an application implemented on a smartphone or table or it could be a direct interaction with the “Things” (e.g. in Amazon Alexa users interact directly with the devices).





# Thank You

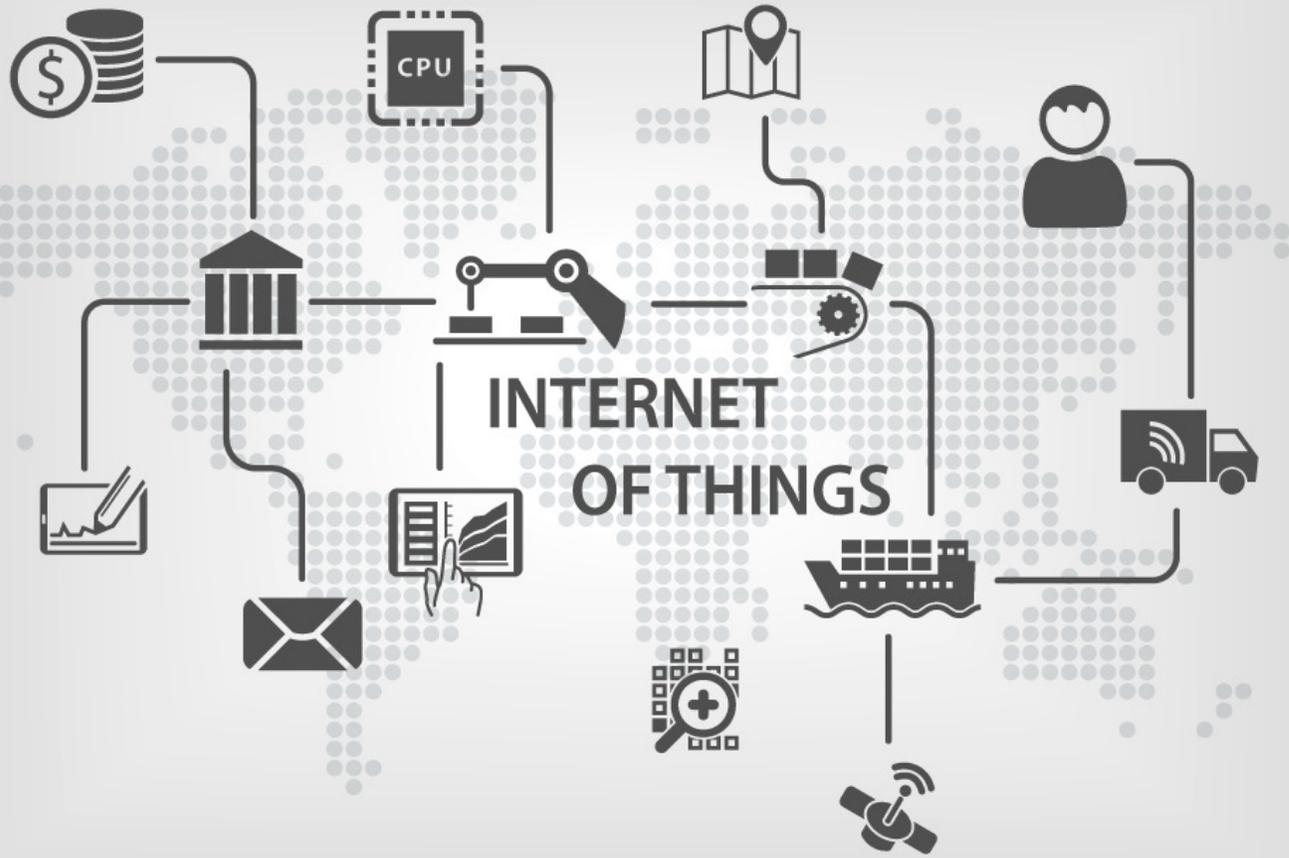
 Dr Marios Raspopoulos

 +357 24694070

 [mraspoulos@uclan.ac.uk](mailto:mraspoulos@uclan.ac.uk)

 <http://www.uclancyprus.ac.uk/>

**PROUDER**  
Introducing Recent Electrical Engineering  
Developments Into Undergraduate Curriculum



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

*This publication was produced with the financial support of the European Union. Its contents are the sole responsibility of the partners of IREEDER project and do not necessarily reflect the views of the European Union*



Co-funded by the  
Erasmus+ Programme  
of the European Union

# Introduction to the Internet of Things

## Lecture 5: IoT architecture and components (2 of 2)

---

**Dr. Omar R. Daoud**

**Dr. Mohammed Bani Younis**

**Dr. Saleh Saraireh**

**Eng. Rasha Gh. Freehat**

Introducing Recent Electrical Engineering  
Developments into undergraduate curriculum

IREEDER

# This week's topics...

- Cyber-Physical System (CPS).
  - Introduction
  - The Rise of CPS
- Basic Concepts of IoT
  - Storage and Central Processing Units
  - Data Movement
  - Input and Output SPI/I2C
  - Accelerators
  - Peripherals

# This week's topics...

- Embedded Memory.
  - Embedded Systems Memory Types
- Causes and Implications of Memory.
  - Compute-Constrained Devices
  - Constrained Node, Constrain Network and Constrained-Node Network
  - The need for Management of constrains devices and constrained devises restrictions
  - Applications for Constrained Devices

# Section Outline

- Introduction
- The Rise of CPS



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

## Section 1

Cyber-Physical System (CPS).

# Introduction

## ○ ***Cyber-Physical System (CPS):***

- CPS stands for the direct interaction between the physical world and the computers.
- The CPS concept was initially supported by the US National Science Foundation (NSF).
- It concerns on the control process of monitoring, sensing, or managing different physical environments that consists of several distributed computing devices.
- It requires expertise and skills from building algorithms point of view, modelling physical environments, and systems actuations, integration and communications.

# Introduction

## ○ ***Cyber-Physical System (CPS):***

- It is considered as the key infrastructure for the modern societies.
- CPS definition is emphasized to be Cyber-Physical Systems of Systems (CPSoS)
- CPS is considered as a transforming interface of interactions between the users and the engineered systems.
- It integrates the processes such as sensing, controlling and networking into physical infrastructure.
- Based on the internet connection for such interactions.

# The Rise of CPS

- The CPS concept began since early 1980s in Carnegie Mellon University.
  - Coke machine to the internet which was able to label it's stoked with either "cold" or not.
- This vision was the seed of developing the concept of the internet of things (IoT)
  - Some scenarios were started to making use of the radio-frequency identification (RFID)
  - The RFID is considered as a small tag with a traceable chip, which is could be traced, controlled or/and managed through the internet.

# The Rise of CPS

- Wireless sensor networks (WSNs) have these days increasing prominent:
  - Due to that a large number of heterogeneous tools and devices are interconnect through the internet; such as dozens of computerized devices, or even hundreds of small sensor nodes.
  - Therefore, the data of the physical world is easily collected, monitored, managed and controlled by the means of wireless communication.
  - These sensing elements permit the real-world data collection and the digital-form handling process, which can be easily distributed making use of the Ad-hoc network distribution.
  - Nowadays, these sensors can be deployed on humans in addition to the impeded sensors in the carried smart phones and devices.

# The Rise of CPS

## CPS Examples:

- Human Personal Health:
  - Integration through the human vital signs could be monitored and managed
    - *to not only suggest the best places and routes for the use but also make an interoperable personalized medical devices or robotic surgeries.*
- Transportation:
  - Either from cruising control point of view or the securely communications with other smart elements on the roads.
    - *This will reduce the road delays besides inspect the danger and disaster zones and many other crucial issues*
- Sustainability:
  - It could be used in many issues such as detecting and deterring fires, combatting the oil spills underwater and many others.
    - *Thus, the used sensors, actuators or any other capabilities will be integrated in order to attain the desired results and targets.*

# The Rise of CPS

## The Research Needs in CPS

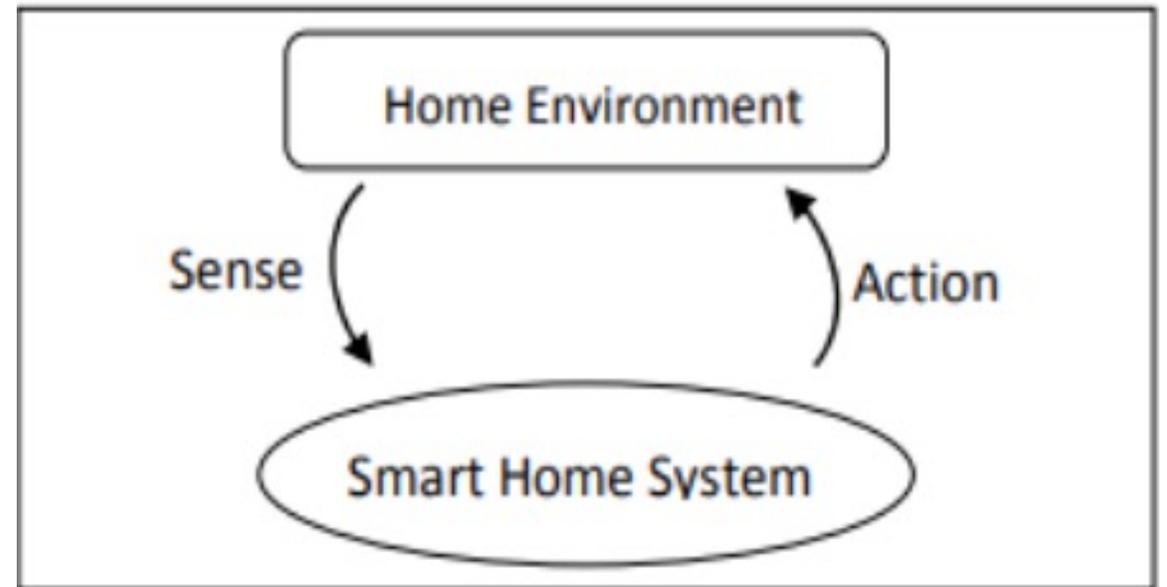
- Abstraction and Architectures:
  - Both of abstraction and architectures should has innovative approaches to offer:
    - *A controlling process with a unified integration.*
    - *A rapid design for communication.*
    - *An accurate deployment for computation.*
- Distributed Computation and Networked Control:
  - Several challenges appear when taking the design of a networked control into account such as the failure issues, the processing time delay, the distribution schemes and the reconfigurations
- Verification and Validation:
  - Due to that the CPS infrastructure lacks of trustworthiness, its components and structure must be developed and deployed beyond the existing technologies and be fully integrated in terms of
    - *Dependability,*
    - *Configurability,*
    - *Certification.*

# Smart Home Systems as a CPS Case study

- Smart home system is a very good example for the CPS
  - It gathers different components (home appliances) each of which has its own functionality in one centric system.
  - Some issues should be taken into consideration in this case :
    - *These components were prototyped using embedded boards,*
    - *Some of them is equipped with a touch screen,*
    - *Some of them have a programmed user interface,*
    - *Smart components (devices) have been programmed based on the device profile for web services (DPWS) stack; i.e. they have the smartness behaviour or device operations.*

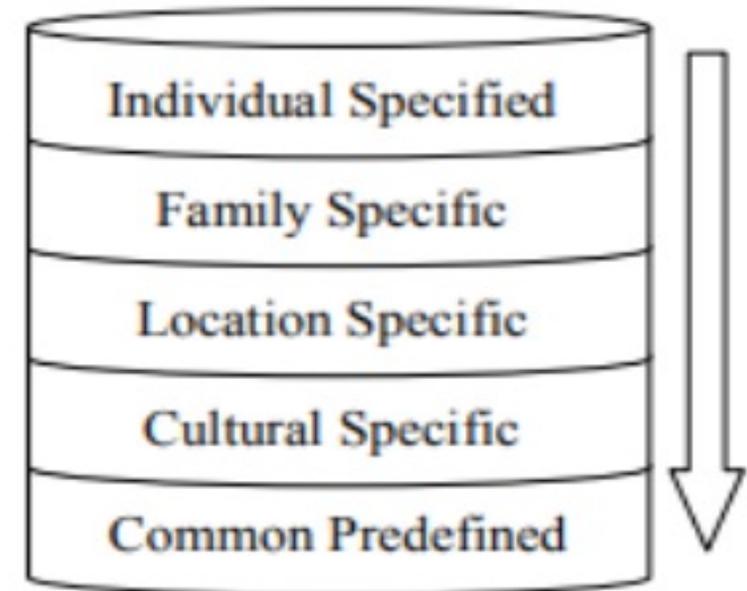
# Smart Home Systems as a CPS Case study

- The home system has been designed from the CPS point of view
  - The user himself is part of the system and his behaviour will describe the context-aware service.
- The sensors are the main parts of this system;
  - any action will be sensed and cause a reaction from the system.
- A decision making process will start as a reaction of the environment changing.
  - These reactions will be drawn a saved context-logic information stack, which has an individualized data for any system



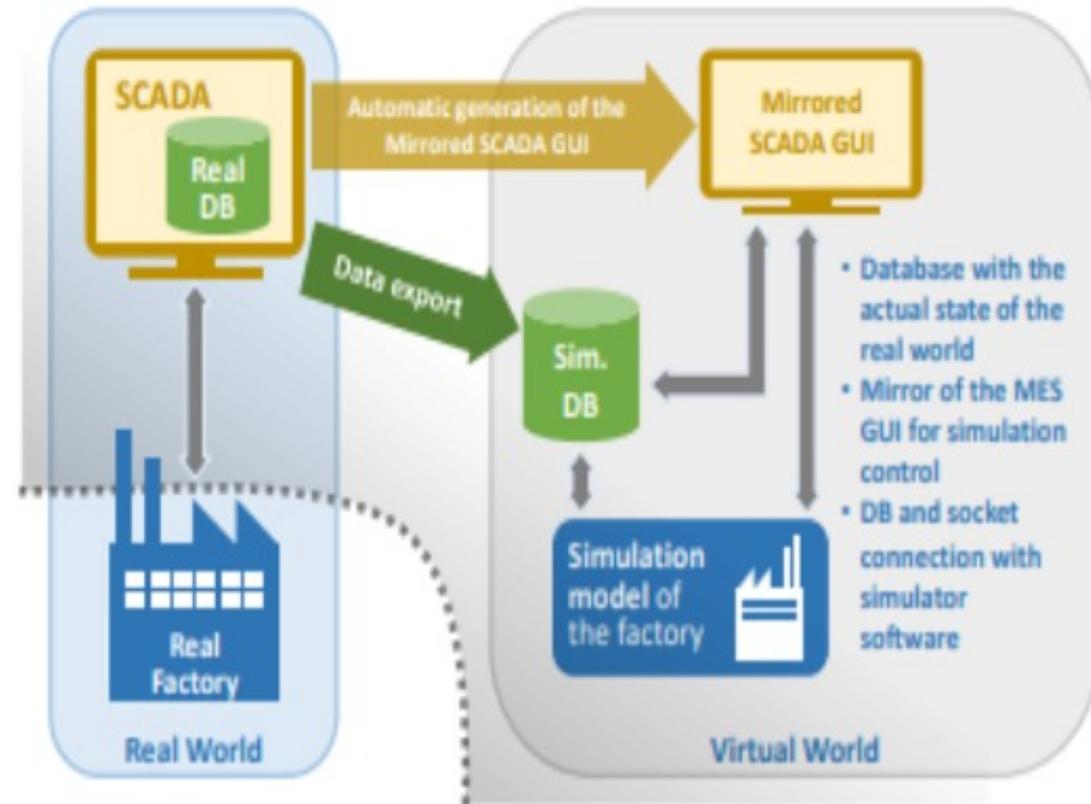
# Smart Home Systems as a CPS Case study

- The logic stack consists of layers
  - each of which can be considered as a specific case.
- This means that according to the preferred data saved in the stack,
  - It should give/propose the appropriate solution(s) intelligently.
- Furthermore, if the collected data is not enough to draw the solution,
  - The system should go downgrades to the next layer.
- Therefore, the lowest layer is the default layer with the factory settings.



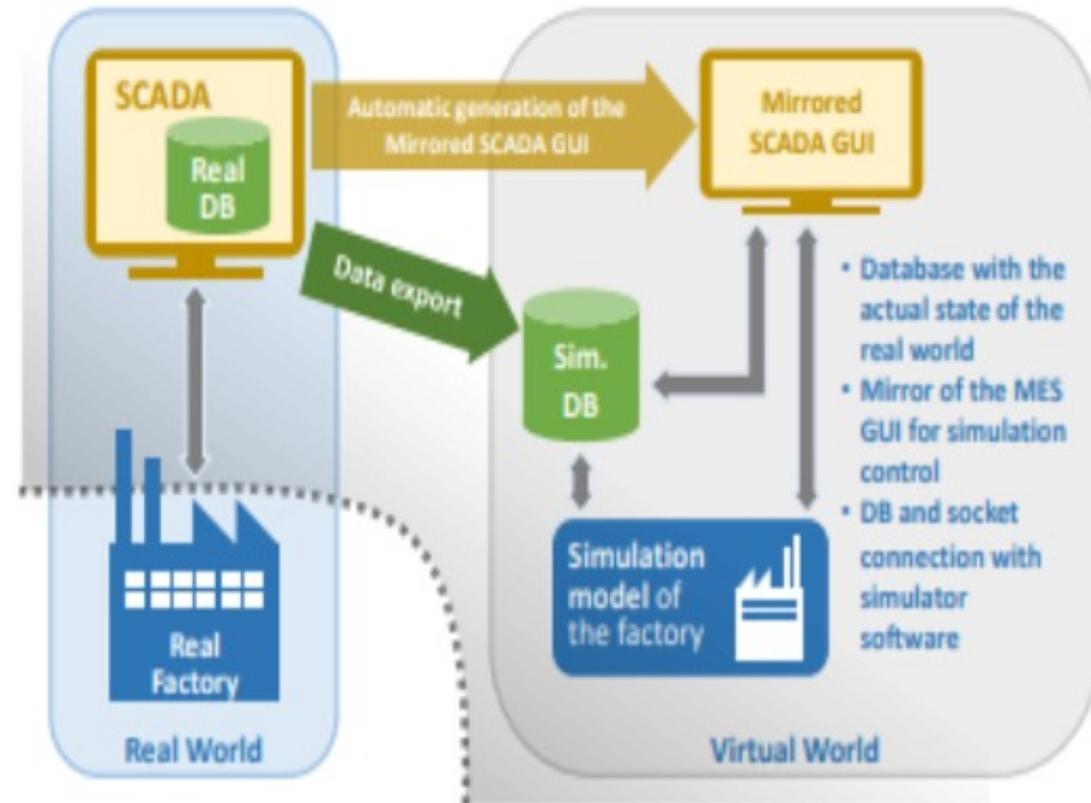
# Smart Home Systems as a CPS Case study

- It depicts the mapping process between the real system and the controllers.
- This is in addition to the synchronization process in the virtual environment.
- In this case, executing the scenarios and experiments will be at the same parameters as of the real system.
- Also, the results will be evaluated using the same performance indicators



# Smart Home Systems as a CPS Case study

- In order to speed up the modelling process, the automated model technique should be used in the building process.
- This technique extracted the control codes from the low-level basis into the database of the model definition, which will be attained using a tailored grammar interpreter.
- As shown in the Figure, the database is used to build up the simulation model automatically by providing the inputs for the methods; i.e. the stored data in the low level controllers.





# Basic concepts of IoT

## Introduction

- The structure of the network of things (IoT) is a system of related figure gadgets, mechanical machines, advanced machines, objects, creatures or people that have been given unique identifiers (UIDs) and the ability to transfer information in the system without requiring human-human or human-computer operation.
- IoT structures nowadays are becoming part of different aspects of our lives thanks to their applications, such as smart homes, medical services, state control and surveillance, smart city communities and brilliant transportation structures.
- With the development of the IoT framework, a huge system of systems linking these obligatory gadgets (sensors, actuators, machines, etc.), and with this association, numerous limitations grow, for example, gracefully limited strength and security that develops with any information.

# Basic concepts of IoT

## Storage and Central Processing Units

- When the sensor receives physical conditions from the environment and converts them into signals, and the actuator converts the signals into physical actions, the microprocessor is the processing system, which generates digital signals.
- The MCU has a central processing unit (CPU), a fixed amount of memory (RAM and ROM), all of which are embedded in the chip.
  - MCU acts as an intermediate device to help interconnect and control endpoint devices.

# Basic concepts of IoT

## Storage and Central Processing Units

- To select an MCU for an IoT application or system design, various characteristics need to be considered, such as:
  - **Bits:** MCUs have different functions in terms of the number of supported bits, which will affect their processing speed.
  - **Memories:** Random Access Memory (RAM) is a high-speed memory that does not store information when the gadget is not forced to act.
  - **Flash or ROM:** When the power is turned off, it is the microcontroller memory that retains the data stored in it. It is not as large as RAM, but it is necessary to support offline storage.
  - **Development Tools and Community:** It extremely valuable for MCUs to accompany advancement instruments and the related documentation to encourage their reconciliation onto IoT arrangement.

# Basic concepts of IoT

## Storage and Central Processing Units

- To select an MCU for an IoT application or system design, various characteristics need to be considered, such as:
  - **General-Purpose Input Output (GPIO) pins:** These are the connection points of sensors and actuators.
  - **Connectivity:** The capacity of the microcontroller to build up an association with the system or the Internet.
  - **Power consumption:** This is a significant viewpoint as it will characterize what number of dynamic sensors and actuators the MCU will have the option to control up and control particularly when the MCU is powered however from elective sources (for example sunlight based, players).

# Basic concepts of IoT

## Storage and Central Processing Units

- Some popular IoT microcontrollers are Arduino, ARM, Raspberry Pi, Udo Neo, LightBlue Bean, TinyDuino, etc
- They run the operating system within the following three main options:
  - **Bare Metal** The main advantage it is the cost-effective and efficient.
  - **Real-Time Operating System (RTOS)** An RTOS system provides exact time operations guarantees.
  - **Linux** Its main advantage it is much easier to program and connect to the Internet, but it does not provide any timing guarantees.

# Data Movement

- Any device attached to an IoT platform should be able to send or receive data from the attached network.
- There are various challenges that need to be considered and dealing with IoT communication:
  - **Identification and Addressing:** There is a very large number of IoT devices attached to the communication network so efficient mechanisms and protocols must exist for identifying these devices through unique identifiers (UIDs). Given the running out of addresses in the IPv4 protocol, IPv6 becomes a necessity in IoT.
  - **Low Power Communication:** IoT devices are typically low power devices with many power restrictions. Therefore, it needs to be ensured that the communication technology does not consume much of the available power on these devices.
  - **Efficient Routing protocols** with low memory requirements
  - **High-Speed Communication**

# Data Movement

- Compelled gadgets utilize a fluctuating transmission conventions, (for example, LPWAN, Wi-Fi, Bluetooth and Zigbee, GPRS and numerous others),
- Gateways talk with these gadgets over a differing conventions and afterward make an interpretation of information to a standard convention, (for example, MQTT, HTTP and CoAP).
- The gateway engineering comprises of three layers:
  - The detecting layers containing the M2M gadgets,
  - The passage layer offering the types of assistance of system associations,
  - The application layer offering types of assistance to clients.

# Input and Output SPI/I2C

- Conversing with equipment by associate sensors, actuators, and more to make IoT framework bursting at the seams with movement, sensation, sound, and so on to speak with other equipment requires sequential protocol like I2C or SPI.
- These protocols are the basic language that chip and extra sheets talk.
- Two normal sequential protocols, I2C and SPI will present:
  - **Serial Peripheral Interface (SPI) Protocol**
  - **Inter Integrated Circuit (I2C) Protocol**

# Input and Output SPI/I2C

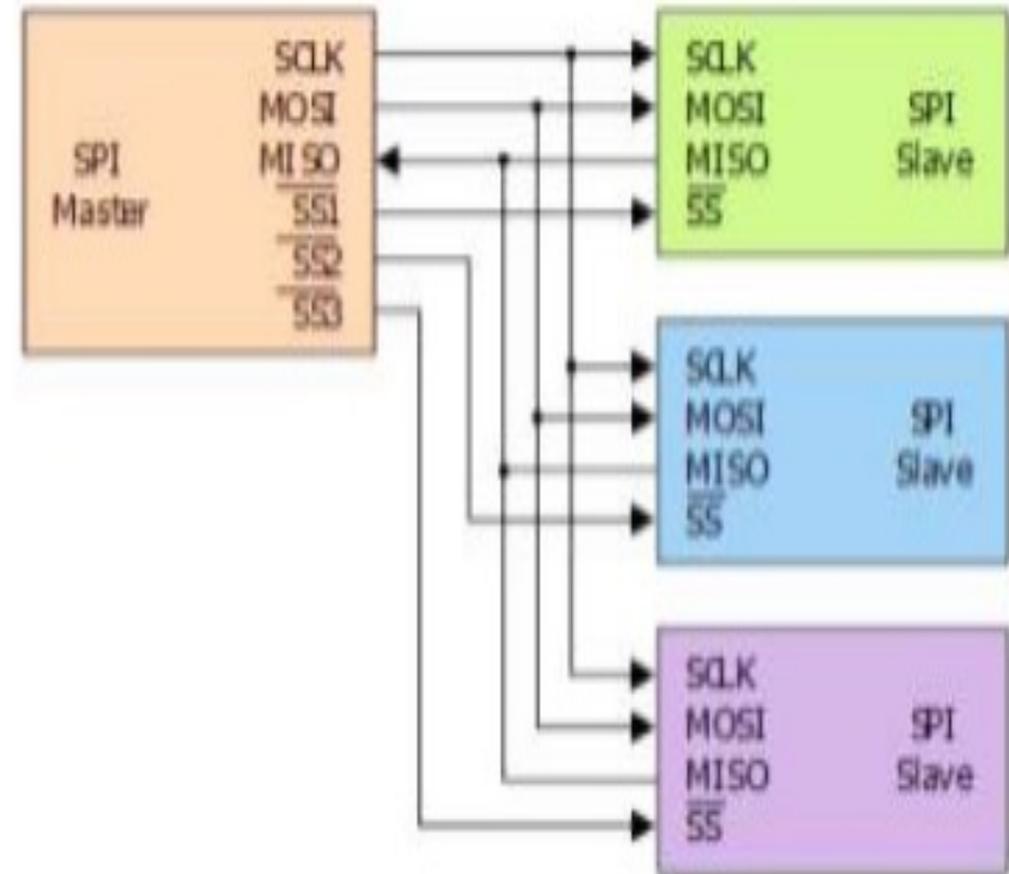
## Serial Peripheral Interface (SPI) Protocol

- It is a bidirectional sequential protocol for two gadgets to send and get information, utilized in short-separation correspondence, essentially in installed frameworks
- It is known as a four-wire sequential transport, differing with three-, two-, and one-wire sequential transports
- SPI is a duplex coordinated sequential information interchanges interface standard which work in a full duplex mode where information can be send and gotten all the while

# Input and Output SPI/I2C

## Serial Peripheral Interface (SPI) Protocol

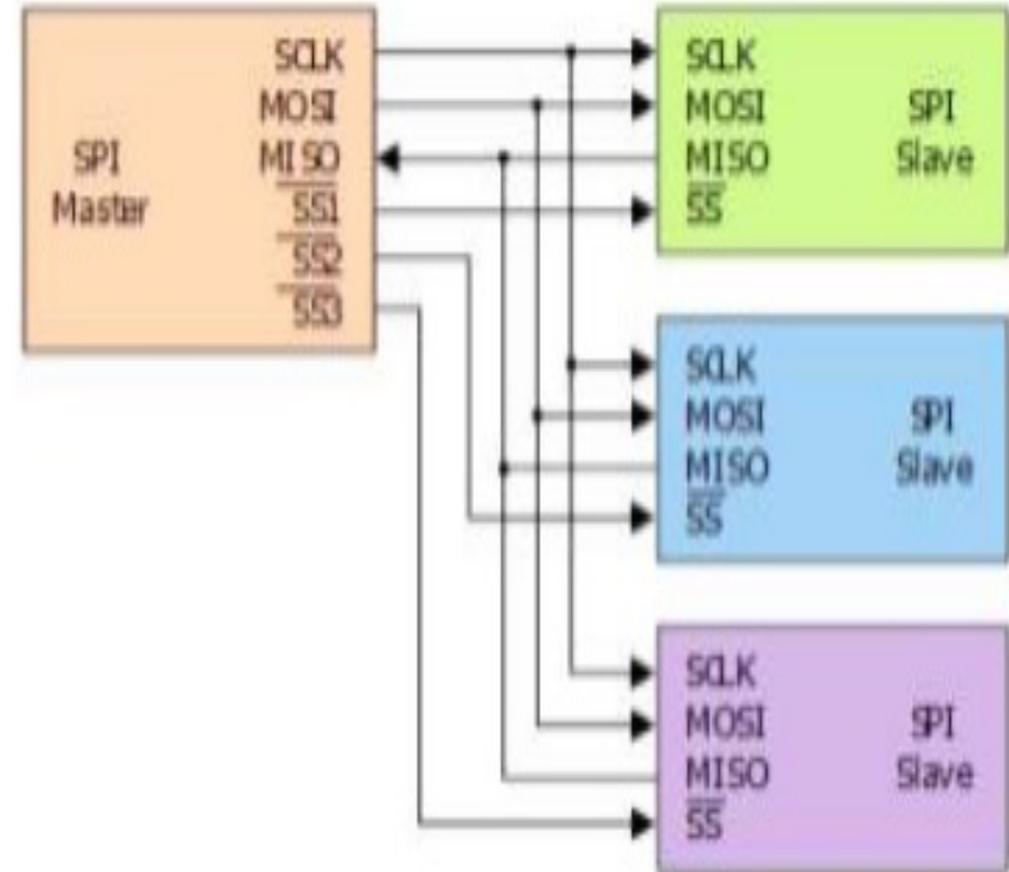
- The Pins: Interface and Function: Coming up next are the SPI transport rationale signals, which in has four lines associate the master to slave, and their function is as follow:
  - **Serial Clock (SCLK):** The output from the master and gives the planning of information trade.
  - **Master Output, Slave Input (MOSI/SIMO):** The output from the master and used to move information from the master to the slave.



# Input and Output SPI/I2C

## Serial Peripheral Interface (SPI) Protocol

- The Pins: Interface and Function: Coming up next are the SPI transport rationale signals, which in has four lines associate the master to slave, and their function is as follow:
  - **Master Input, Slave Output (MISO/SOMI):** The output from the slave and used to move information from the slave to the master.
  - **Slave Select (SS):** The output from the master which is active low. It is utilized by the master in order to address and enact a specific slave so as to begin a correspondence session.



# Input and Output SPI/I2C

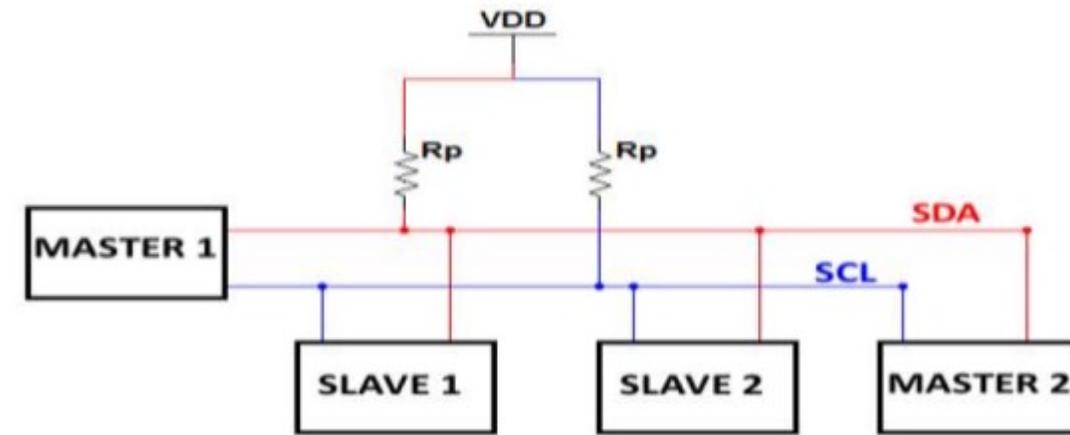
## Inter Integrated Circuit (I2C) Protocol

- It is a sequential interchanges protocol utilized with implanted frameworks and sensors.
- I2C is bidirectional two-wire simultaneous sequential transport and require just two wires to communicate data between gadgets associated with the transport.
- This protocol is helpful for frameworks that require a wide range of parts cooperating (eg. low-speed gadgets like microcontrollers, EEPROMs, A/D and D/A converters sensors, pin, extensions and drivers).
- I2C is slower than the SPI; speed of I2C is additionally needy by information speed, wire quality and outer commotion.

# Input and Output SPI/I2C

## Serial Peripheral Interface (SPI) Protocol

- The Pins: Interface and Function:  
The two lines that structure I2C transport have the accompanying functions:
  - **Serial Data (SDA):** It is used to make the I2C as a half-duplex bus by exchanging information among the devices.
  - **Serial Clock (SCL):** It used by the master to constrain the transmission rate.



# Input and Output SPI/I2C

## Comparison

	SPI	I2C
<b>Complexity</b>	Complex as device increases	Easy to chain many devices
<b>Speed</b>	Faster	slower
<b>Number of wires</b>	4	2
<b>Duplex</b>	Full Duplex	Half Duplex
<b>Number of devices</b>	Many, but there are practical limits and may get complicated	Up to 127 but may get complex as devices increases
<b>Number of masters and slaves</b>	Only 1 master but can have multiple slaves.	Multiple slaves and masters

# The instruction cycle/the fetch-decode-execute cycle

- Fetch-execute cycle is a standard procedure that depicts the means of sending and accepting information from/to the processor and memory that began by press a catch or programming order (turned on) and end with shut-day break, this procedure is finished by the CPU.
- It is consisting of four primary stages:
  - the get stage,
  - the unravel stage,
  - the execute stage and
  - rehash Cycle.

# The instruction cycle/the fetch-decode-execute cycle

- At that point the CPU is prepared to accomplish some work:
  - **Fetching stage:** the initial step the CPU brings a few information and directions (activity that must be performed by the information) from primary memory. At that point store them in registers (its own inner brief memory zones) this is known as the 'fetch' some portion of the cycle.
  - **Decode stage:** in this progression, CPU comprehends the brought guidance; this procedure is called decode.
  - **Execute:** information preparing is done now, and the guidance is done on the information.
  - **Repeat Cycle:** after the execute stage finished, the CPU sets itself up to start another cycle again.

# The instruction cycle/the fetch-decode-execute cycle

- During the fetch-execute cycle the control unit, information transport and address transport are all being used:
  - The control unit will direct the clock speed of the fetch-execute cycle, and enact either the read or write line.
  - The address transport will hold the location that is being gotten to in primary memory.
  - The information transport will move the information contained in the memory address forward and back between the processor and the memory address.

# Accelerator

- Internet of things (IoT) is associated universe of capacities, applications and administrations which empower association between physical articles embraced by advancements (for example Sensors, actuators, RFID, remote, portable, cloud and web).
- This quick increment of advancement, requests and focal points of IoT that accentuates the requirement for IoT quickening agents to:
  - Service the beneficial devices to get new viewpoints into activities and dynamic help to ventures.
  - Provide improvement of uses.
  - Unified cooperation for administrators and supervisors.
  - Raising cautions dependent on real-time data, connection of different occasions alongside examination and disconnected investigation.

# Section Outline

- Embedded Systems Memory Types



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

## Section 3

Embedded Memory

# Embedded Memory

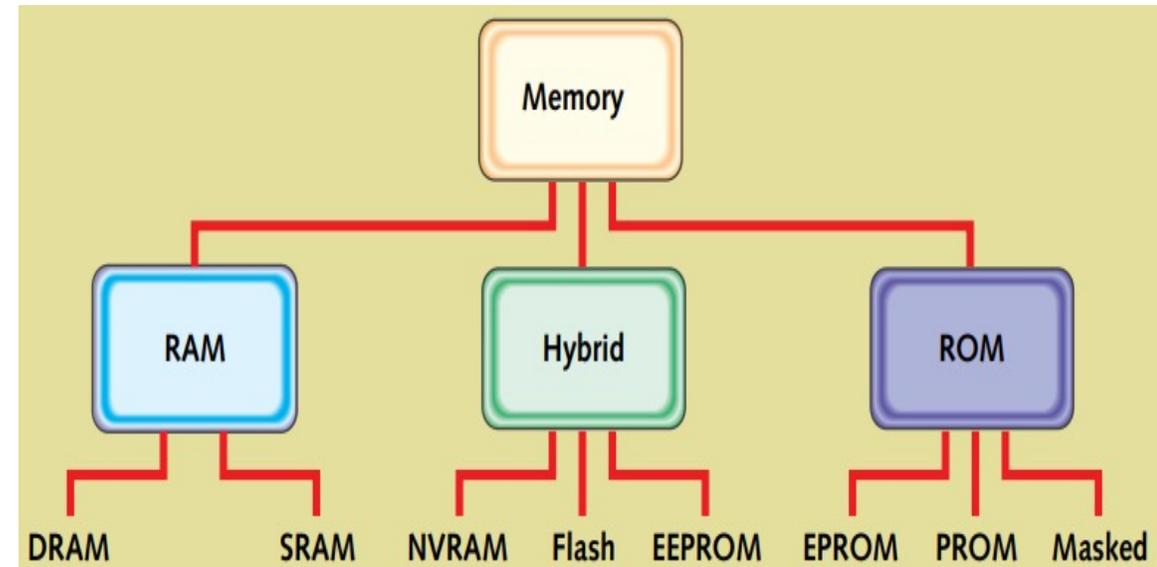
## Introduction

- Due to the increase in the amount of data required for many of these applications (such as video games and communications), embedded memory plays an important role in digital system applications.
- Moreover, the gap between processor speed, main memory and bus speed (memory wall) is getting wider, so more on-chip memory is needed to keep the processor busy and increase throughput.
- In addition to increasing the processor frequency, integrating multiple cores or functional units (called system-on-chip (SOC)) on the same chip also requires a larger memory size.
- Embedded memory occupies more than 50% of the chip area and more than 80% of the number of transistors.

# Embedded Systems Memory Types

## Introduction

- Many types of storage devices can be used in modern computer systems.
- The names of memory types often reflect the historical nature of the development process and are often more confusing than insightful situations.
- They are classified from the memory devices point of view as:
  - RAM,
  - ROM, or
  - a mixture of the two



# Embedded Systems Memory Types

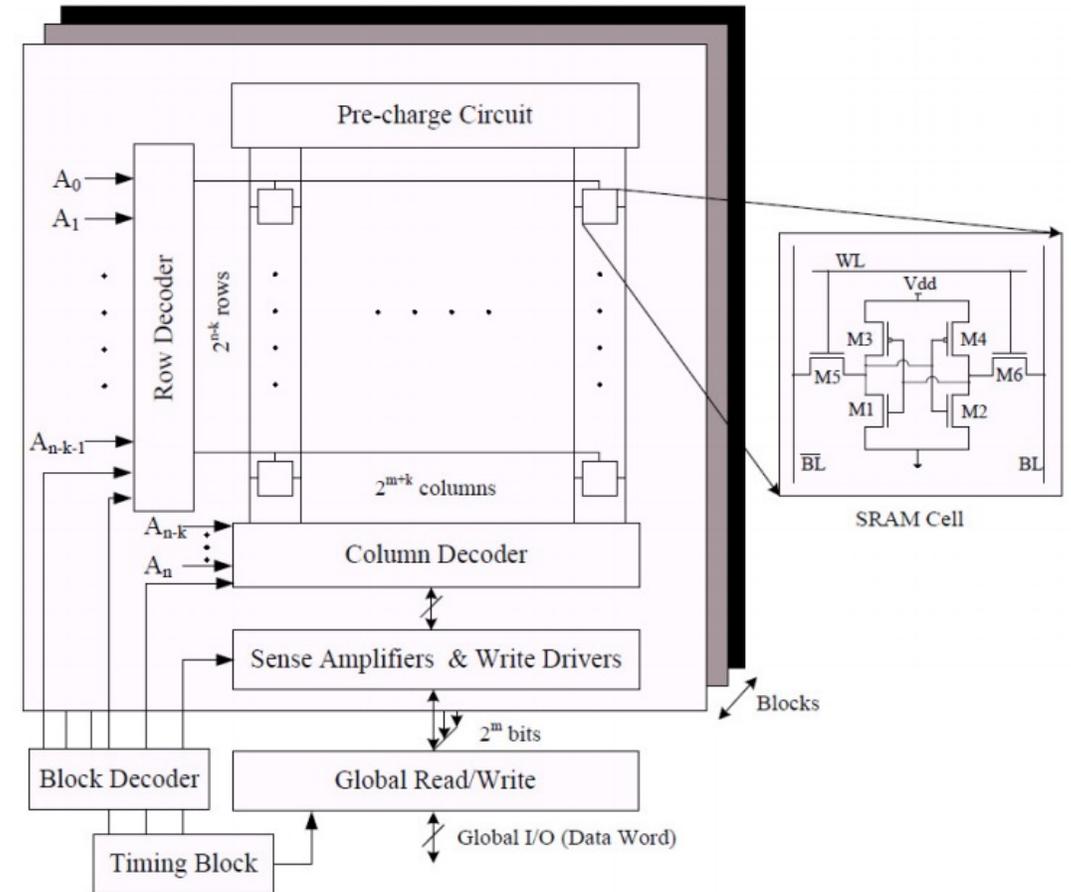
## Static RAM (SRAM)

- SRAM is commonly utilized for fast registers, stores and moderately little memory banks, for example, an edge cushion on a presentation connector.
- Interestingly, the primary memory in a PC is regularly unique Slam (DRAM, D-RAM).
- SRAM is designed to meet two requirements:
  - to provide a direct interface to the CPU at a speed that DRAM cannot reach, and
  - to replace DRAM in systems with very low power consumption.
- A SRAM store comprises of a variety of memory cells alongside fringe hardware, for example, address decoder, sense speakers and compose drivers and so on those empower perusing from and composing into the exhibit.

# Embedded Systems Memory Types

## Static RAM (SRAM)

- The memory cluster comprises of  $2^n$  expressions of  $2^m$  bits each. Each piece of data is put away in one memory cell.
- They share a typical word-line (WL) in each line and a piece line sets (BL, supplement of BL) in every section.
- The elements of each SRAM cluster are restricted by its electrical qualities.
- It comprises of two cross-coupled inverters (M3, M1 and M4, M2) and two access semiconductors (M5 and M6).
- The entrance semiconductors are associated with the wordline at their particular door terminals, and the bitlines at their source/channel terminals.



# Embedded Systems Memory Types

## Dynamic RAM (DRAM)

- For more than a quarter century, dynamic random access memory (DRAM)-ICs have been existed and developed from the most punctual 1-kilobit (Kb) to the ongoing 1-gigabit (Gb) era.
- This evolution has been attained through advances in both semiconductor procedure and circuit structure innovation; huge advances in process innovation that have drastically decreased element size, allowing ever more elevated levels of joining.
- A large number of the advances in process innovation have been joined or empowered by progresses in circuit structure innovation.

# Embedded Systems Memory Types

## Dynamic RAM (DRAM)

- DRAM types and methods of activity will be outlined as follows:
  - **The 1<sup>st</sup> Generation/1k DRAM:** This type of memories is sorted out as a sensible square of memory components. Its cluster is comprised of 1,024 memory components spread out in a square of  $32_{\text{rows}} \times 32_{\text{columns}}$ .
  - **The 2<sup>nd</sup> Generation/4k-64 Meg DRAM:** As a second generation, DRAMs can be distinguished by the presentation of multiplexed address inputs, numerous memory clusters, and the memory cell of a single transistor/ a single capacitor. Besides, they offer more methods of activity for more noteworthy adaptability or higher speed activity.
    - *The size of this type of memories varies between 4k (4,096<sub>address locations</sub> × 1<sub>input/output</sub>) to 64 Meg i.e. 67,108,864 bits in different sizes as:*
      - 16 Meg x 4 (16,777,216<sub>address locations</sub> × 4<sub>input/output</sub> )
      - 8 Meg x 8 (8 Meg<sub>address locations</sub> × 8<sub>input/output</sub> )
      - 4 Meg x 16 (4 Meg<sub>address locations</sub> × 4<sub>input/output</sub> )

# Embedded Systems Memory Types

## Dynamic RAM (DRAM)

- DRAM types and methods of activity will be outlined as follows:
  - **The 2<sup>nd</sup> Generation/4k-64 Meg DRAM:** As a second generation, DRAMs can be distinguished by the presentation of multiplexed address inputs, numerous memory clusters, and the memory cell of a single transistor/ a single capacitor. Besides, they offer more methods of activity for more noteworthy adaptability or higher speed activity.
    - *Furthermore, as a major leap in the second generation is the transition of the power supply to be single 5V power supply at the density of 64kbit. This transition simplifies the system design from different aspects such as them memory and the processor. This is in addition to changing the technology from the NMOS to CMOS in order to fulfil the concerns over the speed, the size and the power the density of 1Mbits.*
  - **The 3<sup>rd</sup> Generation/Synchronous DRAM:** Its design has been modified by includes an interface between 2<sup>nd</sup> Generation DRAM's core and the off-chip control. This is in addition to reserve the clock (CLK) for executing both of commands and operations.

# Embedded Systems Memory Types

## Flash Memory

- In the next few years, convenient frameworks will require more non-volatile recycling, whether it is for information storage applications, with high thickness and high synthesis throughput, or for code execution settings.
  - In other words, it has fast and irregular access rights.
- In view of these market demands, a significant method of arranging Flash projects and relative progress is to characterize two important parts of the application:
  - code storage, where the program or work frame is stored and executed by the chip or microcontroller, which continuously record and peruse information files related to pictures, music and voice.

# Embedded Systems Memory Types

## Flash Memory

- Different types of Flash units and structures have been proposed previously. From design point of view, flash memories can be divided as:
  - Fowler-Nordheim Tunneling (FN),
  - Channel Hot Electrons (CHE),
  - Hot Holes (HH) and
  - Source Side Hot Electronics (SSHE).

## Section Outline

- Compute-Constrained Devices
- Constrained Node, Constrain Network and Constrained-Node Network
- The need for Management of constrains devices and constrained devises restrictions
- Applications for Constrained Devices



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

# Section 4

## Causes and Implications of Memory

# Causes and Implications of Memory

## Introduction

- IoT system classified into three networks:
  - First is the Edge network which contains devices (Embedded, sensors, actuators that can be constrained or unconstrained) that operate with external world monitoring, sensing, actuating, etc.
  - Second is the Fog network which is the Gateway and high-end servers that broker, data collecting and processing, commanding, analytics, etc.
  - Third is the cloud network which is the cloud platforms and high-end servers that store and Machine, deep learning, without direct active management by the user.

# Compute-Constrained Devices

- Constrained devices are the edge nodes having (sensors, actuators, smart object or smart devices micro controller) which can handle a specific application target, linked to gateway-like devices and return communication with the IoT cloud platforms.
- They communicate with low-power and data rate- wireless protocols.
- IoT have embedded computing devices spread within it and they are considered the resource constrained.
- This resources constraint affects memory, processing capabilities, the low-power radio standards utilized, and constraint the network interfaces.

# Compute-Constrained Devices

## Comparison

Type	CPU	RAM	Flash/ROM
<b>Crossbow TelosB</b>	16-Bit MSP430	10 KB	48 KB
<b>RedBee EconoTAG</b>	32-Bit MC13224v	96 KB	128 KB
<b>Atmel AVR Raven</b>	8-Bit ATmega1284P	16 KB	128 KB
<b>Crossbow Mica2</b>	8-Bit ATmega 128L	4 KB	128 KB

# Constrained Node, Constrain Network and Constrained-Node Network

- Some basic terms should be identified in order to help in understanding the work of constrained environment, which is according to the Terminology for Constrained-Node Networks as:
  - **Constrained Node:** A node where some of the characteristics that are otherwise pretty much taken for granted for Internet nodes at the time of writing are not attainable, often due to cost constraints and/or physical constraints on characteristics such as size, weight, and available power and energy.
  - **Constrained-Node Network:** A network whose characteristics are influenced by being composed of a significant portion of constrained nodes.

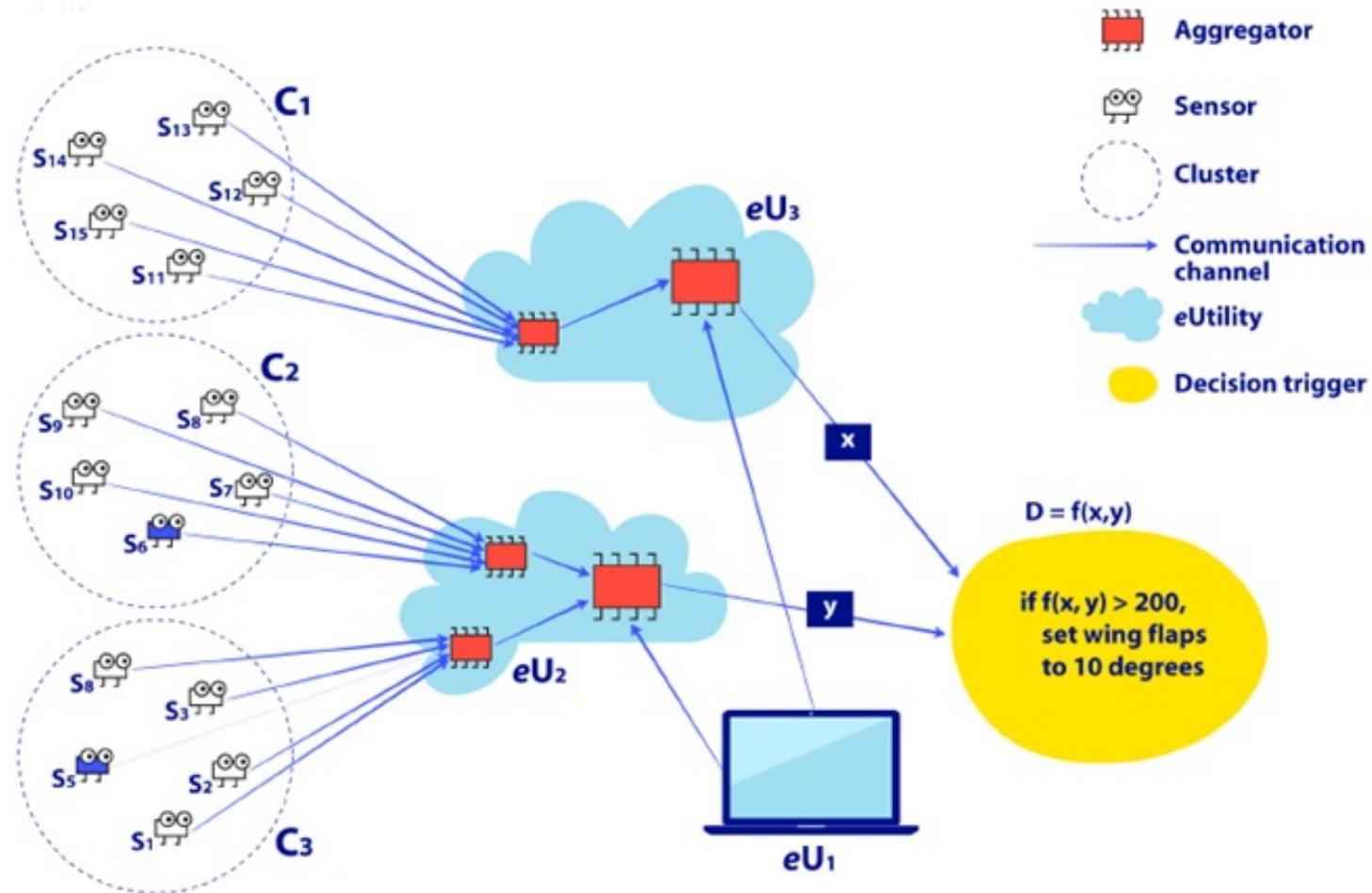
# Constrained Node, Constrain Network and Constrained-Node Network

- Some basic terms should be identified in order to help in understanding the work of constrained environment, which is according to the Terminology for Constrained-Node Networks as:
  - **Sensors:** A sensor is an electronic utility that measures physical properties such as temperature, acceleration, weight, sound, location, presence, identity, etc.
  - **Actuator:** Actuator is the unit that obtain physical movement by converting energy usually electrical, air or hydraulic into mechanical action (enables movement), like the muscles in human body.
  - **Cluster:** Cluster is a set of sensors and actuators with the data they output, they are not physical, and can change their sensors and data any time, in constrained-node network grouping similar objects (sensors, actuator)

# Constrained Node, Constrain Network and Constrained-Node Network

- Some basic terms should be identified in order to help in understanding the work of constrained environment, which is according to the Terminology for Constrained-Node Networks as:
  - **Communication channel:** Communication channel is the physical transmission medium in which data is transfer (. e.g. USB, wireless, direct)
  - **Aggregators:** An aggregator is a software implementation based on mathematical function(s) that transforms groups of raw data into intermediate, aggregated data.
  - **eUtility:** eUtility is a service or hardware or software that support aggregators suppling data and enable computing resources and storage as need.
  - **Decision Trigger:** Decision Trigger is the end-purpose software which computes and takes action if needed to satisfy the purpose, specification, and requirements.

# Constrained Node, Constrain Network and Constrained-Node Network



# The need for Management of constrained devices and constrained devices restrictions

- It is now clear that constrained devices have limited storage capabilities CPU memory, so this network itself considered may be as challenging one.
- Constrained devices can be connected to unconstrained network, constrained devices responsible of gathering the information, generating and sending the information to one or more station.
- Energy-Management for constrained devices that is connected to a network form a challenge, Energy-Management responsible for supply energy to various devices and component even a devices contained batteries they also monitored and managed.
- Constrained devices are having considerable small size and are not fixed. Because of their size and frequently changing location property they are not able to access the power all the time.

# The need for Management of constrained devices and constrained devices restrictions

- Device life cycle and quality with time and environment must consider to not causing a restriction while using constrained devices.
- The need for access technology rises up with constrained devices to not impose a restriction, so the constrained access technologies are a demand to connect the constrained and traditional unconstrained networks.
- The need for Constrained Access Technologies arises due to resource restrictions, embedded devices use sensors and actuators use the low-power, low-data-rate wireless access technologies like Digital Enhanced Cordless Telecommunication (DECT), Low-Rate Wireless Personal Area Networks (LR-WPAN), Ultra Low Energy (ULE), or Bluetooth for network connectivity.

# The need for Management of constrains devices and constrained devises restrictions

- Security and privacy issue is essential parameter for reliable and safe communication between constrained-node networks.
- However protecting the communication between devices with limited resources is a complex issue specially with transmitting sensitive data such military and medical application.
- On other hand the cost factor will rise up. Some protocols developed for reliable and safe communication between constrained-node network to insure securing and packet delivery 100% such the ContikiMAC Radio Duty cycle protocols.

# Applications for Constrained Devices

- The constrained devices have wide range of applications in my areas, which they are summarized as:
  - **Medical Applications:** constrained devices used in wide range for health su
  - **Building Automation:** lighting, pumps, boilers, environment temperature, building plants irrigation, Elevator control, air fans, CO2 levels, and more equipment prevision and control.
  - **Building automation:** It requires the deployment of a large number (10 to 100,000) of sensors that monitor the status of devices, parameters inside the building.
  - **Transport Application:** connecting cars, vehicle tracking systems, smart parking, electronic toll-collection systems, traffic light, logistic and fleet management, vehicle control, traffic and transportation management.
  - **Home Automation:** It controls lighting, heating, doors and windows, air conditioning, appliances; gardens .





# Thank You

 Dr Marios Raspopoulos

 +357 24694070

 [mraspoulos@uclan.ac.uk](mailto:mraspoulos@uclan.ac.uk)

 <http://www.uclancyprus.ac.uk/>

**PROUDER**  
Introducing Recent Electrical Engineering  
Developments Into Undergraduate Curriculum



# This week's topics...

- Common Microcontrollers (Arduino uno/mega2560, Raspberry-Pi, ARM), Real-time systems and embedded software
- OS and Drivers (End Device Program)
- Hardware & Software Requirements
- Sensing components and devices
- Sensor modules, nodes and systems (Typical IoT Sensors: e.g. Temperature, proximity, inertial, Sonar, LIDAR etc.)
- Actuators

# Section Outline

Microcontrollers

Real-time systems

Embedded software



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

## Section 1

Common Microcontrollers, Real-time systems and embedded software

# Implementing the Internet of Things

## Introduction

- The Internet of Things can be implemented by integrating the following technologies:
  - Microcontrollers: the «brain» capable of interacting with the World (sensing and actuating);
  - Real-time systems: IoT systems should be capable of operating in real-time and with bounded delay;
  - Embedded software: specific software written for microcontrollers
- The following slides provide an introduction to those concepts.

# Microcontrollers

## Introduction

- A microcontroller (**MCU** for microcontroller unit) is a small computer on a single metal-oxide-semiconductor (MOS) integrated circuit (IC) chip.
- In modern terminology, it is similar to, but less sophisticated than, a system on a chip (SoC); a SoC may include a microcontroller as one of its components.
- A microcontroller contains one or more CPUs (processor cores) along with memory and programmable input/output peripherals.
- Program memory in the form of ferroelectric RAM, NOR flash or OTP ROM is also often included on chip, as well as a small amount of RAM.
- Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications consisting of various discrete chips.

# Microcontrollers

## Examples



Two ATmega microcontrollers



A PIC microcontroller

Pictures from Wikipedia

# Arduino Uno

## Specifications

- Arduino Uno is a microcontroller board based on the ATmega328P.
- 14 digital input/output pins (of which 6 can be used as PWM outputs)
- 6 analog inputs
- a 16 MHz ceramic resonator (CSTCE16M0V53-R0)
- a USB connection, a power jack, an ICSP header and a reset button.

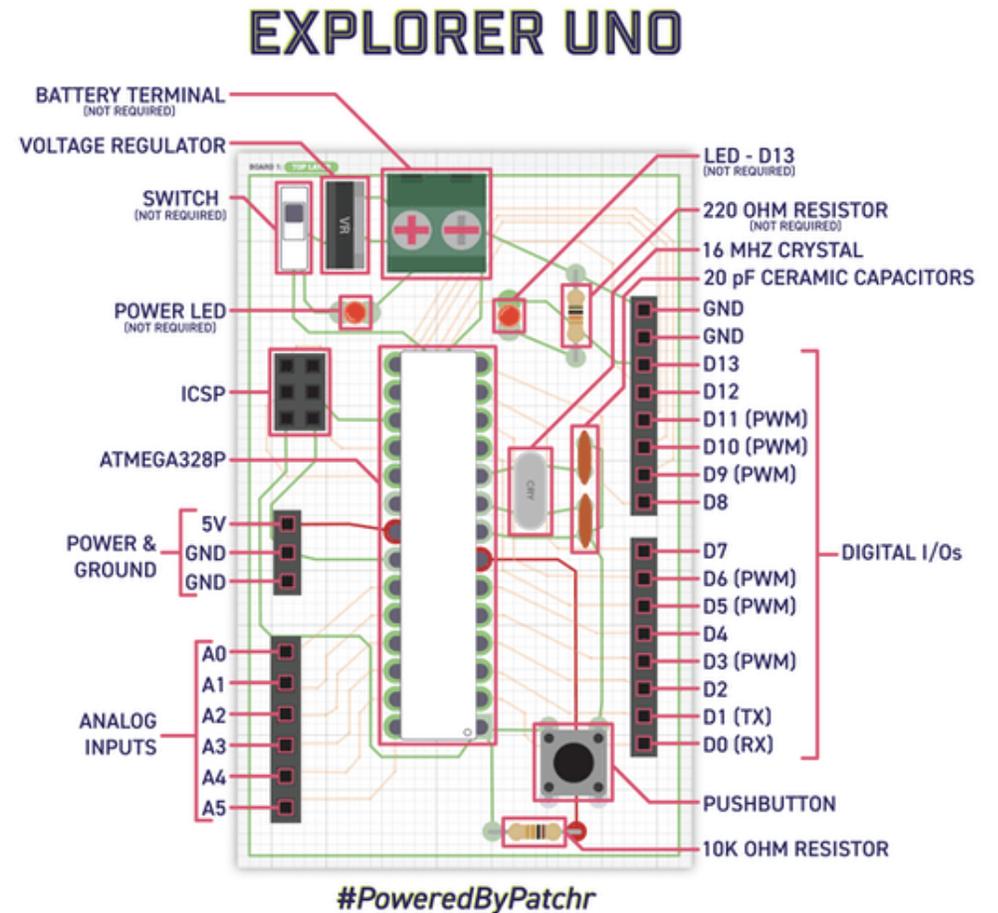
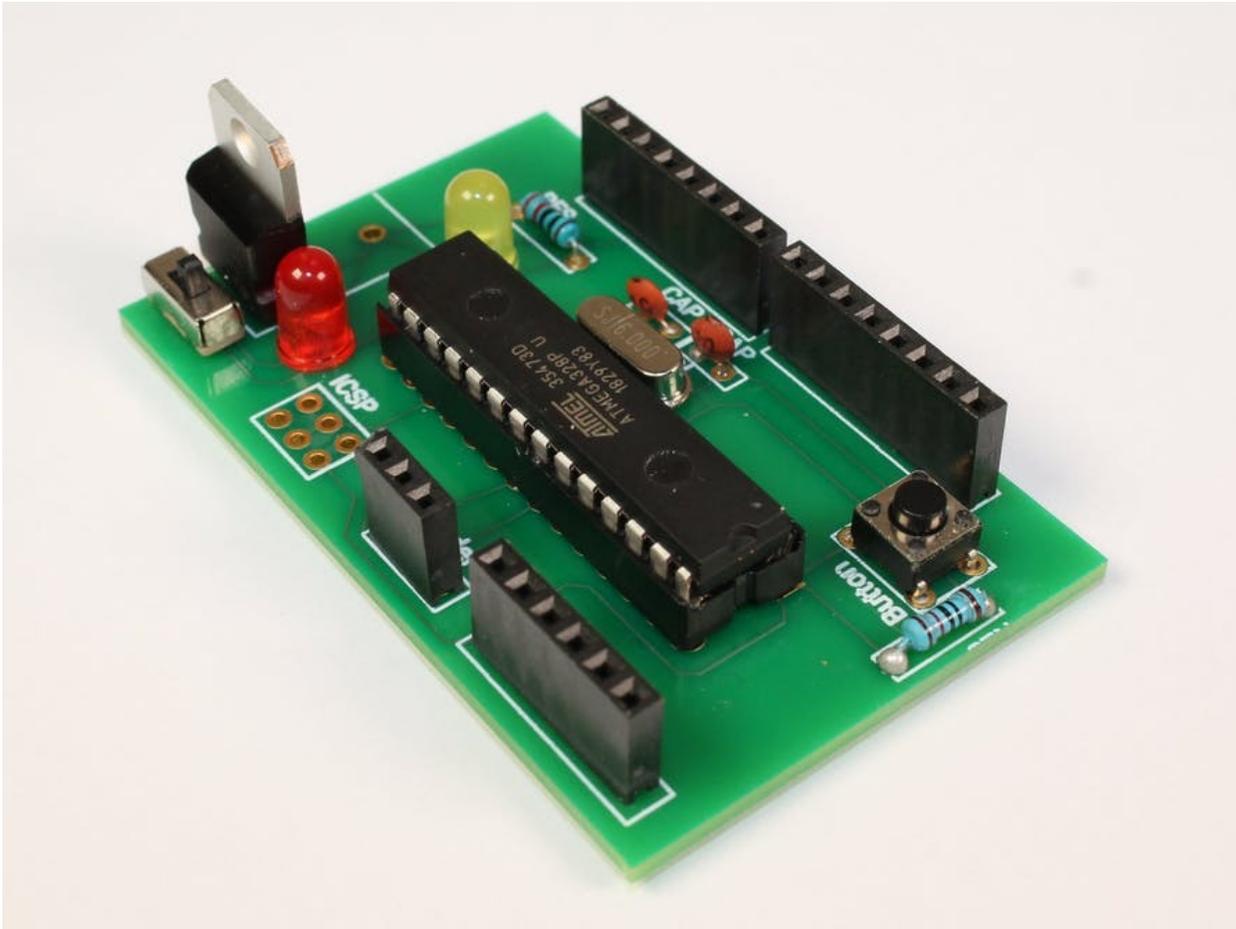


From: <https://store.arduino.cc/arduino-uno-rev3>

# Arduino UNO

## Build your own Arduino Uno!

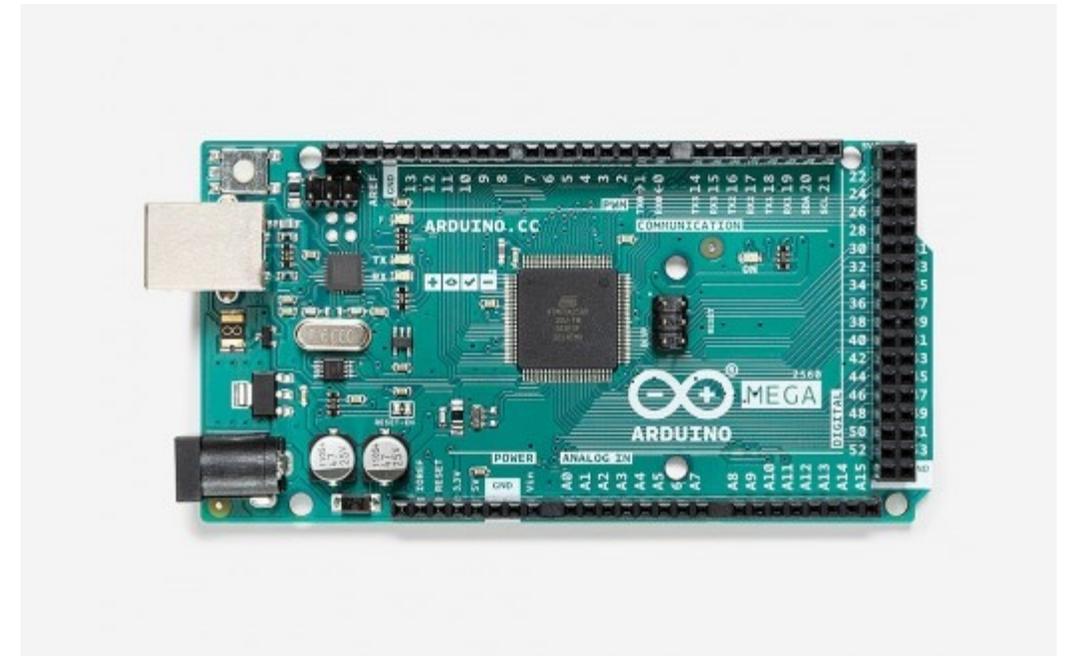
- [https://create.arduino.cc/projecthub/patchr\\_io/explorer-uno-pcb-template-design-your-own-de89da](https://create.arduino.cc/projecthub/patchr_io/explorer-uno-pcb-template-design-your-own-de89da)



# Arduino Mega

## Specifications

- The Arduino Mega 2560 is a microcontroller board based on the ATmega2560.
- 54 digital input/output pins (of which 15 can be used as PWM outputs)
- 16 analog inputs, 4 UARTs (hardware serial ports)
- a 16 MHz crystal oscillator
- a USB connection, a power jack, an ICSP header, and a reset button.

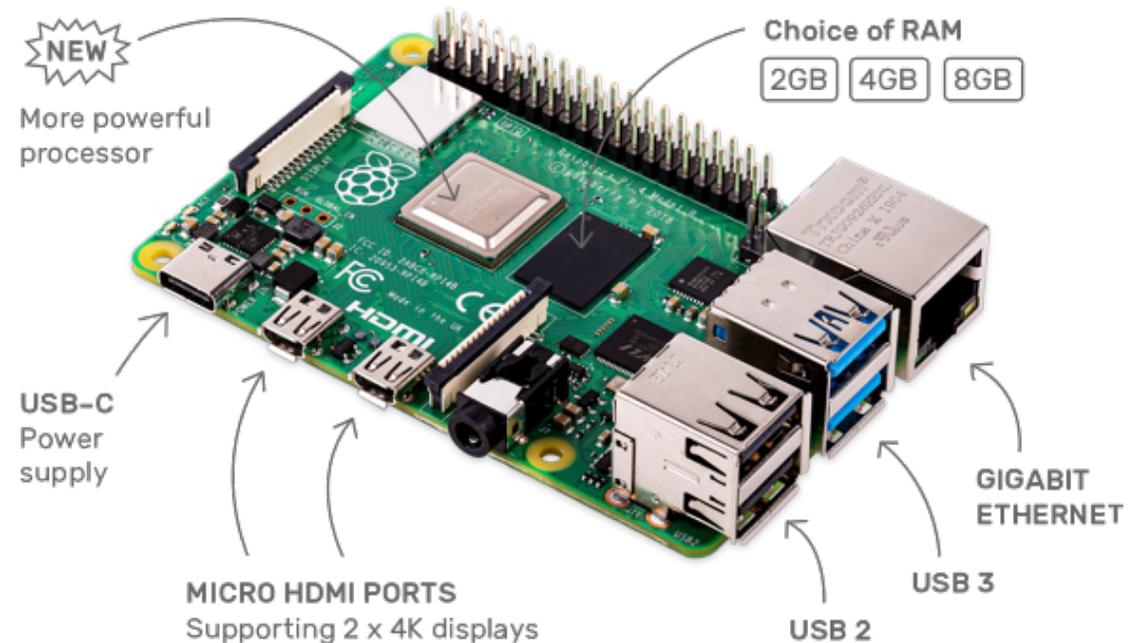


From: <https://store.arduino.cc/arduino-mega-2560-rev3>

# Raspberry Pi

## Specifications

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- 2GB, 4GB or 8GB LPDDR4-3200 SDRAM (depending on model)
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 USB 3.0 ports; 2 USB 2.0 ports.
- Raspberry Pi standard 40 pin GPIO header (fully backwards compatible with previous boards)
- 2 × micro-HDMI ports (up to 4kp60 supported)
- 4-pole stereo audio and composite video port
- H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)
- OpenGL ES 3.0 graphics
- Micro-SD card slot for loading operating system and data storage
- Power over Ethernet (PoE) enabled (requires separate PoE HAT)



# ARM Microcontrollers

## Sample of ARM MCU specs



**STM32L4+ MCU Series**  
**32-bit Arm® Cortex®-M4 (DSP + FPU) – 120 MHz**



	Product line	Flash (KB)	RAM (KB)	Memory I/F	Op-Amp	Comp.	Sigma Delta Interface	12-bit ADC 5 Msps 16-bit HW oversampling	USB2.0 OTG	TFT Display Interface	*Chrom-GRC™	MIPI-DSI	AES 128-/256-bit	
<ul style="list-style-type: none"> <li>• USART, SPI, I2C</li> <li>• 2x Quad-SPI</li> <li>• 16- and 32-bit timers</li> <li>• SAI + audio PLL</li> <li>• CAN</li> <li>• Camera IF</li> <li>• ART Accelerator™</li> <li>• Chrom-ART Accelerator™</li> <li>• 2x 12-bit DACs</li> <li>• Temperature sensor</li> <li>• Low voltage 1.71 to 3.6V</li> <li>• VBAT mode</li> <li>• Unique ID</li> <li>• Capacitive touch-sensing</li> </ul>	<b>STM32L4R9/S9</b>													
	<b>STM32L4R9 USB OTG &amp; MIPI-DSI</b>	1024 to 2048	640	SDIO FSMC	2	2	8x ch	1	•W	•	•	•		
	<b>STM32L4S9 USB OTG &amp; MIPI-DSI &amp; AES</b>	1024 to 2048	640	SDIO FSMC	2	2	8x ch	1	•	•	•	•	•	
	<b>STM32L4R7/S7</b>													
	<b>STM32L4R7 USB OTG &amp; TFT Interface</b>	1024 to 2048	640	SDIO FSMC	2	2	8x ch	1	•	•	•			
	<b>STM32L4S7 USB OTG &amp; TFT Interface &amp; AES</b>	2048	640	SDIO FSMC	2	2	8x ch	1	•	•	•		•	
	<b>STM32L4R5/S5</b>													
	<b>STM32L4R5 USB OTG</b>	1024 to 2048	640	SDIO FSMC	2	2	8x ch	1	•					
	<b>STM32L4S5 USB OTG &amp; AES</b>	2048	640	SDIO FSMC	2	2	8x ch	1	•					•
	<b>STM32L4P5/Q5</b>													
<b>STM32L4P5 USB OTG</b>	512 to 1024	320	SDIO FSMC	2	2	4 ch	2	•	•					
<b>STM32L4Q5 USB OTG &amp; AES</b>	1024	320	SDIO FSMC	2	2	4 ch	2	•	•				•	

# Real-time Systems

## Introduction

- A real-time system has been described as one which "controls an environment by receiving data, processing them, and returning the results sufficiently quickly to affect the environment at that time".
- Real-time software may use one or more of the following: synchronous programming languages, real-time operating systems, and real-time networks, each of which provide essential frameworks on which to build a real-time software application.

# Embedded Software

## Introduction

- Embedded software is computer software, written to control machines or devices that are not typically thought of as computers, commonly known as embedded systems. It is typically specialized for the particular hardware that it runs on and has time and memory constraints. This term is sometimes used interchangeably with firmware.
- A precise and stable characteristic feature is that no or not all functions of embedded software are initiated/controlled via a human interface, but through machine-interfaces instead.
- Manufacturers build embedded software into the electronics of cars, telephones, modems, robots, appliances, toys, security systems, pacemakers, televisions and set-top boxes, and digital watches, for example

# Section Outline

Introduction

Most diffused OSs in brief



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

## Section 2

Operating systems and drivers

Hardware and Software requirements

# IoT Operating Systems

## Introduction

- The main idea of the internet of things is connectivity between the web and sensor-based tiny devices on a system.
- Each IoT device has its perspective. So variability is obvious for the operating systems.
- IoT operating system is software that ensures connectivity between IoT applications and embedded devices.
- The following slides propose some open source IoT operating systems which are practical to use for IoT devices.

# IoT Operating Systems

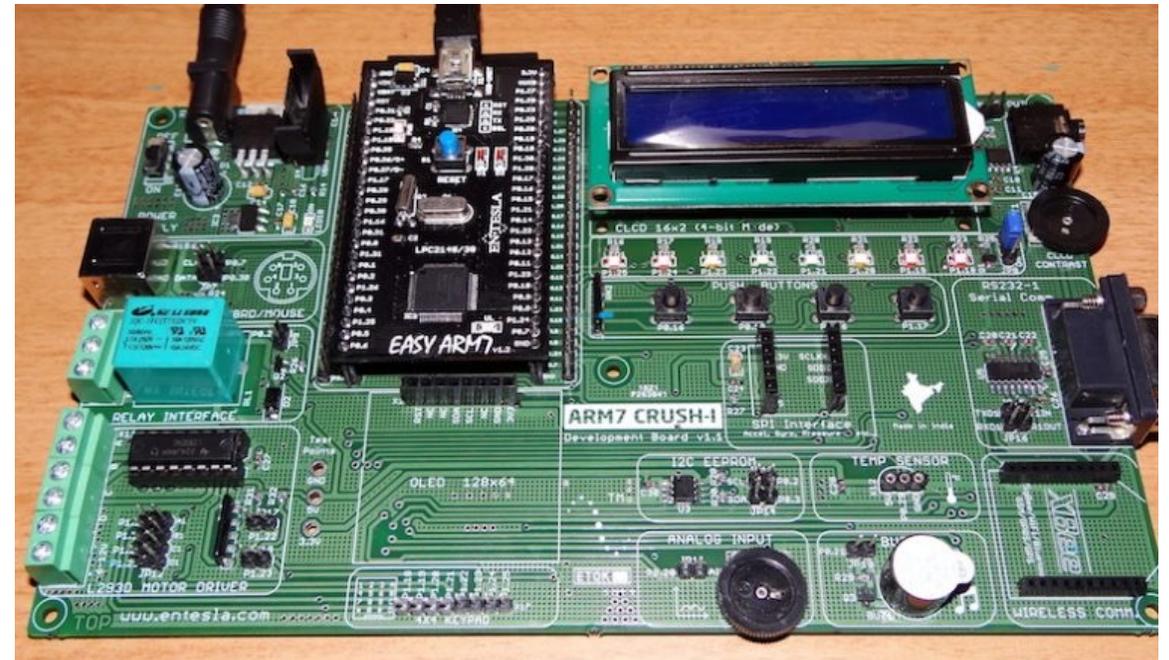
## Arduino IDE

- The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board.
- It runs on Windows, Mac OS X, and Linux.
- The environment is written in Java and based on Processing and other open-source software.
- This software can be used with any Arduino board.
- <https://www.arduino.cc/en/main/software>

# IoT Operating Systems

## ARM MCU Programming

- Different Microcontroller Development Kits for ARM based microcontrollers are available (including Arduino IDE)
- Need for a hardware programmer (see photo)
- Libraries are hardware-specific



# IoT Operating Systems

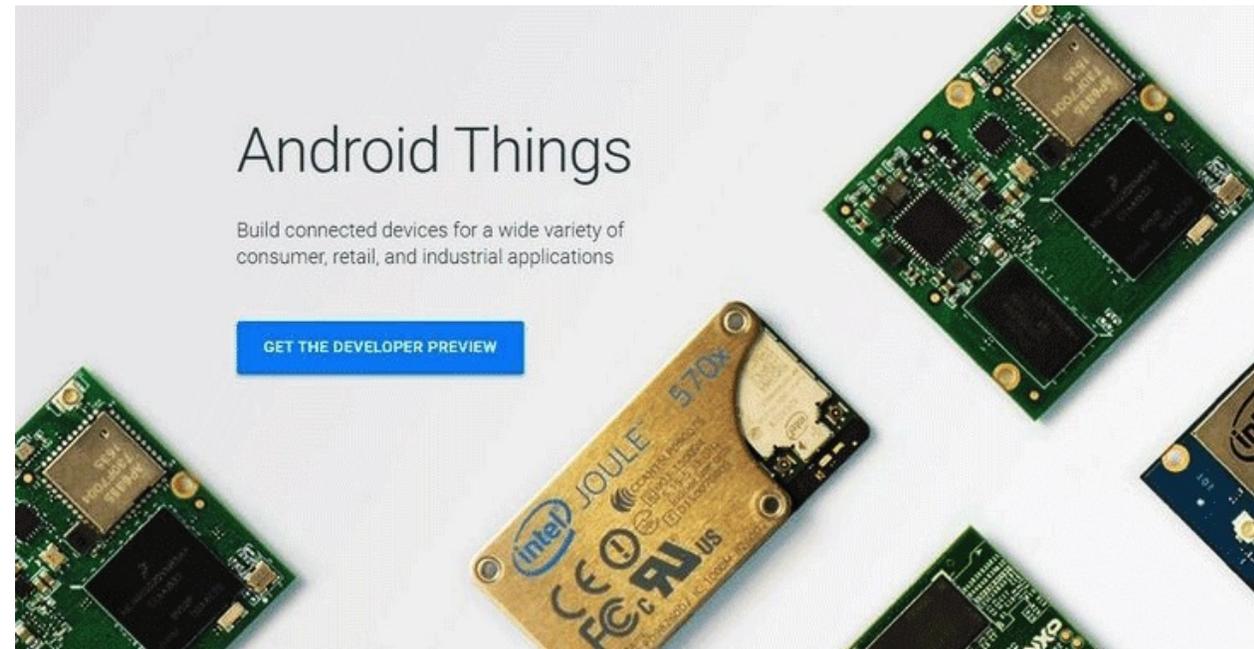
## Contiki

- Contiki is an open-source IoT operating system for low power microcontrollers and other IoT devices to run using Internet protocol IPv6, and IPv4.
- This operating system supports wireless standard CoAP, 6lowpan, RPL.
- It contains a built-in Internet protocol suite.
- Only 10kb of RAM and 30 kb of ROM is needed to run this Operating system.
- The core language of this operating system is C language.
- <https://sourceforge.net/projects/contiki/>

# IoT Operating Systems

## Android Things

- Android Things is an IoT Operating System by Google.
- It can run on low power and supports Bluetooth and WiFi technology.
- Android Things uses only 32-64 Kb of RAM as it is a lightweight operating system.
- <https://github.com/androidthings>



# IoT Operating Systems

## Riot

- Riot is one of the free open source IoT operating systems built for IoT services.
- Riot has a huge development community, and it was released under an unclonable GNU Lesser General Public License. For these two reasons, Riot is called Linux of the IoT world.
- With low power use capacity, Riot is built upon microkernel architecture with C, C++ language.
- This open source IoT os supports full multithreading and SSL/TSL libraries, for example, wolfSSL.
- The processor of Riot is 8bit, 16bit and 32 bit.
- A port of this operating system makes it possible to run as Linux or macOS process.
- Provides content-centric networking and network protocols such as TCP, UDP, and CoAp.
- <https://github.com/RIOT-OS/RIOT>

# IoT Operating Systems

## Apache Mynewt

- This IoT OS is built for tiny embedded IoT devices.
- This is a real-time operating system under Apache License 2.0 which provides a complete environment for developing, managing, and operations of IoT devices.
- With rich libraries, modular-based operating systems like Apache Mynewt can work for long times.
- With 6 kb kernel Mynewt is very useful for building embedded systems (industrial IoT equipment, medical devices) among various microcontrollers.
- Maintains up to 32 connections simultaneously.
- Console, shell, and bootloader support this operating system.
- Apache Mynewt supports priority-based scheduling, preemptive multithreading, multistage software watchdog, memory heap and memory pool allocation, etc.
- <https://github.com/apache/mynewt-core>

# IoT Operating Systems

## Huawei LightOS

- IoT OS of Huawei provides a standard API for the diverse IoT fields. LightOS is a secure, interoperable, low-power operating system.
- LightOS uses middleware to remove the extra cost for the development of IoT devices.
- LightOS contains the smallest kernel (6kb) comparing with other operating systems.
- Various network access protocols are supported: NB-IoT, Ethernet, Bluetooth, Wifi, Zigbee, and more.
- <https://github.com/LiteOS>

# IoT Operating Systems

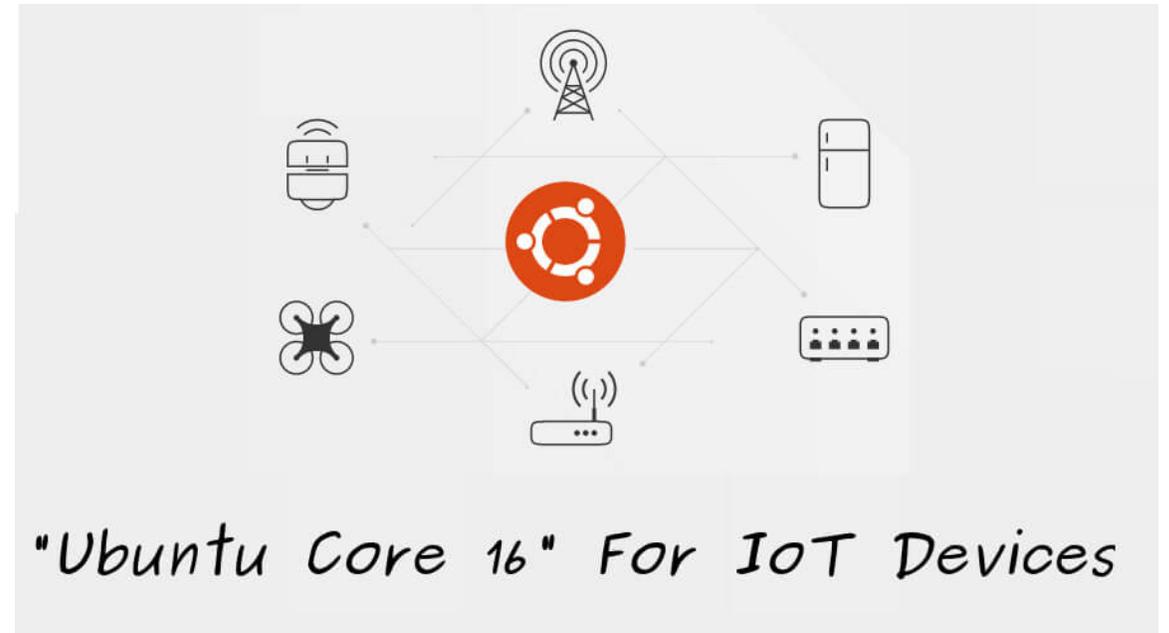
## Zephyr

- Zephyr is a real-time operating system (RTOS) built for IoT applications that get support from Linux Foundation.
- Easy integration of various IoT architectures makes it popular among IoT specialists.
- Interconnectivity technology (Bluetooth LE, Wifi, 6Lowpan, NFC) is the most prominent characteristic of this IoT Operating System.
- It is a library-based operating system with reliable memory protection.
- 8kb of Ram and 512 kb of ROM is necessary to operate this operating system.
- <https://github.com/zephyrproject-rtos/zephyr>

# IoT Operating Systems

## Zephyr

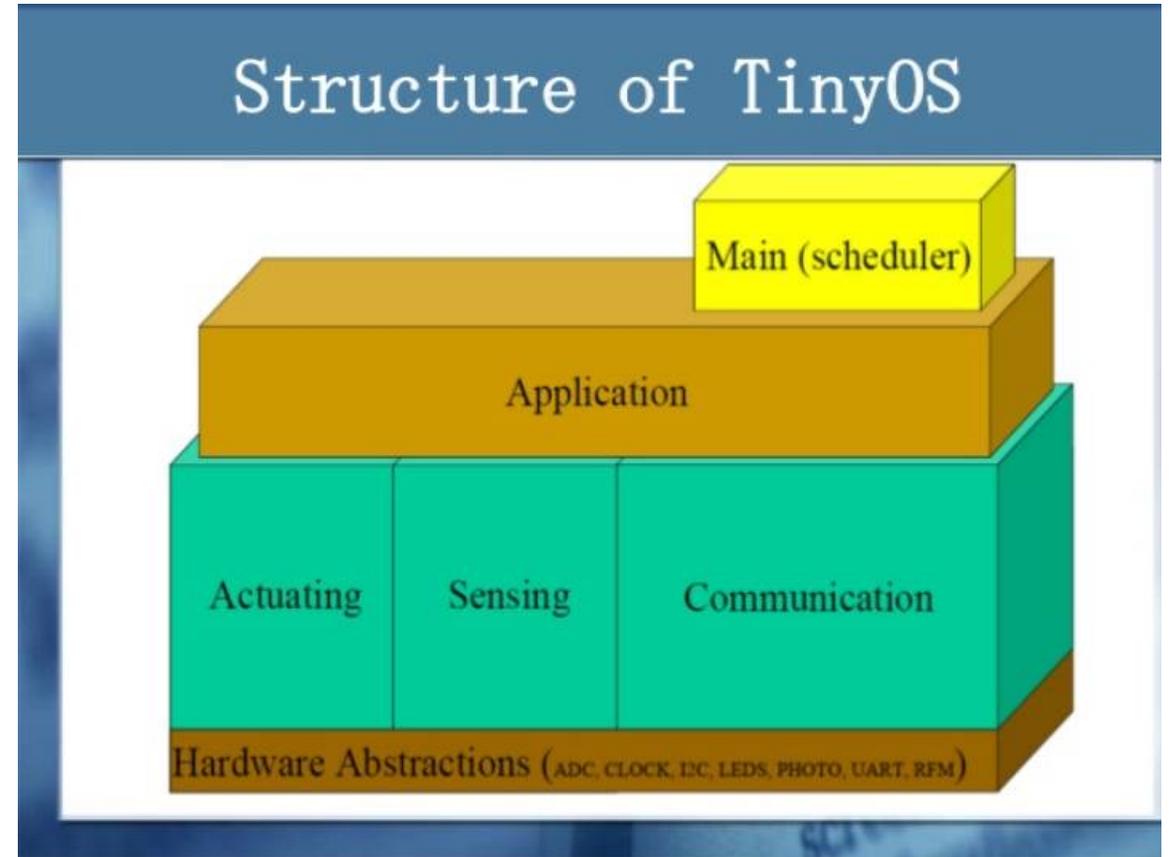
- Snappy is a Ubuntu core IoT OS.
- It derives from Linux package snap which includes libraries, kernel, and major applications.
- Snappy guarantees strong security to IoT devices with the help of Ubuntu community research.
- Distributes applications as Snap is a native packaging system.
- <https://ubuntu.com/download/iot/raspberry-pi-2-3-core>



# IoT Operating Systems

## TinyOS

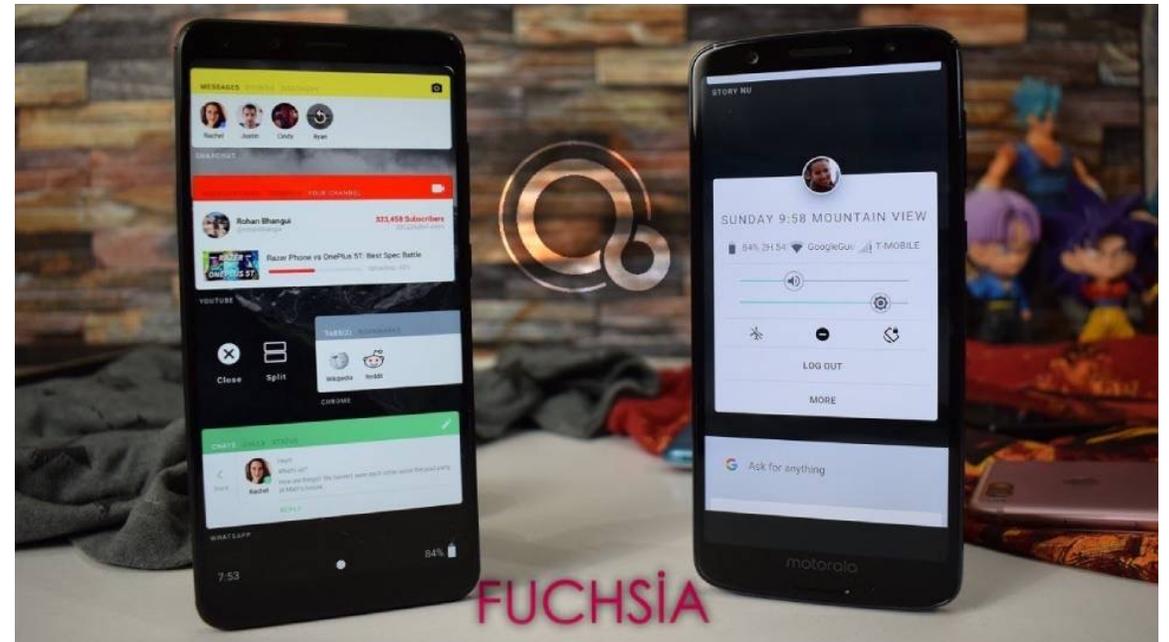
- TinyOS is a component-based open-source operating system.
- The core language of TinyOS is nesC which is a dialect of C language.
- A component of TinyOS enables abstractions of IoT systems, for example, sensing, packet communication, routing, etc.
- The developer group of this IoT Operating System is TinyOS Alliance.
- <https://github.com/saikatbsk/tinyOS>



# IoT Operating Systems

## Fuchsia

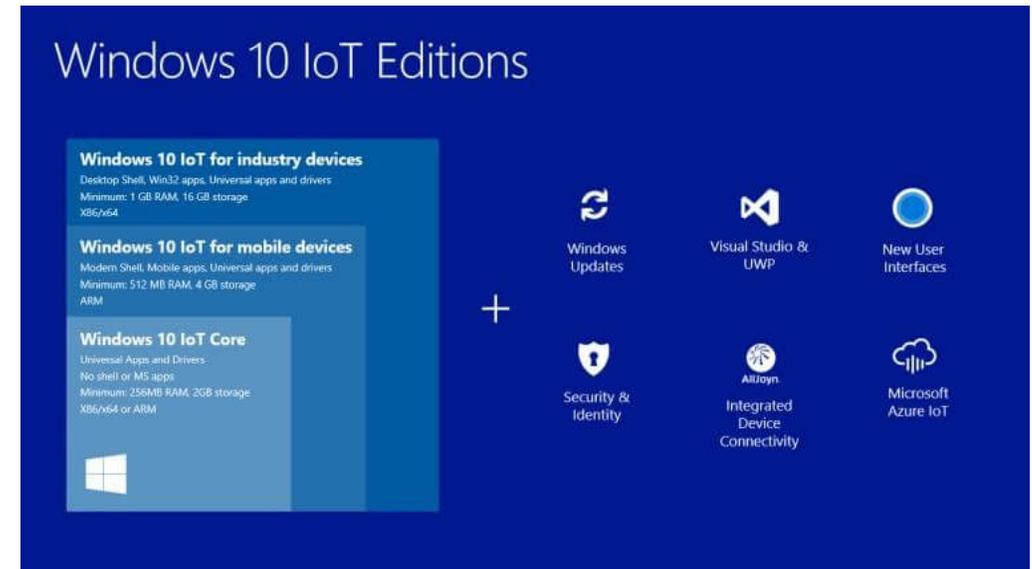
- Fuchsia is a microkernel-based operating system with effective connectivity solutions.
- Fuchsia runs well in low powered devices.
- The use of Node.js on operating system ensures application to run on the phone, tablets as well as IoT devices.
- The development language can be Dart, Go, Rust, C, C++.
- <https://github.com/FuchsiaOS>



# IoT Operating Systems

## Windows IoT

- Windows 10 IoT is a family of Windows 10 operating systems for the IoT sector.
- IoT enterprise operating system runs on the ARM processor.
- It leverages IoT connectivity, cloud experience, and offers various organizations to connect with IoT devices.
- This is not an open source OS



# IoT Operating Systems

## TizenRT

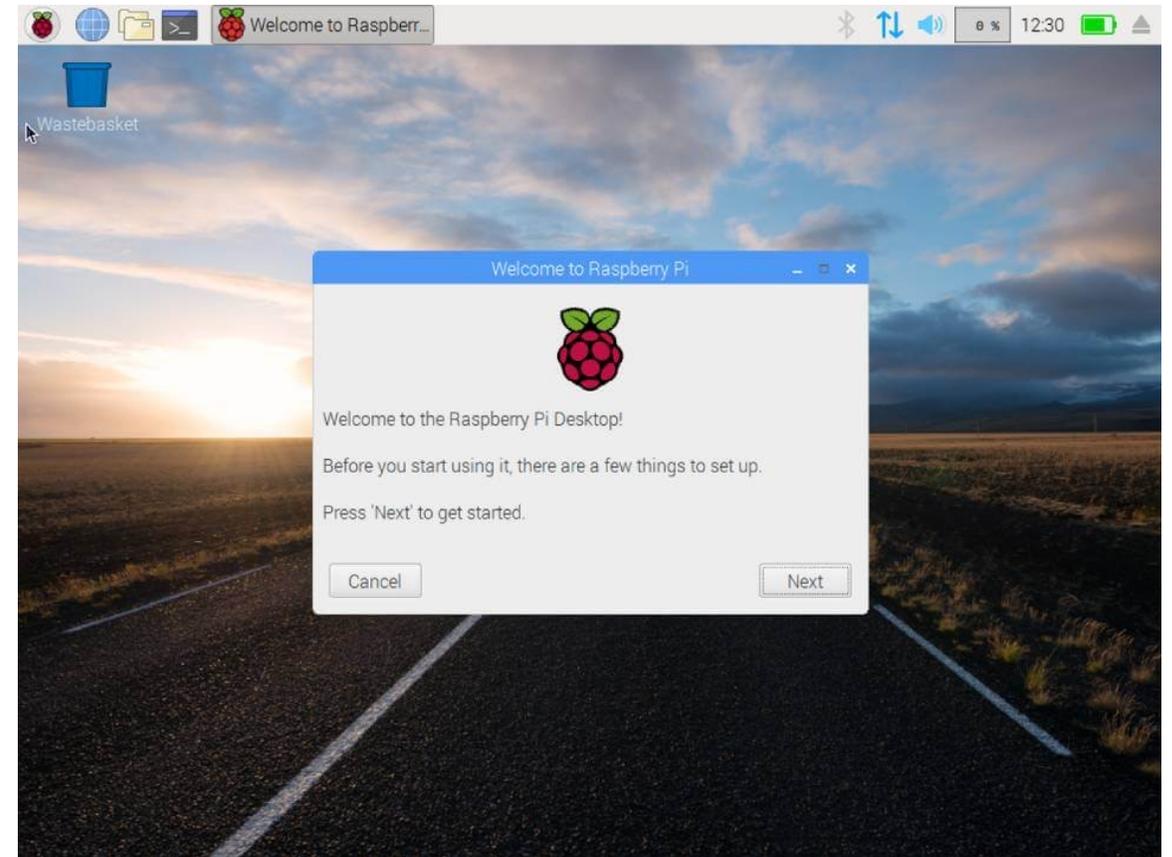
- It is a Linux based operating system for both mobile applications and small embedded systems.
- An upgraded version of Tizen can support smart TV, vehicles, home appliances, and more.
- It uses a shared infrastructure called “Tizen Common” to sustain the primary purpose of IoT development.
- Programming language C, C++, and HTML5 are the languages to develop Tizen.
- <https://github.com/Samsung/TizenRT>



# IoT Operating Systems

## Raspbian or Raspberry Pi OS

- Raspberry Pi is one of the most used devices for IoT development, and Raspbian is its own operating system.
- Raspbian is highly flexible for Raspberry Pi lines CPUs.
- Raspbian provides a huge number of pre-installed IoT software for general use, experimental, educational purposes, etc.
- This is Debian based IoT Operating System for all models of Raspberry Pi.
- <https://www.raspberrypi.org/downloads/raspberry-pi-os/>



# IoT Operating Systems

## Raspbian or Raspberry Pi OS

- Amazon FreeRTOS is an open-source microcontroller-based operating system for IoT development developed by Amazon.
- Enriched software libraries make it easy to connect with small IoT devices.
- This IoT Operating System uses the cloud service of Amazon Web Service called AWS IoT Core to run the IoT applications.
- The memory footprint is only 6-15kb which makes it a more adaptable small powered microcontroller.
- <https://github.com/aws/amazon-freertos>



# IoT Operating Systems

## Embedded Linux

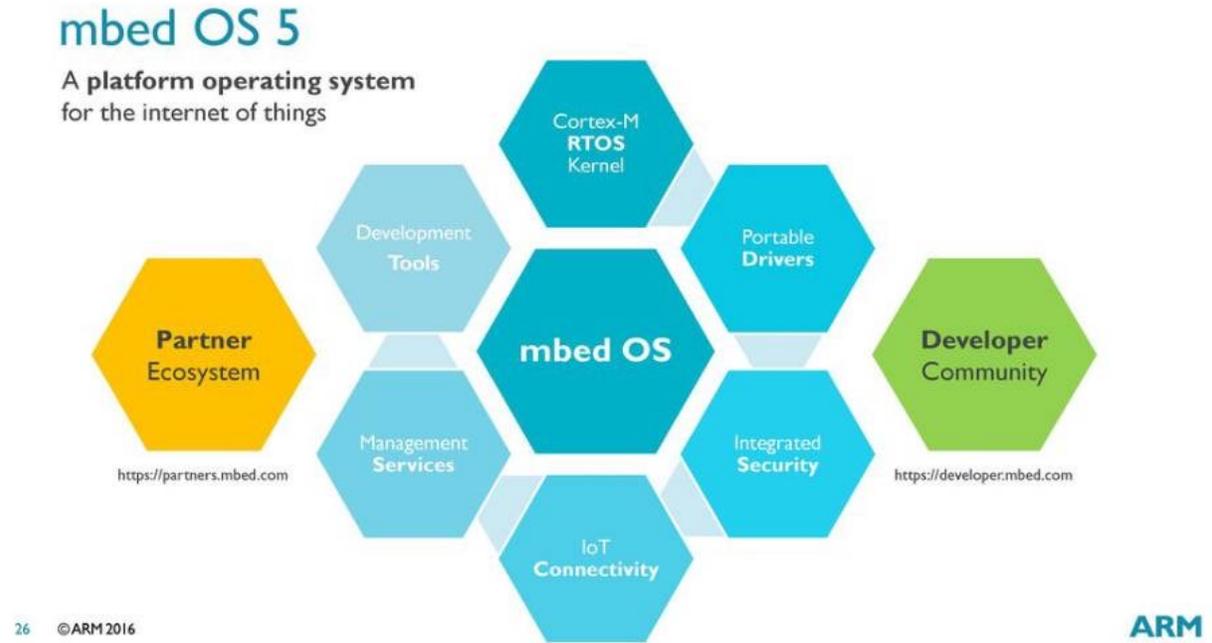
- Embedded Linux is an operating system built for embedded devices although it uses the Linux kernel.
- Smaller size and power of embedded Linux helps to integrate all requirements of IoT devices.
- It covers only 100kb space in memory which makes it faster and reliable.
- Embedded applications (SQL Lite, Boa, tthttpd, PEG, NANO) supported.
- <https://github.com/fkromer/awesome-embedded-linux>



# IoT Operating Systems

## mbed OS

- Mbed operating system works on an ARM processor.
- It is a free, open-source operating system focusing on IoT projects.
- A significant number of connectivity options include Wifi, Bluetooth, 6LowPan, Ethernet, Cellular, RFID, NFC, Thread, and more.
- <https://github.com/ARMmbed/mbed-os/releases/tag/mbed-os-5.13.4>



# Section Outline

Sensors and transducers

Typical IoT Sensors:

Position

Temperature

Light

Sound

Distance (sonar, lidar)



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

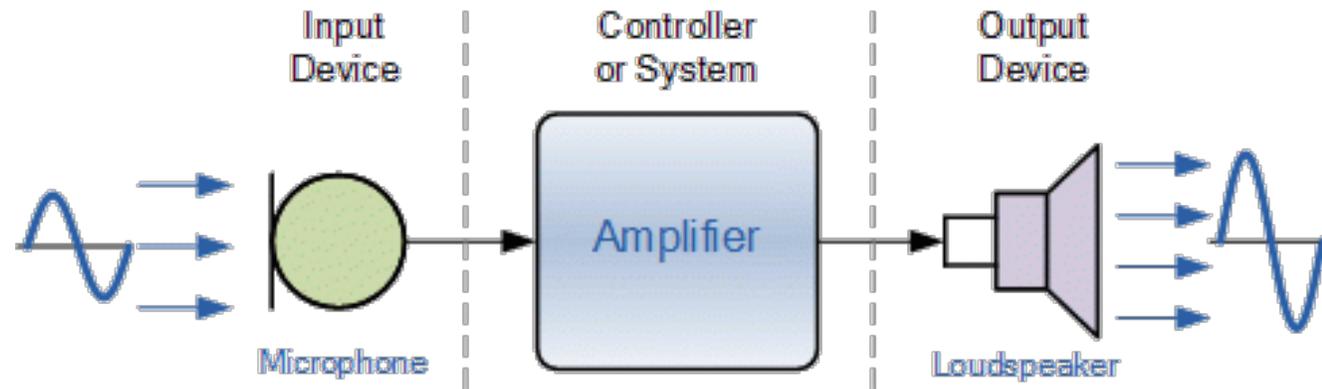
## Section 3

Sensing components and devices

# Sensors

## Introduction

- Sensors and transducers
- Simple stand alone electronic circuits can be made to repeatedly flash a light or play a musical note.
- In order for an electronic circuit or system to perform any useful task or function it needs to be able to communicate with the “real world”
  - by reading an input signal from an “ON/OFF” switch or
  - by activating some form of output device to illuminate a single light



See [https://www.electronics-tutorials.ws/io/io\\_1.html](https://www.electronics-tutorials.ws/io/io_1.html) for a detailed survey

# Sensors

## Introduction

- **Sensors** can be used to sense a wide range of different energy forms such as movement, electrical signals, radiant energy, thermal or magnetic energy etc.
- **Actuators** can be used to switch voltages or currents.
- Devices which perform an “Input” function are commonly called **Sensors** because they “sense” a physical change in some characteristic that changes in response to some excitation, for example heat or force and convert that into an electrical signal.
- Devices which perform an “Output” function are generally called **Actuators** and are used to control some external device, for example movement or sound.

# Sensors

## Introduction

- Electrical **Transducers** are used to convert energy of one kind into energy of another kind, so for example, a microphone (input device) converts sound waves into electrical signals for the amplifier to amplify (a process), and a loudspeaker (output device) converts these electrical signals back into sound waves.

# Sensors

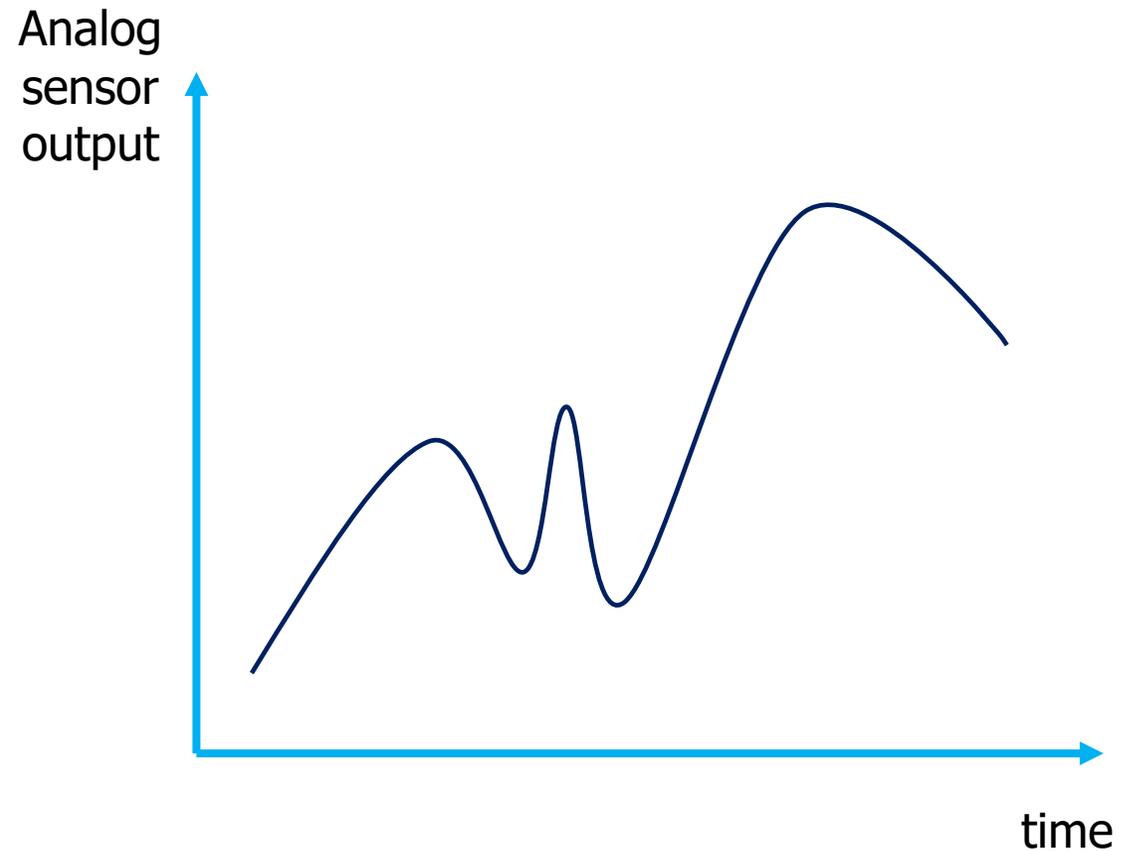
## Common Sensors and Transducers

Quantity being Measured	Input Device (Sensor)	Output Device (Actuator)
Light Level	Light Dependant Resistor (LDR) Photodiode Photo-transistor Solar Cell	Lights & Lamps LED's & Displays Fibre Optics
Temperature	Thermocouple Thermistor Thermostat Resistive Temperature Detectors	Heater Fan
Force/Pressure	Strain Gauge Pressure Switch Load Cells	Lifts & Jacks Electromagnet Vibration
Position	Potentiometer Encoders Reflective/Slotted Opto-switch LVDT	Motor Solenoid Panel Meters
Speed	Tacho-generator Reflective/Slotted Opto-coupler Doppler Effect Sensors	AC and DC Motors Stepper Motor Brake
Sound	Carbon Microphone Piezo-electric Crystal	Bell Buzzer Loudspeaker

# Sensors

## Analog Sensors

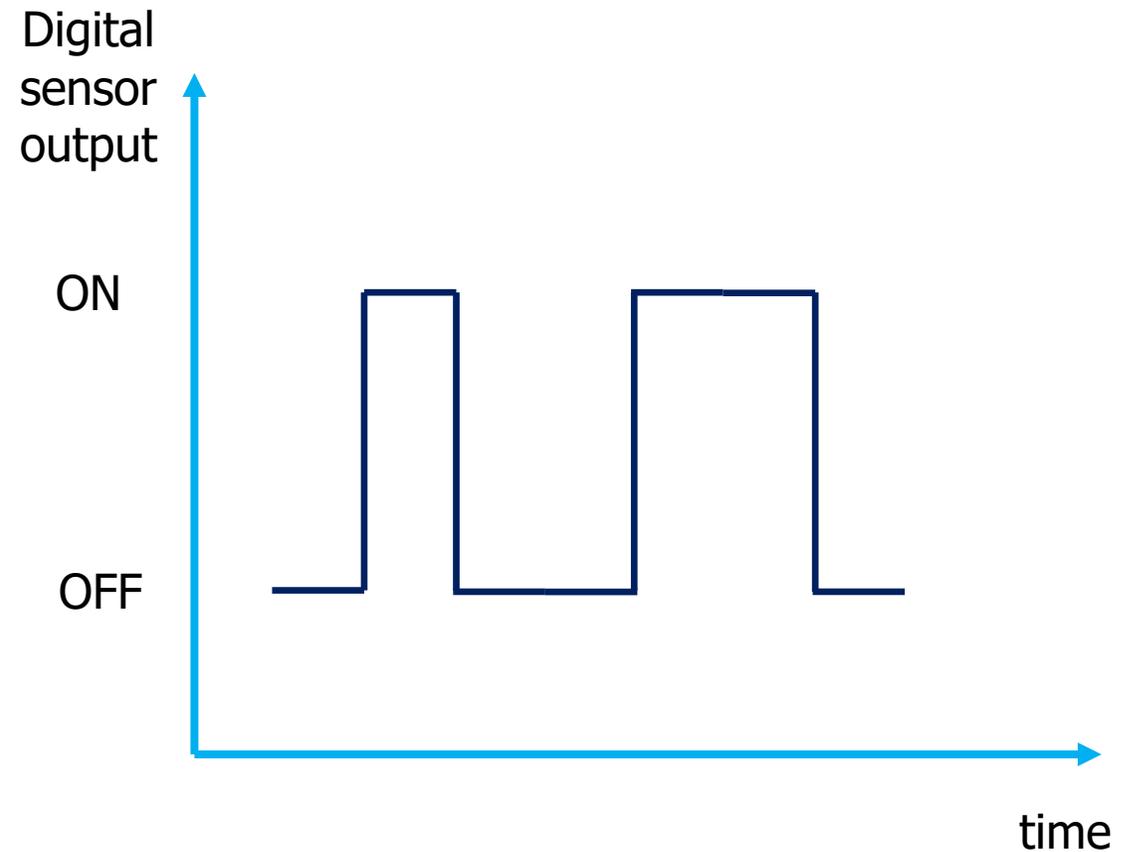
- **Analog Sensors** produce a continuous output signal or voltage which is generally proportional to the quantity being measured.
- Physical quantities such as Temperature, Speed, Pressure, Displacement, Strain etc are all analogue quantities as they tend to be continuous in nature



# Sensors

## Digital Sensors

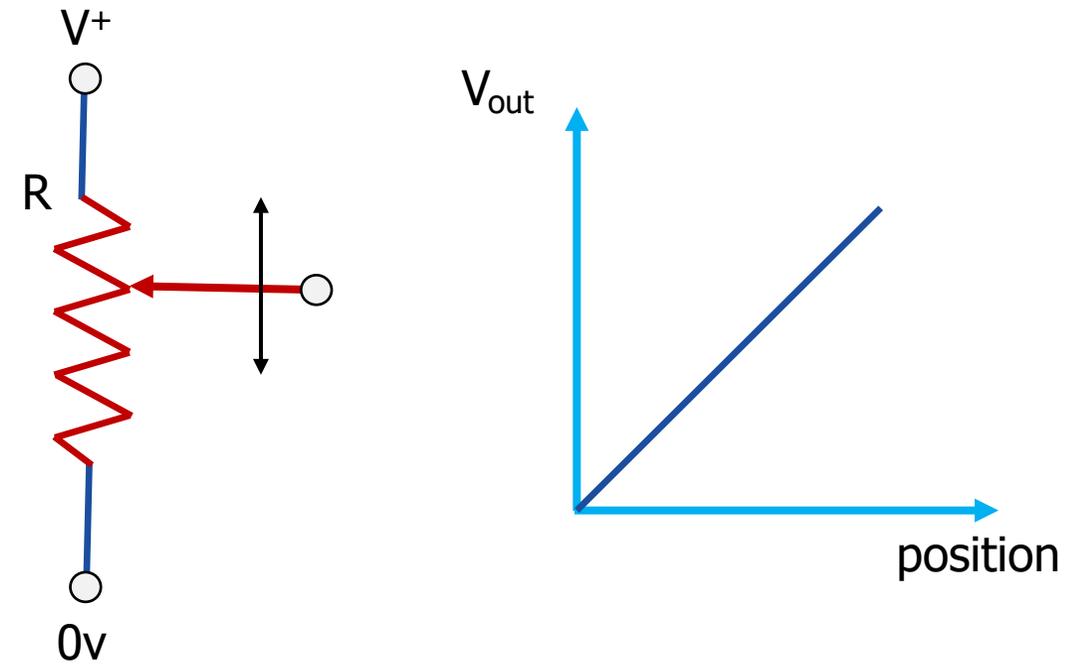
- **Digital Sensors** produce a discrete digital output signals or voltages that are a digital representation of the quantity being measured.
- Digital sensors produce a Binary output signal in the form of a logic "1" or a logic "0", ("ON" or "OFF").
- This means then that a digital signal only produces discrete (non-continuous) values which may be outputted as a single "bit", (serial transmission) or by combining the bits to produce a single "byte" output (parallel transmission).



# Sensor

## Position Sensors

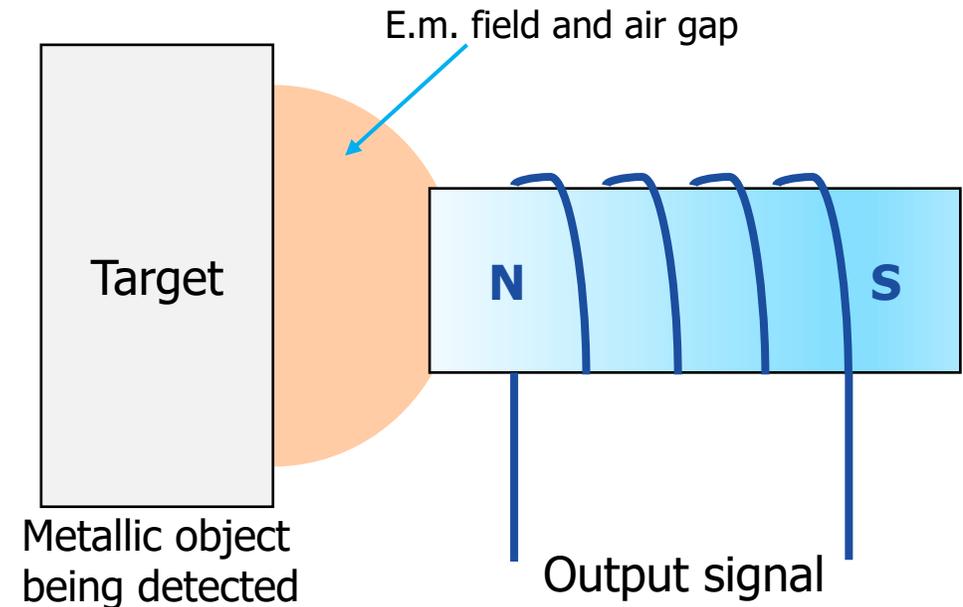
- The most commonly used of all the “Position Sensors”, is the *potentiometer* because it is an inexpensive and easy to use position sensor.
- It has a wiper contact linked to a mechanical shaft that can be either angular (rotational) or linear (slider type) in its movement, and which causes the resistance value between the wiper/slider and the two end connections to change giving an electrical signal output that has a proportional relationship between the actual wiper position on the resistive track and its resistance value.
- Resistance is proportional to position.



# Sensors

## Position Sensors

- An inductive proximity sensor has four main components; The *oscillator* which produces the electromagnetic field, the *coil* which generates the magnetic field, the *detection circuit* which detects any change in the field when an object enters it and the *output circuit* which produces the output signal, either with normally closed (NC) or normally open (NO) contacts.
- Inductive proximity sensors allow for the detection of metallic objects in front of the sensor head without any physical contact of the object itself being detected.



# Sensors

## Temperature Sensors

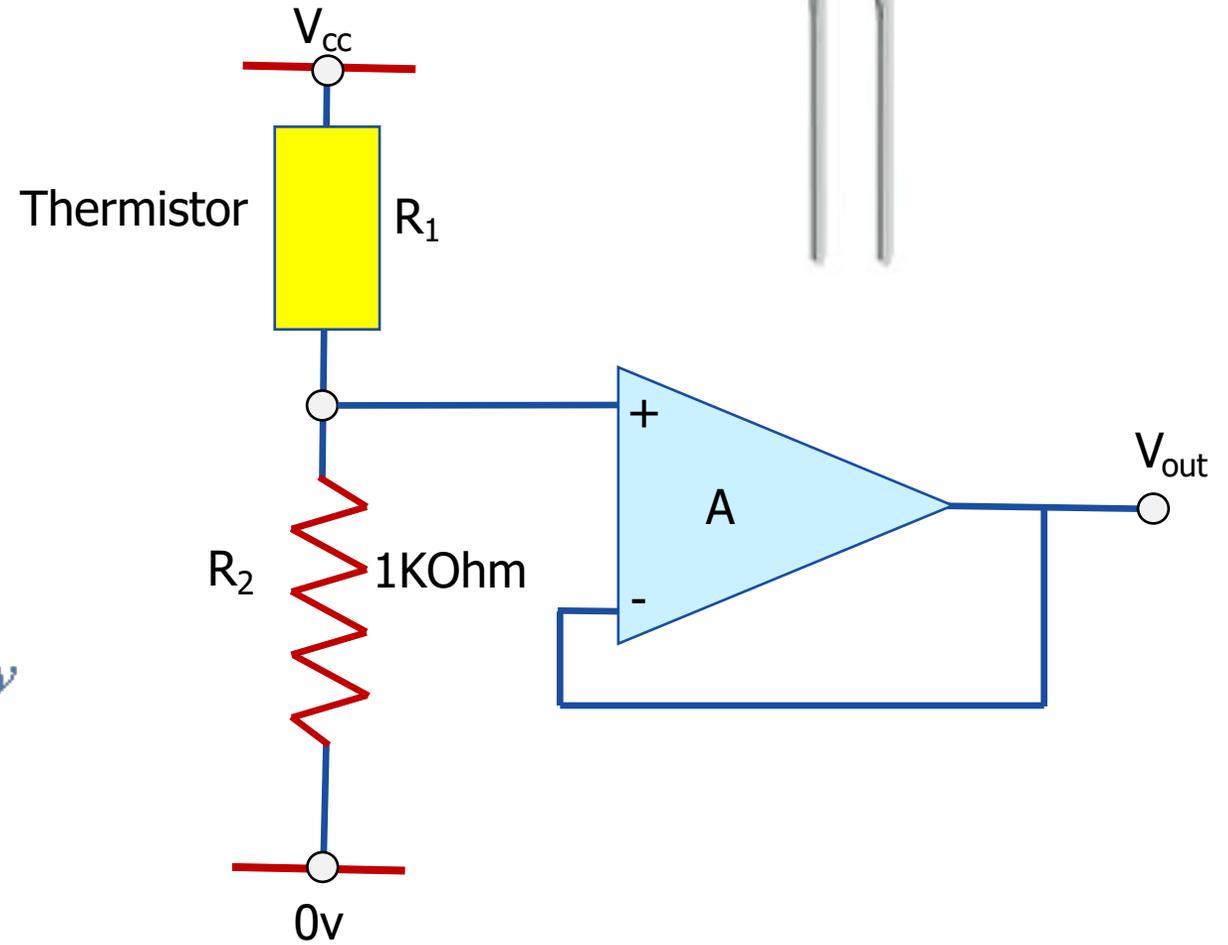
- A thermistor is a special type of resistor which changes its physical resistance when exposed to changes in temperature.

At 25 degrees C:

$$V_{out} = \frac{R_2}{R_1 + R_2} \times V = \frac{1000}{10000 + 1000} \times 12v = 1.09v$$

At 100 degrees C:

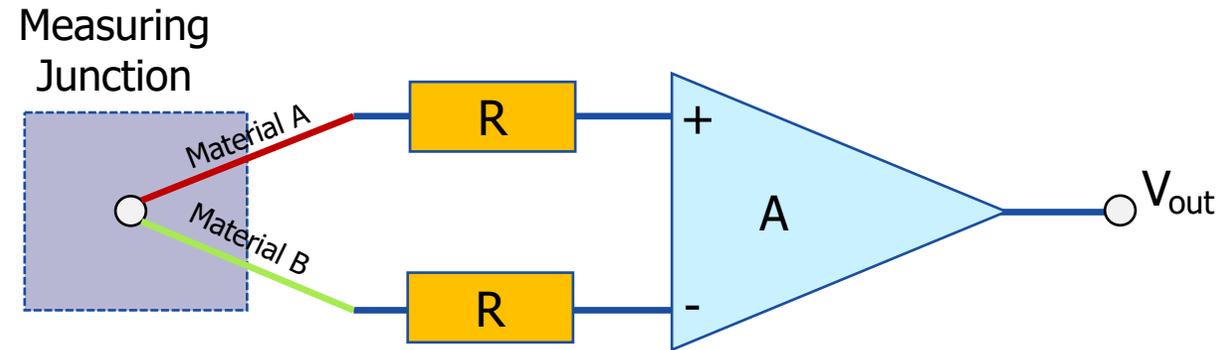
$$V_{out} = \frac{R_2}{R_1 + R_2} \times V = \frac{1000}{100 + 1000} \times 12v = 10.9v$$



# Sensors

## Termocouple

- The operating principal of a thermocouple is very simple and basic.
- When fused together the junction of the two dissimilar metals such as copper and constantan produces a “thermo-electric” effect which gives a constant potential difference of only a few millivolts (mV) between them.



# Sensors

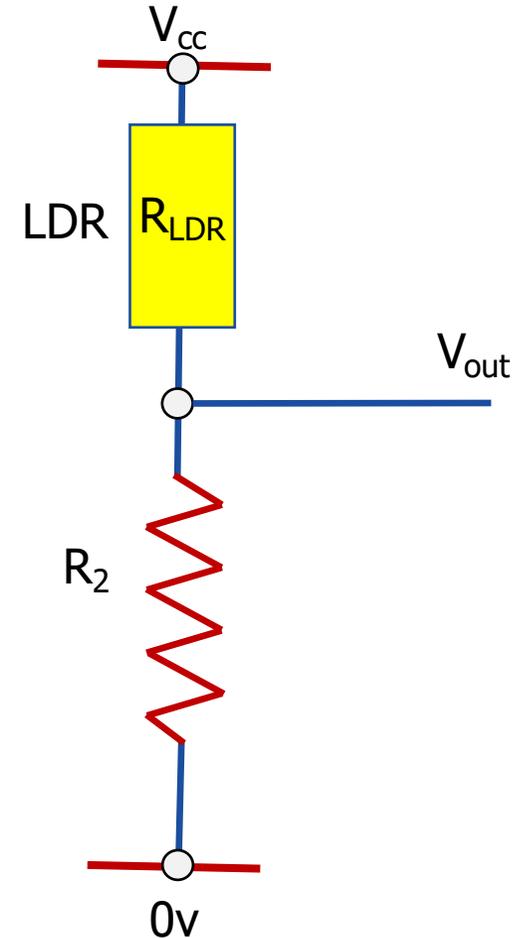
## Light Sensors

- Light Sensors are photoelectric devices that convert light energy (photons) whether visible or infra-red light into an electrical (electrons) signal.

Light Dependant Resistor (LDR)



$$V_{\text{out}} = V_{\text{cc}} \times R_2 / (R_{\text{LDR}} + R_2)$$



# Sensors

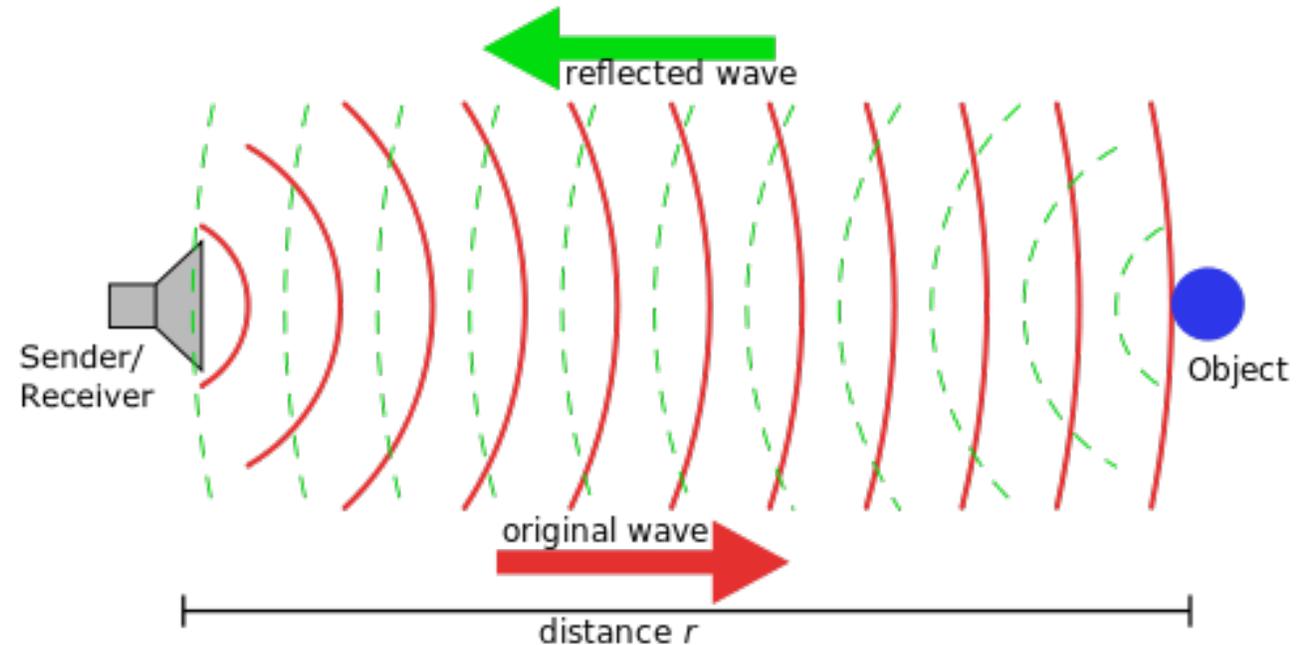
## Sound Sensor

- The **Microphone** is a sound transducer that can be classed as a “sound sensor”.
- It produces an electrical analogue output signal which is proportional to the “acoustic” sound wave acting upon its flexible diaphragm.

# Sensors

## Sonar

- Ultrasonic sensors work by emitting sound waves with a frequency that is too high for a human to hear.
- These sound waves travel through the air with the speed of sound, roughly 343 m/s.
- If there is an object in front of the sensor, the sound waves get reflected back and the receiver of the ultrasonic sensor detects them.
- By measuring how much time passed between sending and receiving the sound waves, the distance between the sensor and the object can be calculated.



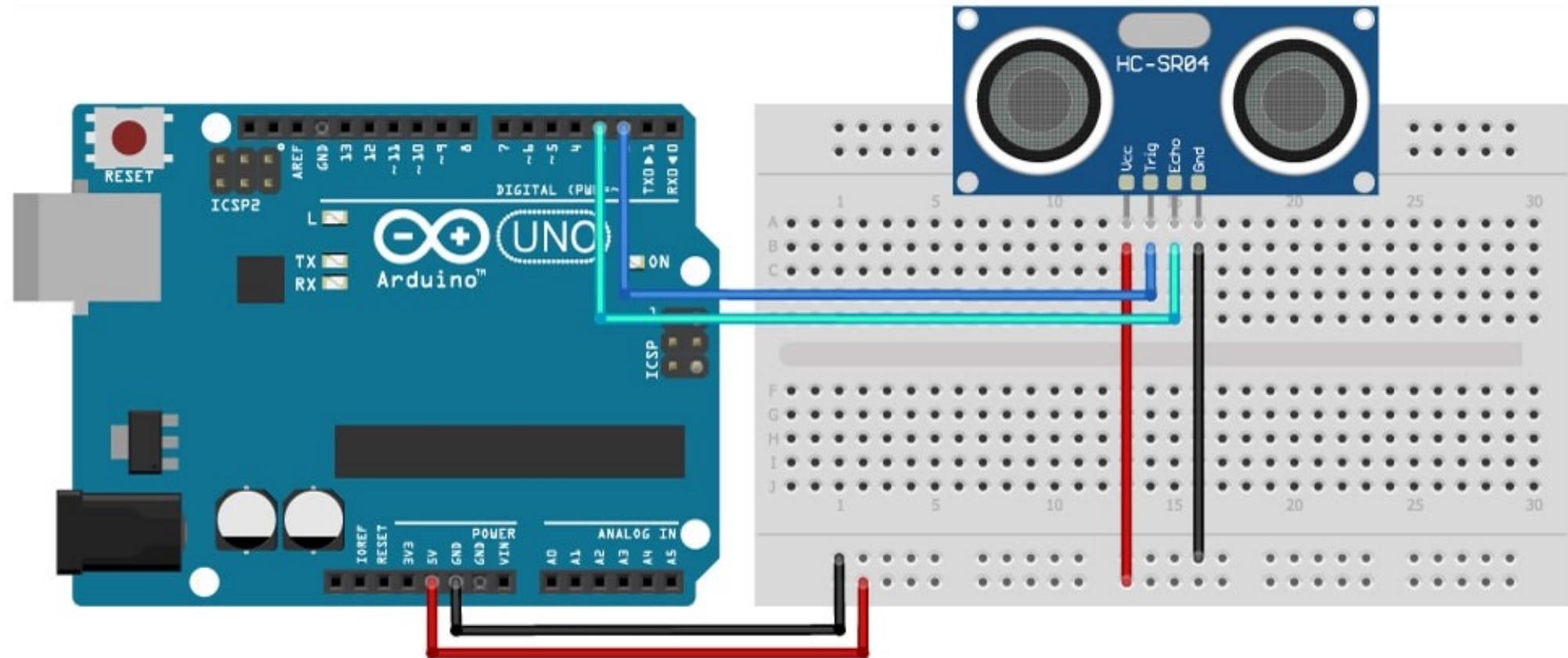
$$\text{Distance (cm)} = \text{Speed of sound (cm}/\mu\text{s)} \times \text{Time } (\mu\text{s}) / 2$$

E.g.:

$$\text{Distance (cm)} = 0.0343 \text{ (cm}/\mu\text{s)} \times 2000 \text{ } (\mu\text{s}) / 2 = 34.3$$

# Sensors

## Sonar: Wiring Example



fritzing

For a complete tutorial, visit <https://www.makerguides.com/hc-sr04-arduino-tutorial/>

# Sensors

## Lidar

- LiDAR, or light detection ranging (sometimes also referred to as active laser scanning) is one remote sensing method that can be used to map structure including vegetation height, density and other characteristics across a region.
- LiDAR directly measures the height and density of vegetation on the ground making it an ideal tool for scientists studying vegetation over large areas.

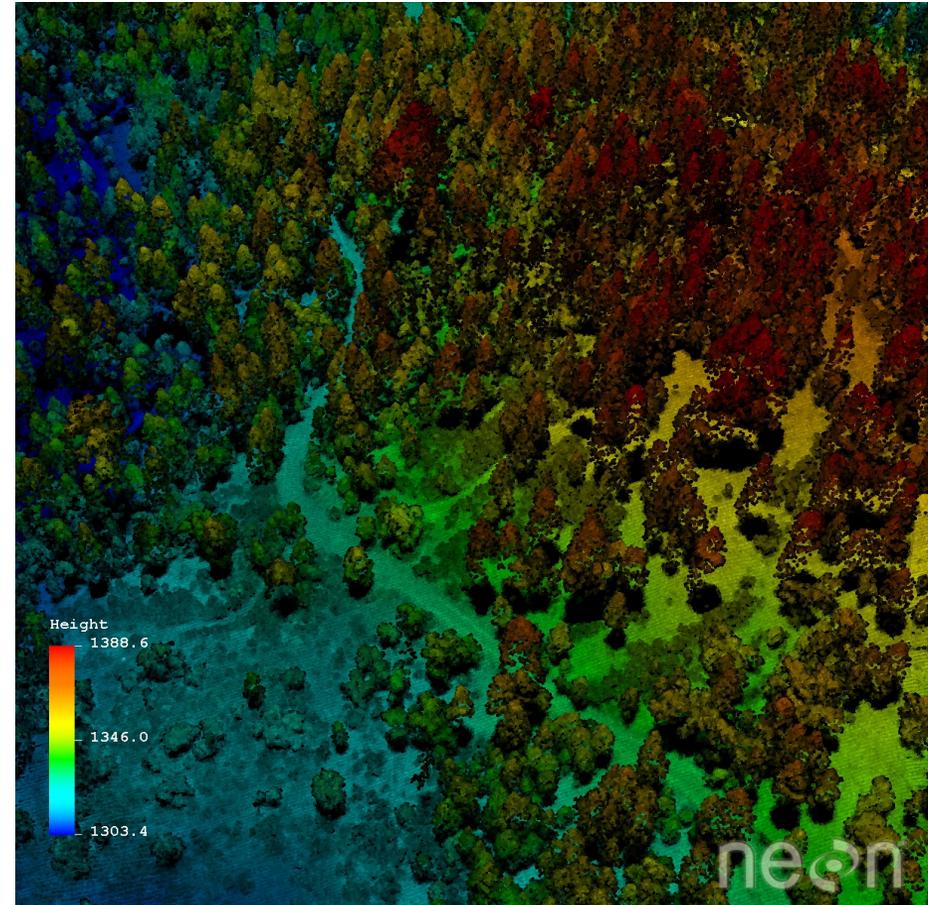


Image and tutorial at <https://www.neonscience.org/lidar-basics>

# Section Outline

Actuators:

Electrical Relay

DC Motor

DC Servo Motor

Loudspeaker



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

## Section 4

Actuators

# Actuators

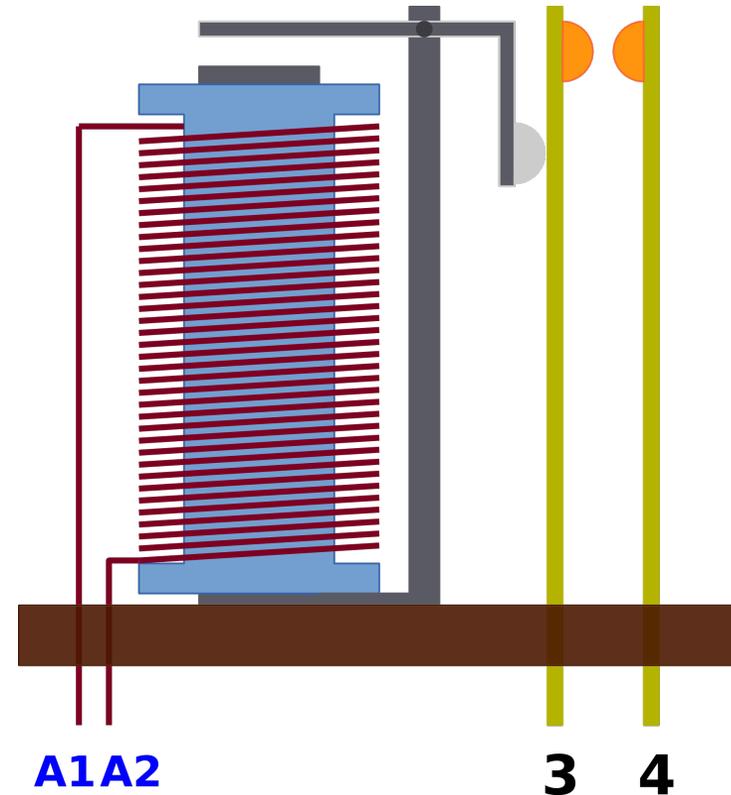
## Introduction

- Actuators convert an electrical signal into a corresponding physical quantity such as movement, force, sound etc.
- An actuator is also classified as a transducer because it changes one type of physical quantity into another and is usually activated or operated by a low voltage command signal.
- Actuators can be classed as either binary or continuous devices based upon the number of stable states their output has.

# Actuators

## Electrical Relay

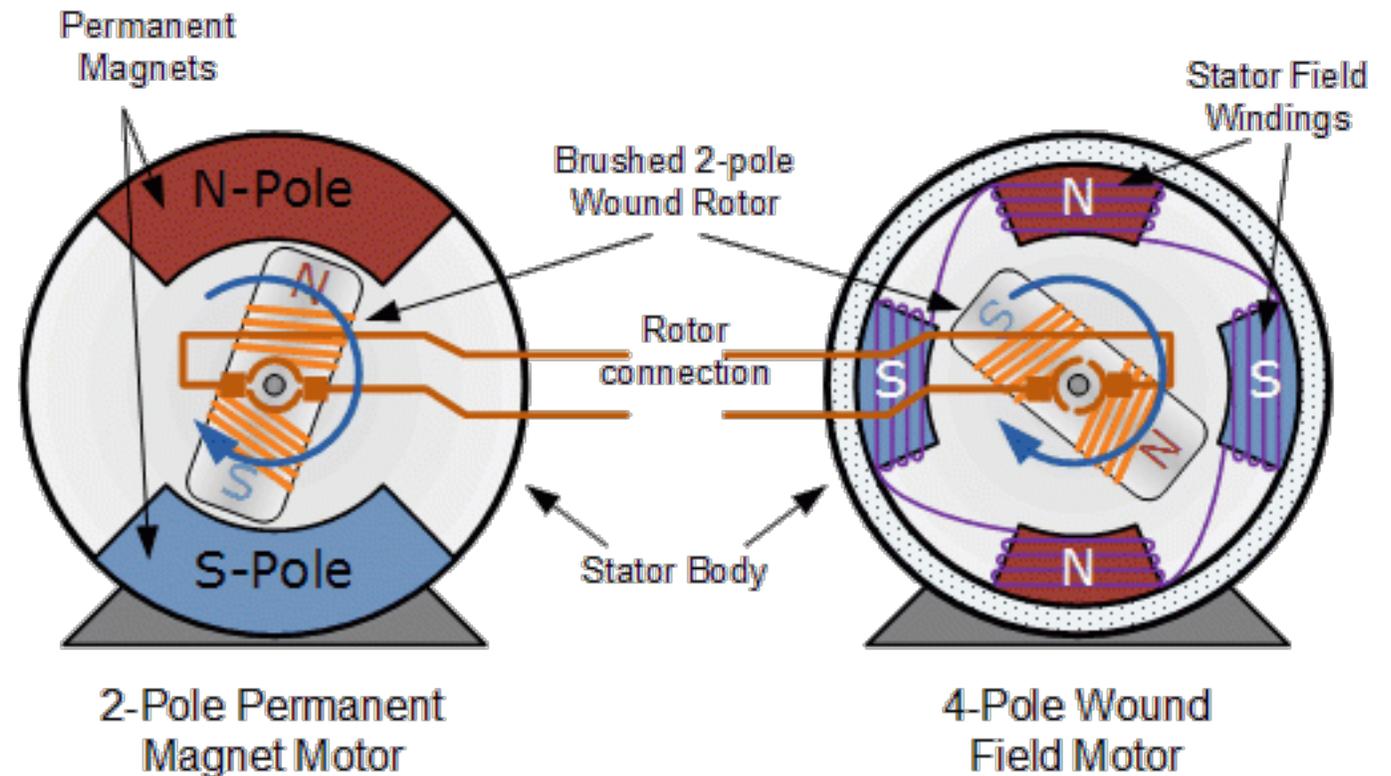
- Electrical relays and contactors use a low level control signal to switch a higher voltage or current supply using a number of different contact arrangements



# Actuators

## DC Motors

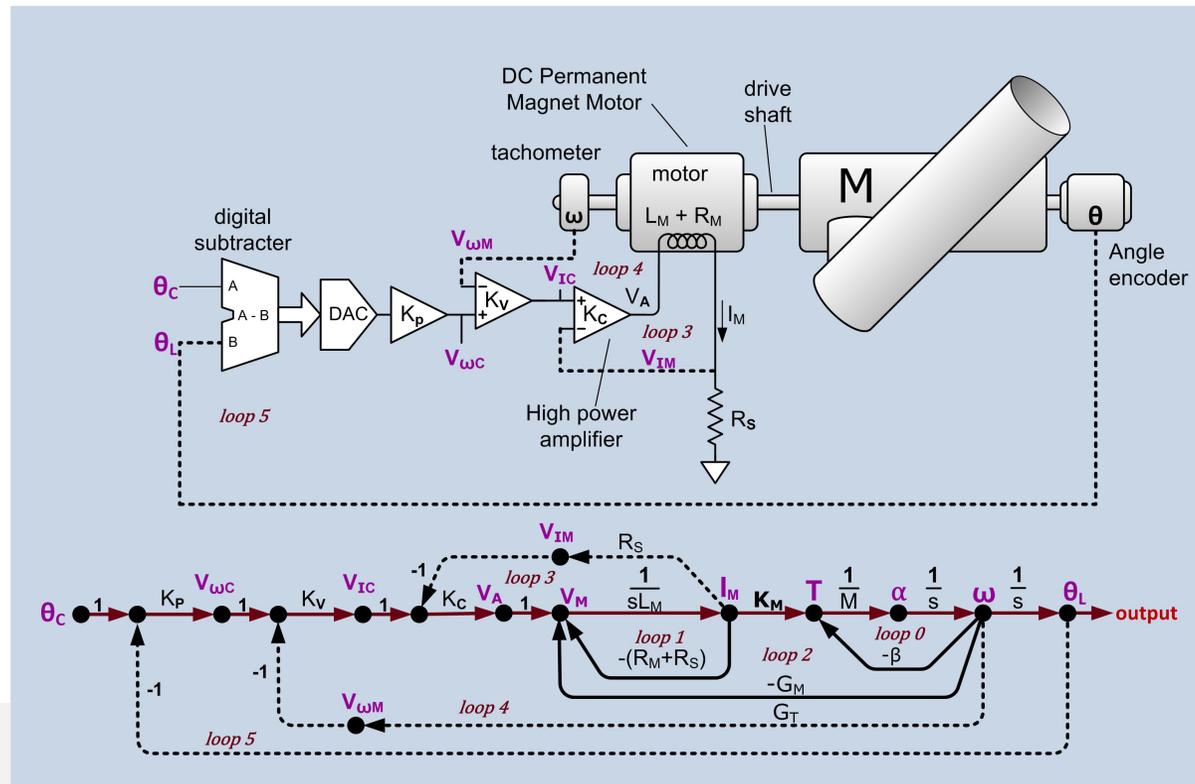
- DC Motors are electromechanical devices which use the interaction of magnetic fields and conductors to convert the electrical energy into rotary mechanical energy.



# Actuators

## DC Servo Motor

- A servo motor consists of a DC motor, reduction gearbox, positional feedback device and some form of error correction.
- The speed or position is controlled in relation to a positional input signal or reference signal applied to the device.



# Actuators

## Loudspeaker

- This type of sound transducer can be classed as a pressure generating output device.





# Thank You

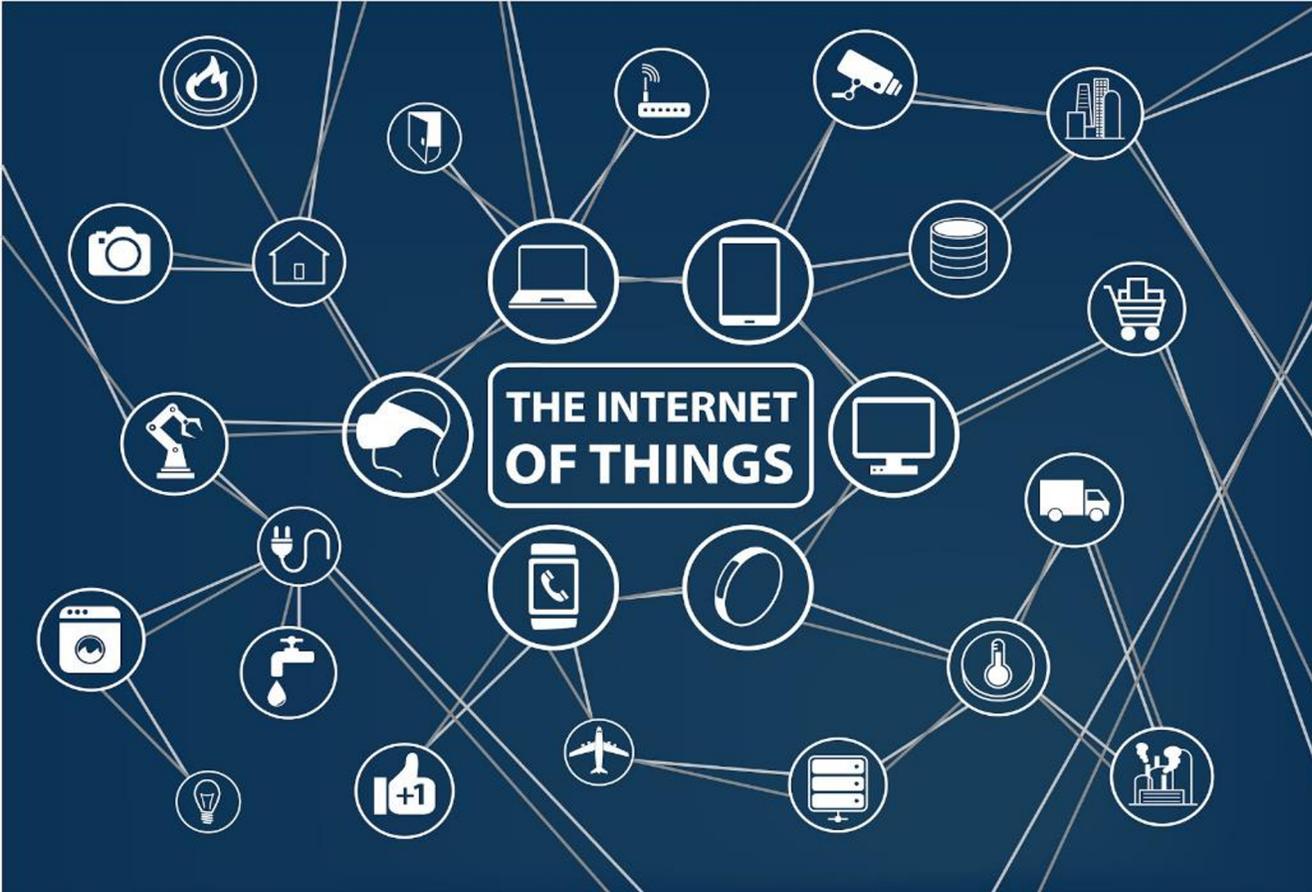
 Prof. Fabrizio Granelli

 +39 0461 282062

 [fabrizio.granelli@unitn.it](mailto:fabrizio.granelli@unitn.it)

 <http://www.unitn.it>

**PROCEED**  
Introducing Recent Electrical Engineering  
Developments Into Undergraduate Curriculum



This Photo by Unknown Author is licensed under [CC BY-NC-ND](https://creativecommons.org/licenses/by-nc-nd/4.0/)

*This publication was produced with the financial support of the European Union. Its contents are the sole responsibility of the partners of IREEDER project and do not necessarily reflect the views of the European Union*



Co-funded by the  
Erasmus+ Programme  
of the European Union

**Felipe Gil Castiñeira**  
**Cristina López Bravo**  
**René Lastra Cid**

# 7. IoT Connectivity Technologies

---

**IREEDER**  
Introducing Recent Electrical Engineering  
Developments into undergraduate curriculum

# This week's topics...

- Introduction
  - Technologies for connectivity
- Short range wireless connectivity options for IoT (Local Area)
  - Wi-Fi
  - Bluetooth
  - ZigBee
- Cellular connectivity (Wide Area)
  - LoRa, Sigfox
  - 3GPP (NB-IoT)
- Wireless Sensor Networks
  - Introduction: multi-hop networks
  - 6LowPAN

# Technologies for connectivity

An essential part of IoT is the connectivity that allows devices to “talk” among them and with backend services.



## Section I

### Introduction

---

# Connectivity

One of the foundations for IoT

- There is an overwhelming number of options for IoT connectivity
- Requirements:
  - Wireless (simpler installation, reconfigurability, mobility, etc.)
  - Trade-off
    - *Power consumption*
    - *Range*
    - *Bandwidth*
    - *QoS requirements*

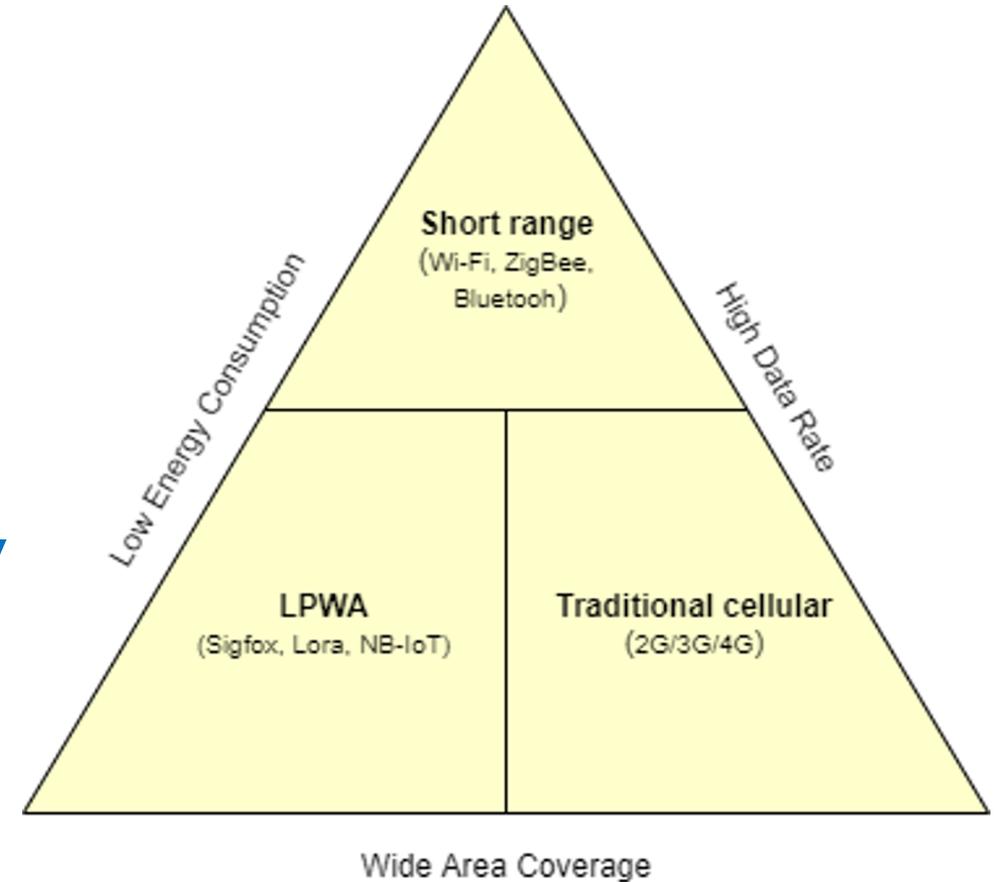


Figure 1: Wireless communications design constraints (Source: Telenor)

# Spectrum and range

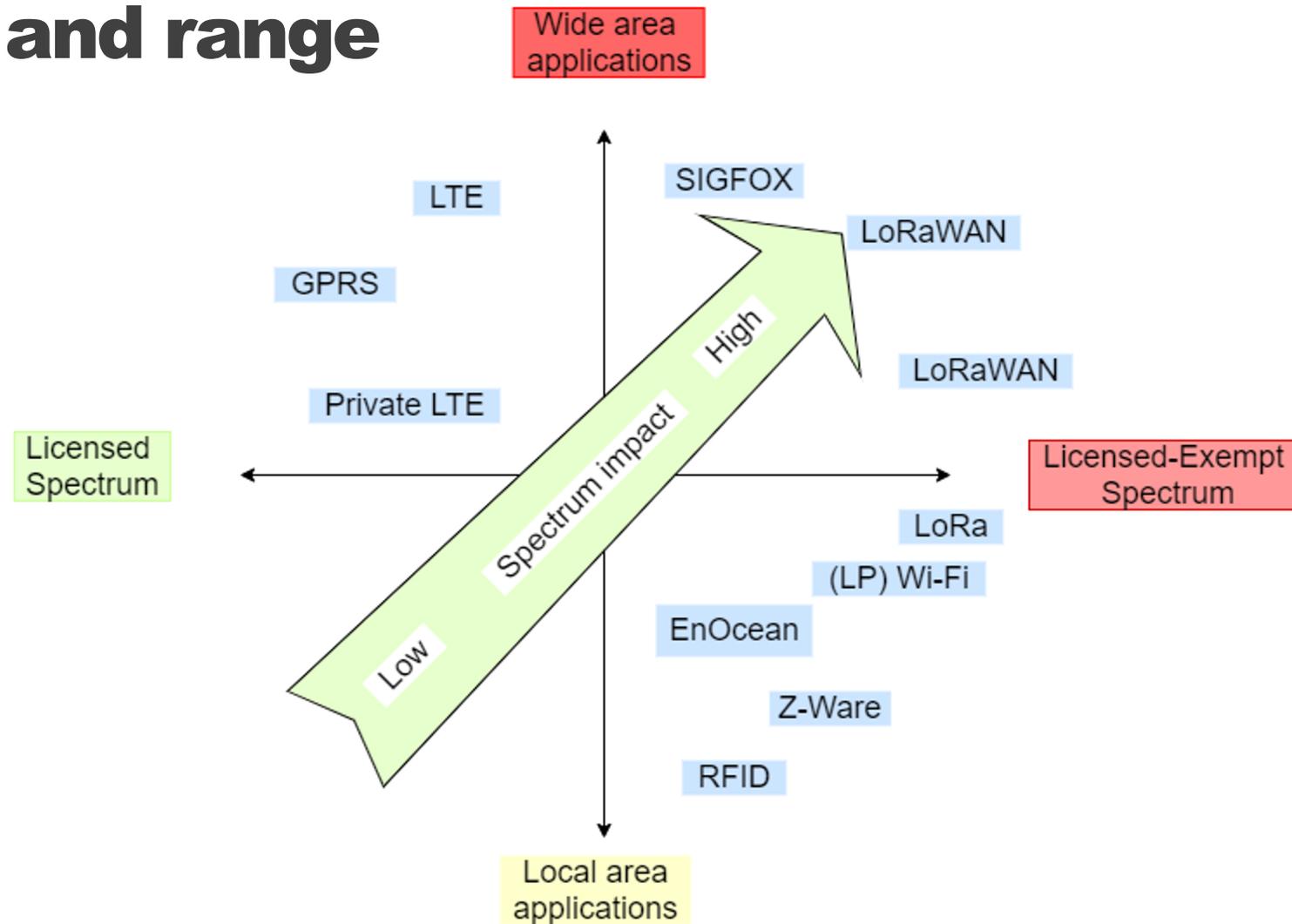


Figure 2: Spectrum impact and coverage for connectivity technologies for IoT (Source: ITU)

# Communications stack

APPLICATION LAYER		AMQP, CoAP, DDS, MQTT, XMPP, REST, etc.
NETWORK	ENCAPSULATION	6LowPAN, Thread
	ROUTING	CARP, RPL, DSR, AODV, etc.
DATALINK		Bluetooth / BLE, Wi-Fi / Wi-Fi HaLow, LoRaWAN, SigFox, Z-Wave, ZigBee, USB, LTE/5G

Table 1: Communications stacks

# Short range communications

Different personal and local wireless communication technologies can be used to connect IoT devices



## Section II

Short range wireless connectivity options for IoT (Local Area)

---



## Short range wireless connectivity options for IoT (Local Area)

- Wi-Fi

# Wi-Fi

- Along with cellular technology, Wi-Fi is the best-known connectivity protocol and is present in almost every home in the world.
  - First instance: Delivering connectivity "on the road" in airports, hotels, Internet cafes, and shopping malls
    - *The goal was to provide web browsing, email and, for business users, access to the office network through Virtual Private Network (VPN) applications.*
  - Later, wireless LAN moved firmly into the home and office environment
  - Now, available in many devices: computers, printers, games consoles, media servers, scanners
    - *from devices as small as a smartphone or as large as a screen in an auditorium*
  - Wi-Fi can be used to easily link together IoT devices, as well as connecting them to wireless access points that in turn connect to cloud-base systems.

# Wi-Fi: IEEE 802.11 Standard

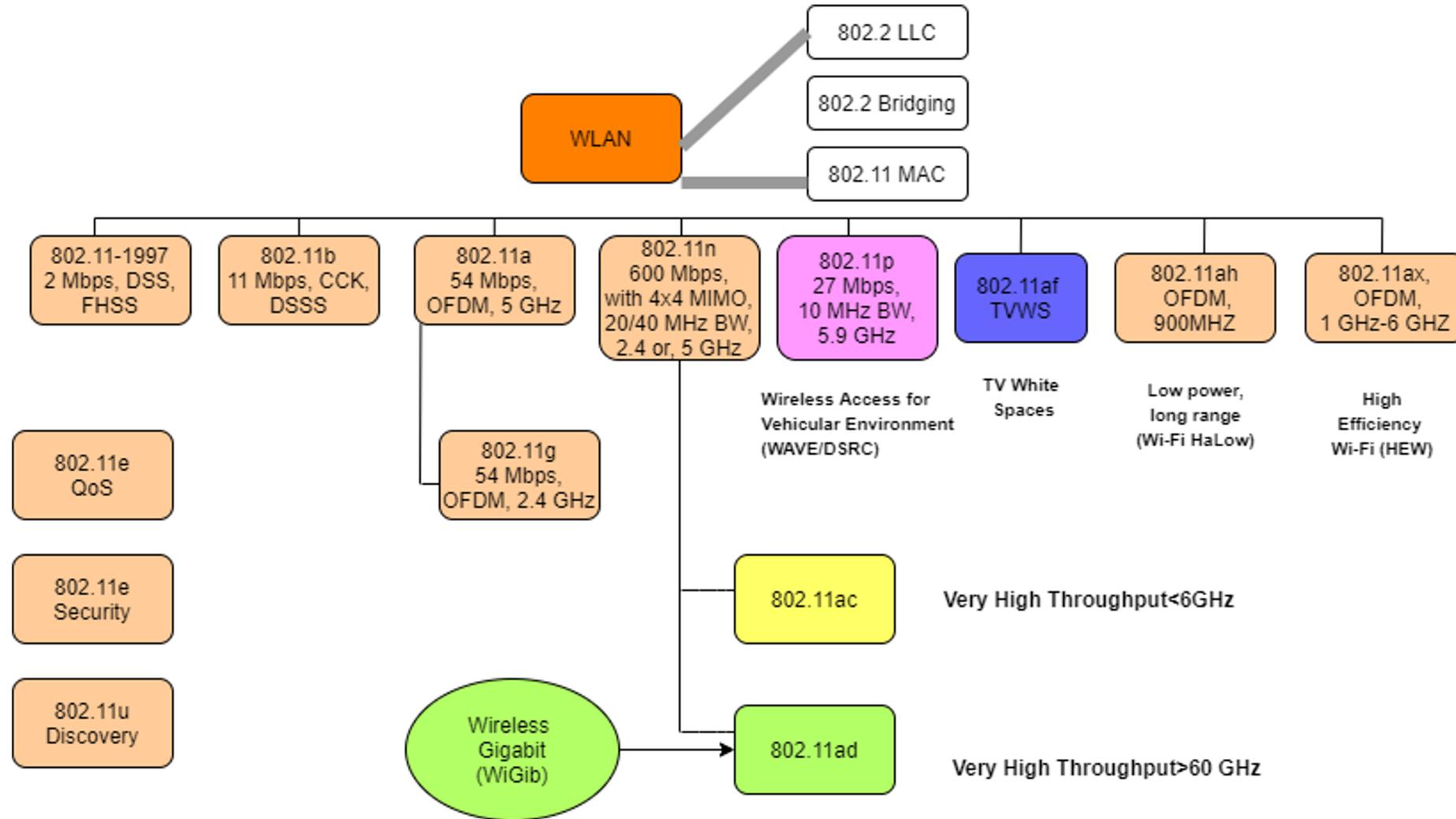


Figure 3: 802.11 Family of standards

# Wi-Fi: Architecture

## Point-to-point (ad hoc) network

- An important feature of WLANs is that they can be used independently of wired networks.
- WLAN can be used as stand-alone networks anywhere to link multiple computers together without having to build or extend wired networks.
- In peer-to-peer topology, client devices within a cell communicate directly to each other
  - Nodes must be in the same transmission range
  - To reach further, forwarding of data is required
  - Nodes are more complex since they need to incorporate management, forwarding and routing functions
- Different IBSS are possible by spatial separation or by using different carrier frequencies

# Wi-Fi: Architecture

## Infrastructure network

- AP-based technology uses access points to bridge traffic onto a wired backbone.
- AP enables a wireless client device to communicate with any other wired or wireless device on the network.
- Each AP manages communications within its range
  - Medium Access Control functions
  - Mobility Management functions
  - Authentication functions
    - *This way wireless devices can be equipped with minimal functionality*

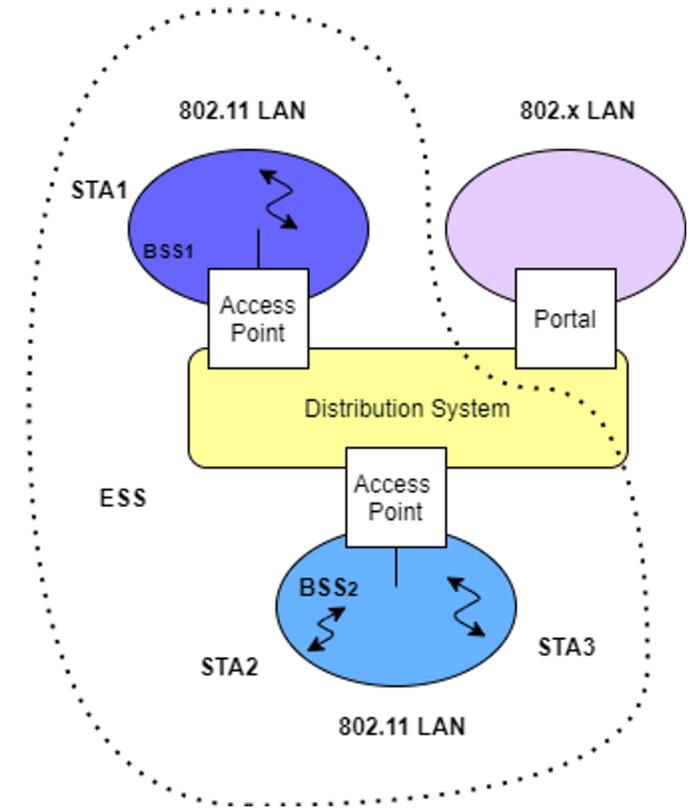


Figure 4: Wi-Fi infrastructure network architecture

Adapted from: Jochen H. Schiller [www.jochenschiller.de](http://www.jochenschiller.de)

# Wi-Fi: Architecture

## Mesh Network

- **PROS**
  - *Reduced dependence on wired networks*
  - *Reduced installation time*
  - *Easily expandable*
- **CONS**
  - *Might reduce performance*
  - *Somewhat difficult to maintain*
  - *Electrical power dependency*

# 802.11 - Layers and functions

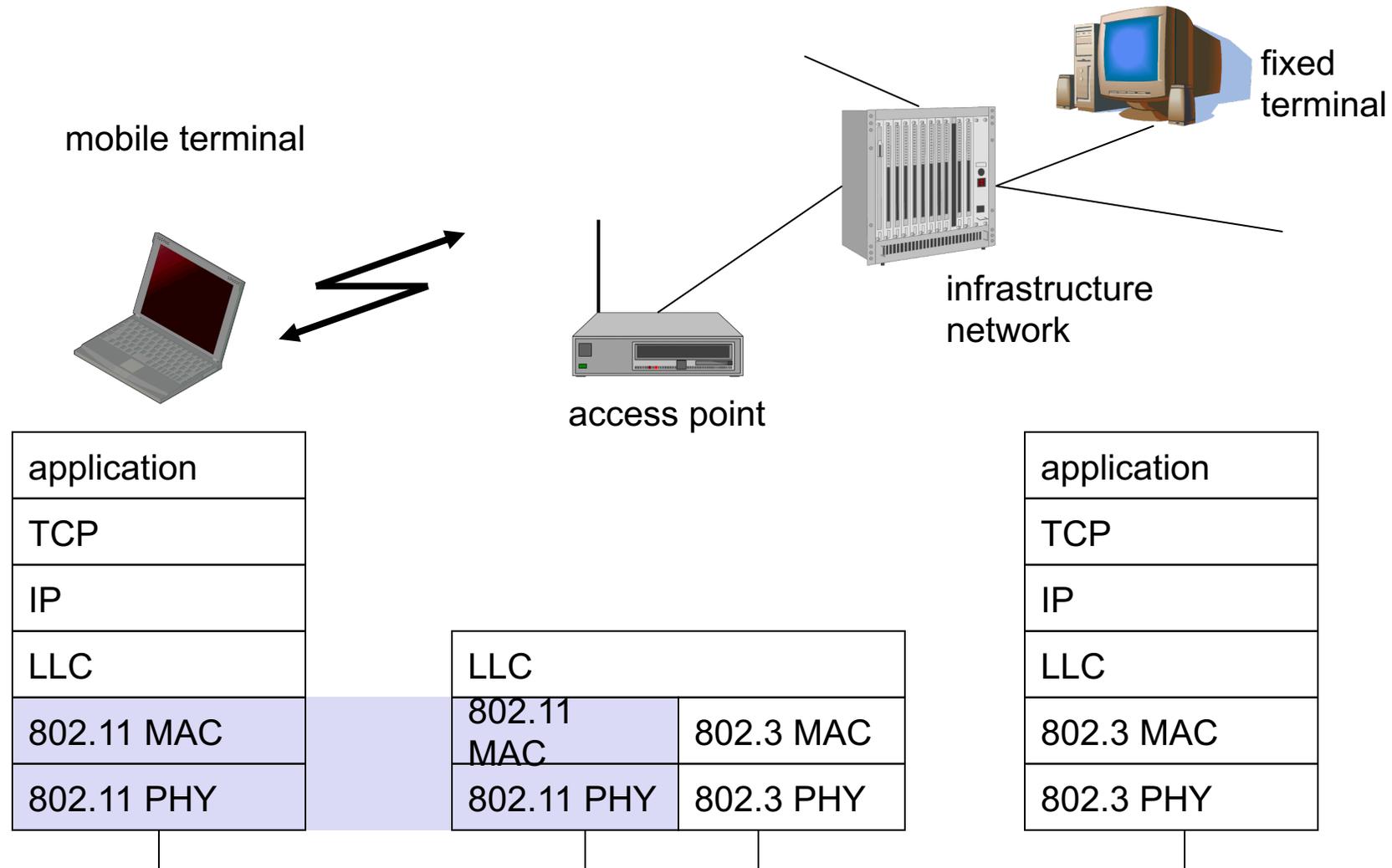


Figure 5: 802.11 protocol architecture

Source: Jochen H. Schiller  
[www.jochenschiller.de](http://www.jochenschiller.de)

# 802.11 - Layers and functions

- MAC
  - access mechanisms, fragmentation, encryption
- MAC Management
  - synchronization, roaming, MIB, power management

- PHY
  - clear channel assessment (carrier sense)
  - modulation, coding
- PHY Management
  - channel selection, MIB
- Station Management
  - coordination of all management functions

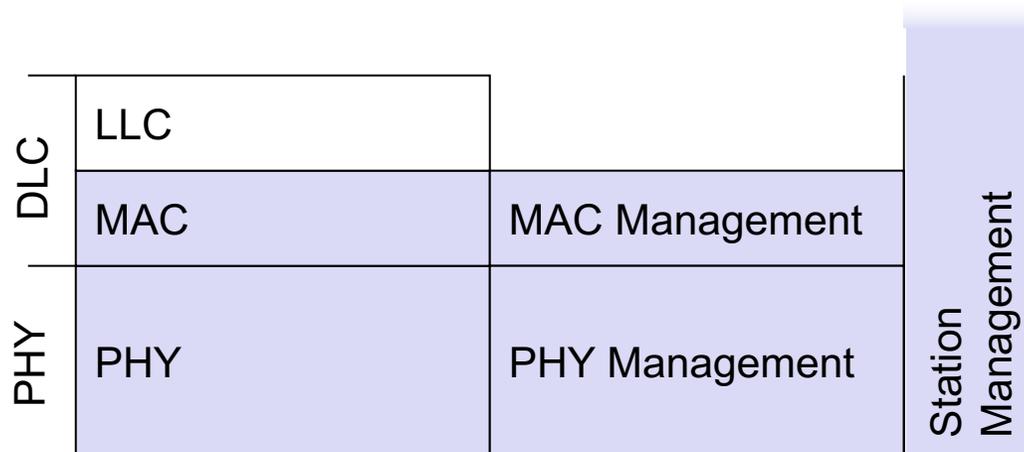


Figure 6: Detail of 802.11 protocol architecture

Source: Jochen H. Schiller  
[www.jochenschiller.de](http://www.jochenschiller.de)

# Wi-Fi HaLow

- Based on the IEEE 802.11ah standard
- Offers the range, data rates, penetration, and low power consumption profiles expected in IoT settings
- Use cases:
  - Industrial automation
  - Logistics and Transportation
  - Agriculture
  - Home and building automation
  - Smart cities

Wi-Fi HaLow for IoT	
Features	Benefits
Sub-1 GHz spectrum operation	Long range: approximately 1 Km
Narrow band OFDM channels	Penetration through walls and other obstacles
Native IP support	Supports coin cell battery devices for months or years
Latest Wi-Fi security	No need for proprietary hubs or gateways

Source: Wi-Fi Alliance  
[www.wi-fi.org/discover-wi-fi/wi-fi-halow](http://www.wi-fi.org/discover-wi-fi/wi-fi-halow)

# Wi-Fi HaLow

Attributes	Values
Frequency	To support product development for worldwide deployment, Wi-Fi Alliance actively advocates for globally harmonized access to spectrum for Wi-Fi HaLow in the 915 MHz to 925 MHz range.
Data Rates	150 Kb - 86.7 Mb (MSC9, 16MHz channels, short guard intervals)
Range (m)	> 1km
Modulation	OFDM over BPSK, QPSK, 16/64/256 QAM
Battery life	Years
Security	WPA3
OTA firmware updates	Support
Subscription required	No
TCP/IP	Support
Network Topology	Star / Relays

Source: Wi-Fi Alliance  
[www.wi-fi.org/discover-wi-fi/wi-fi-halow](http://www.wi-fi.org/discover-wi-fi/wi-fi-halow)

# Wi-Fi HaLow

- New MAC functionality enables devices in a Wi-Fi HaLow network to save energy, reduce congestion, and increase both capacity and device density
  - Extended max idle
    - *Extending the period during which a client device is permitted to sleep before the AP disassociates the client.*
    - *This preserved status allows power sensitive sensors to avoid wasting energy having to reauthenticate if they have otherwise been dropped.*
    - *The feature provides for a maximum idle period greater than five years. In practice, the idle period will depend upon the implementation and application requirements.*

# Wi-Fi HaLow

- Target Wake Time (TWT)
  - *Client devices that expect to sleep for long time periods can negotiate a TWT contract with the AP.*
  - *The AP stores any traffic destined for the client until the agreed upon wake time is reached.*
  - *When the client device wakes at the prescribed time, it listens for its beacon and engages the AP to receive and transmit any data required before returning to its sleep state.*
  - *The interval between wake times, negotiated by the client and AP, can vary from especially short (microseconds) to very long (years).*
- Restricted Access Window (RAW)
  - *For systems with predictable activity periods, an AP can grant a subset of clients with RAW privileges to transfer their data, while others can be forced to sleep, buffer non-urgent data, or both.*
  - *The client devices save power and leave more network capacity available for other time-critical traffic.*
- By combining TWT and RAW functions, a network designer can minimize channel contention and save power throughout the system.



## Wireless connectivity options for IoT

- Bluetooth (WPAN: Wireless Personal Area Network)

# Bluetooth

- Bluetooth is a specification for the use of low-power radio communications to link phones, computers and other network devices over short distances without wires.
  - Universal radio interface for ad-hoc wireless connectivity
  - Interconnecting computer and peripherals, handheld devices, PDAs, cell phones – replacement of IrDA
  - Embedded in other devices – low cost ( $< 1 \$$ )
  - Short range (10 m), low power consumption, license-free, 2.4 GHz ISM
  - Voice and data transmission

# Bluetooth

## Markets, applications and services

### Markets

---

- Automotive
- Beacons & retail
- Consumer electronics
- Health and wellness
- Mobile telephony
- PC & peripherals
- Sport & fitness
- Smart home
- Wearable technology

### Applications and services

---

- Wire replacement
- Data and voice access points
- File Exchange and synchronization
- Ad hoc networks
- Medical applications
- Stock control (industry)
- Marketing
- Authentication

# Bluetooth – Protocol stack

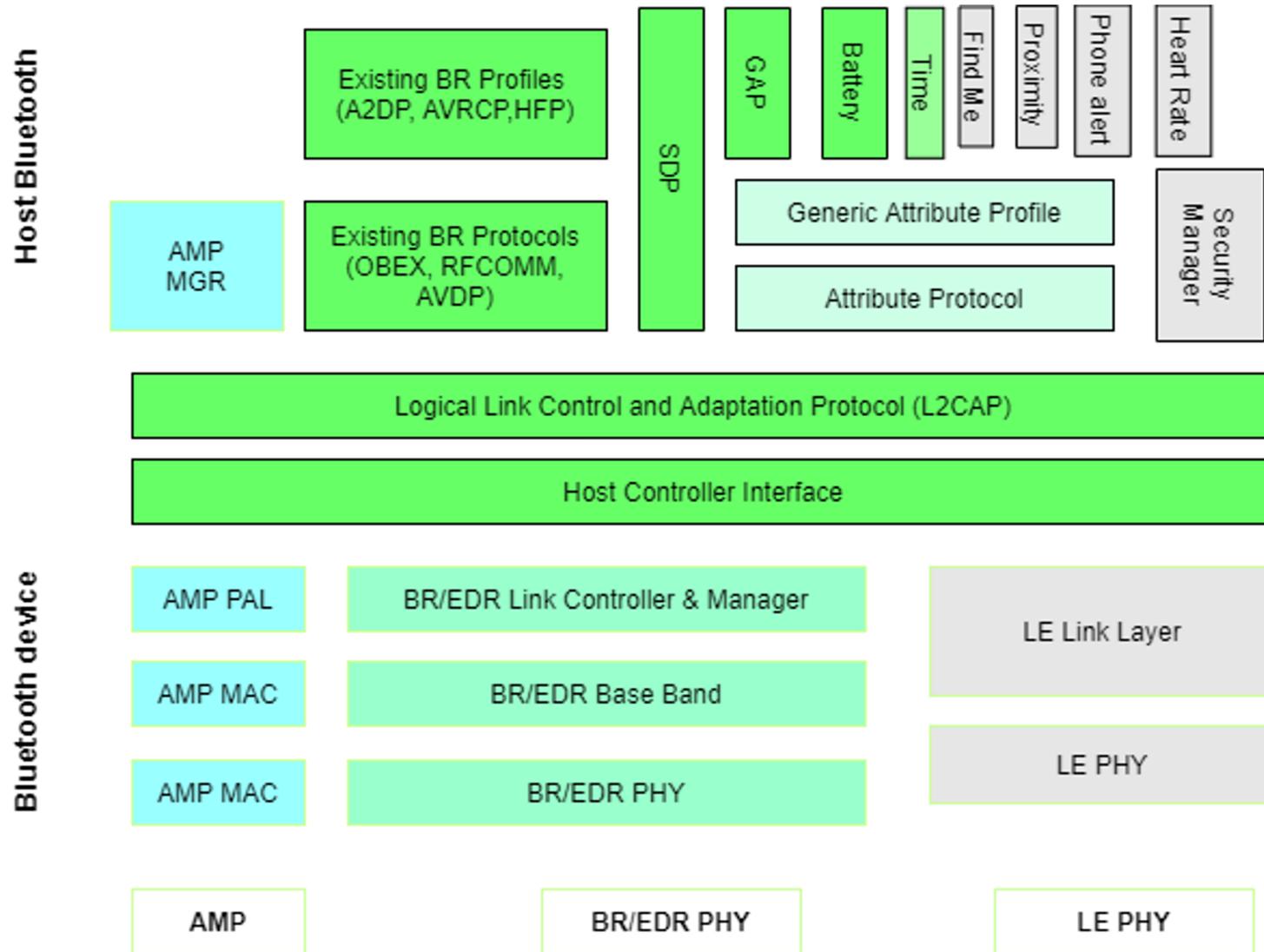


Figure 8: Bluetooth stack

# Bluetooth Smart

Optimized for low power consumption

- Short packets reduce TX peak current
- Short packets reduce RX time
- Less RF channels to improve discovery and connection time
- Simple state machine
- Designed for the transmission of small pieces of data (1 Mbps, but not optimized for data transmission)
- ...

# Bluetooth Smart

## Technical details

Range	~ 150 meters open field
Output Power	~ 10 mW (10dBm)
Max Current	~ 15 mA
Latency	3 ms
Topology	Star
Connections	> 2 billion
Modulation	GFSK @ 2.4 GHz
Robustness	Adaptive Frequency Hopping, 24 bit CRC
Security	128bit AES CCM
Sleep current	~ 1 $\mu$ A
Modes	Broadcast, Connection, Event Data Models, Reads, Writes
Packet size	8-27 bytes

Table 2: Bluetooth LE main technical characteristics

# Bluetooth Smart

## Dual Mode devices

- Bluetooth BR/EDR and LE
- Used anywhere that BR/EDR is used today

## Single Mode devices

- Implements only Bluetooth low energy
- Used in new devices and applications

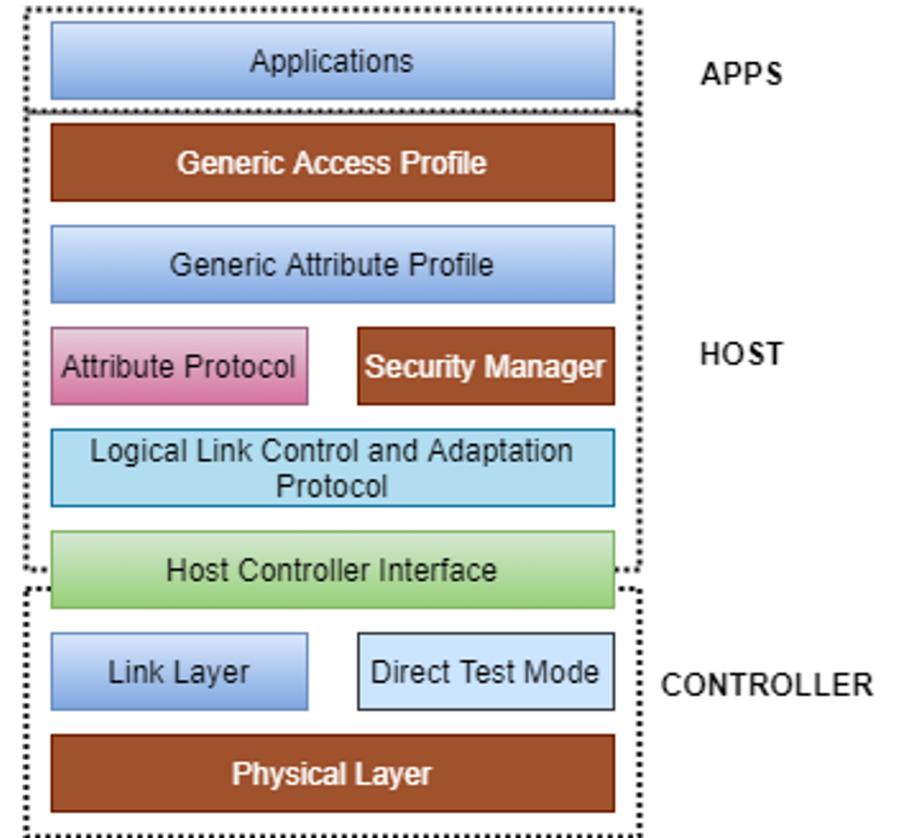


Figure 9: Bluetooth Low Energy Protocol Stack

# Bluetooth Smart

## Physical layer

- 2.4 GHz ISM band
- 1Mbps GFSK
  - *Larger modulation index than Bluetooth (better range)*
- 40 Channels on 2 MHz spacing

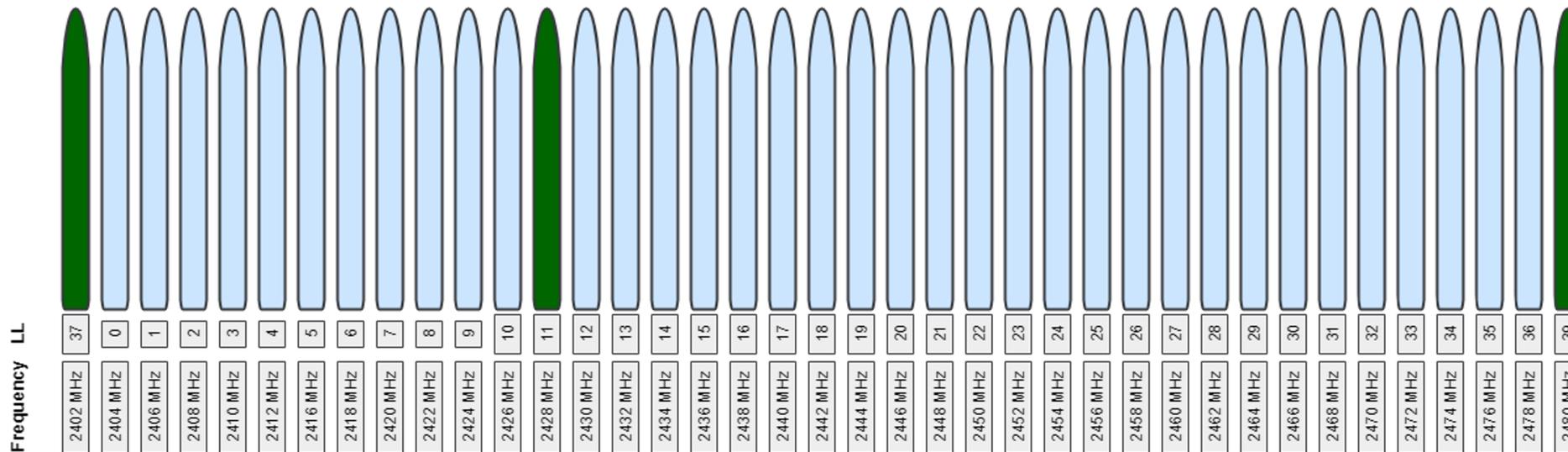
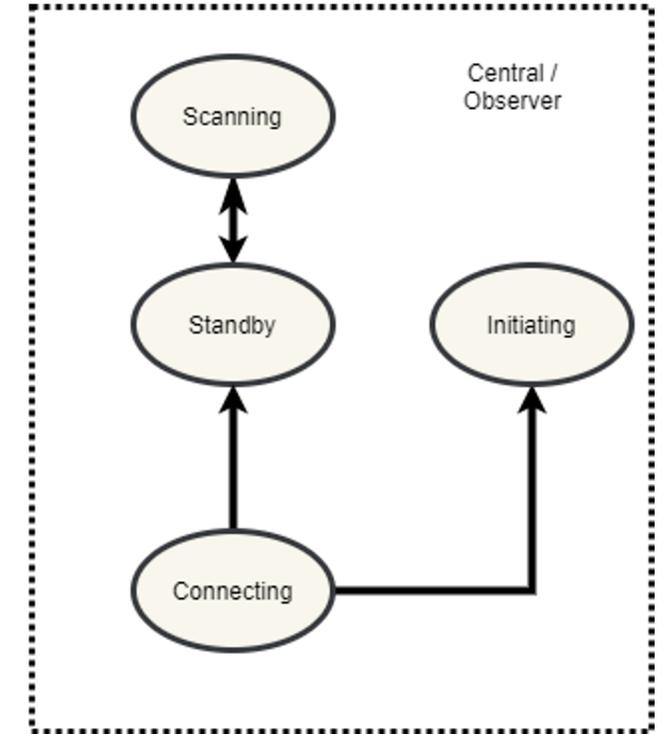
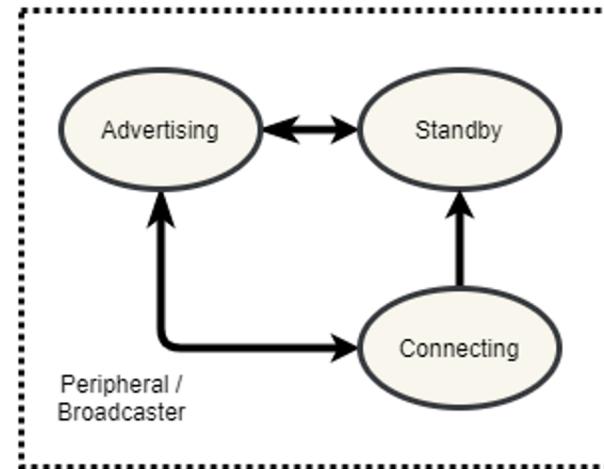
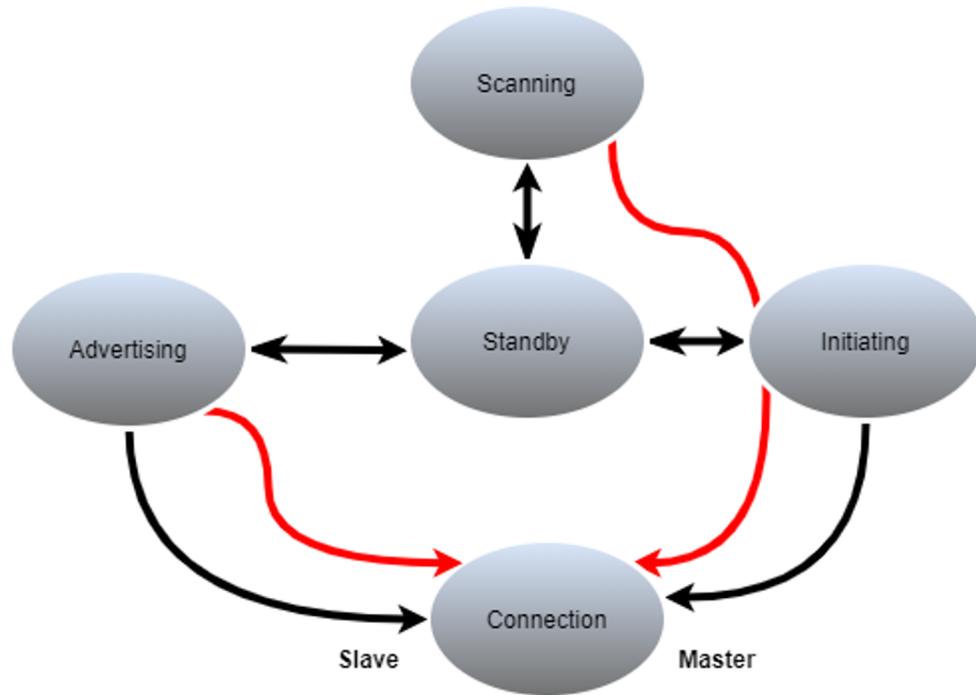


Figure 10: Bluetooth LE channels

Source: *Feasibility Study. Minimum Viable Device to support Internet of Things Realization*

# Bluetooth Smart

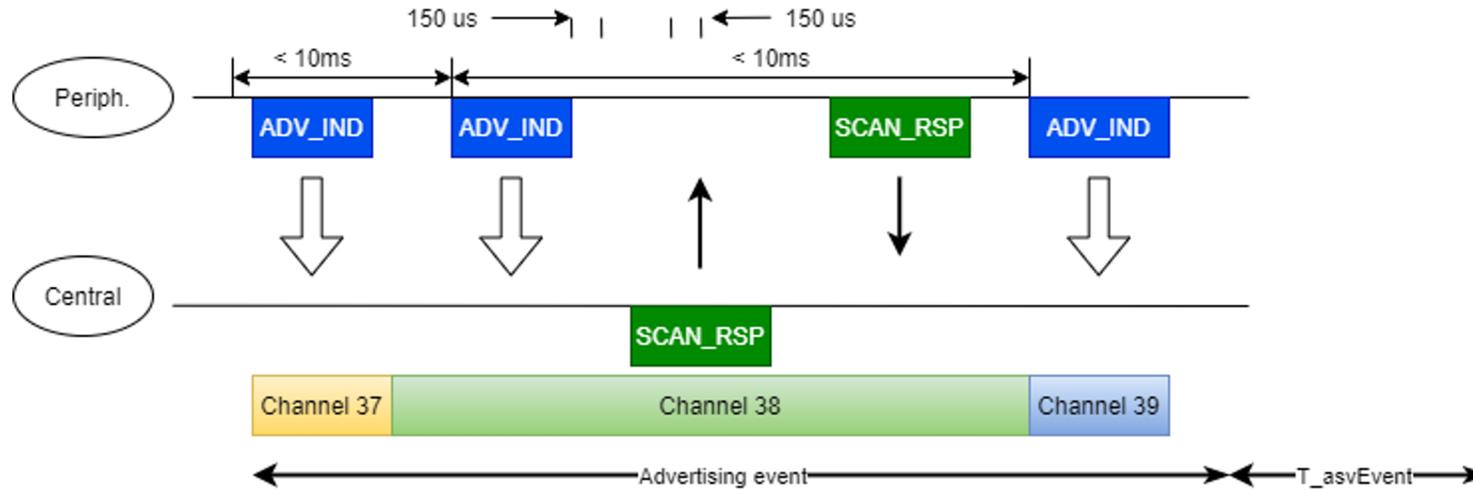
## Link Layer state machine



Source: *Feasibility Study. Minimum Viable Device to support Internet of Things Realization*

Figure 11: Bluetooth LE link layer states

# Bluetooth Smart



Source: *Feasibility Study.*  
*Minimum Viable Device to support*  
*Internet of Things Realization*

Figure 12: Bluetooth LE link layer advertising

Devices can advertise for a variety of reasons:

- To broadcast promiscuously
- To transmit signed data to a previously bonded device
- To advertise their presence to a device wanting to connect
- To reconnect asynchronously due to a local event



# Bluetooth Smart

## Topology

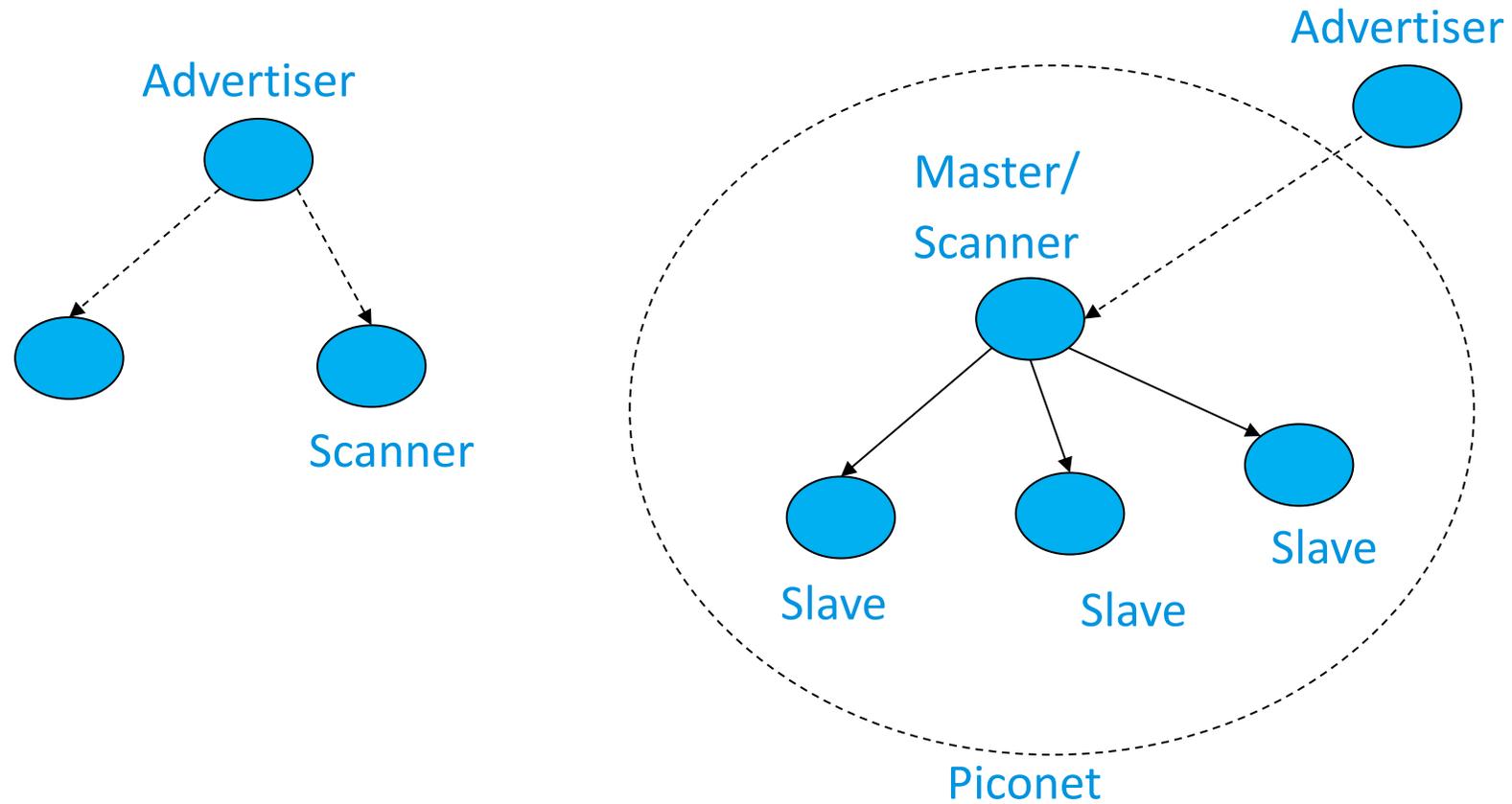


Figure 14: Bluetooth LE Topology

# Bluetooth Smart - ATT

## Attribute Protocol

- The attribute protocol is a simple client/server stateless protocol based on attributes presented by a device
  - *Client: discover attributes, read and write attributes, request data to server*
  - *Server: contains attributes*
- Attributes
  - *Handle: identifier to access the attribute*
  - *UUID: type and nature of the data contained in the value*
  - *Permissions: readable, writable, notify, encryption required, authentication required, authorization required ...*
  - *Value*

# Bluetooth Smart - GATT

## Generic Attribute Profile

- Built on the Attribute Protocol (ATT), adds a hierarchy and data abstraction model on top of it
- Establishes in detail how to exchange all profile and user data over BLE connection
- Client: sends requests to a server and receives responses from it
  - *First: Service discovery*
  - *Reads and writes attributes found in the server*
  - *Receives server-initiated updates*
- Server: responsible of storing and making user data available to the client, organized in attributes

# Bluetooth MESH

Point-to-Point (1:1)	Point-to-Point (1:1)	Broadcast(1:m)	Mesh (m:m)
<i>audio streaming</i>	<i>data transfer</i>	<i>localized information</i>	<i>large device networks</i>
<ul style="list-style-type: none"> <li>• wireless headsets</li> </ul>	<ul style="list-style-type: none"> <li>• sports &amp; fitness devices</li> </ul>	<ul style="list-style-type: none"> <li>• point of interest beacons</li> </ul>	<ul style="list-style-type: none"> <li>• building automation</li> </ul>
<ul style="list-style-type: none"> <li>• wireless speakers</li> </ul>	<ul style="list-style-type: none"> <li>• health &amp; wellness devices</li> </ul>	<ul style="list-style-type: none"> <li>• item finding beacons</li> </ul>	<ul style="list-style-type: none"> <li>• wireless sensor networks</li> </ul>
<ul style="list-style-type: none"> <li>• in-car audio</li> </ul>	<ul style="list-style-type: none"> <li>• peripherals &amp; accessories</li> </ul>	<ul style="list-style-type: none"> <li>• way finding beacons</li> </ul>	<ul style="list-style-type: none"> <li>• asset tracking</li> </ul>

# Bluetooth MESH

## Design

- The mesh network operation is designed to:
  - *enable messages to be sent from one element to one or more elements (many-to-many (m:m) device communications);*
  - *allow messages to be relayed via other nodes to extend the range of communication;*
  - *secure messages against known security attacks, including eavesdropping attacks, man-in-the-middle attacks, replay attacks, trash-can attacks, brute-force key attacks, ...;*
  - *work on existing devices in the market today;*
  - *deliver messages in a timely manner;*
  - *continue to work when one or more devices are moved or stop operating; and*
  - *have built-in forward compatibility to support future versions of the Mesh Profile specification.*

# Bluetooth MESH

## Design

- **The publish/subscribe model**

- *The exchange of data within the mesh network is described as using a publish/subscribe paradigm.*
- *Nodes that generate messages publish the messages to an address, and nodes that are interested in receiving the messages will subscribe to such an address.*
- *This allows for flexible address assignment and group casting.*

# Bluetooth MESH

## Flooding

- Managed-flood-based network
  - *Uses broadcast channels to transmit*
    - Other nodes can **receive** messages and **relay** messages
  - *To restrict the unlimited relaying of messages several methods are used*
    - Heartbeats
    - Time to live
    - Network message cache
    - Subnets

# Bluetooth 5 & MESH

## Use cases

- Building automation
  - *Lighting control*
  - *HVAC*
  - *Access control*
  - *Presence detection*
- Street lighting
- Device Firmware upgrade
  - *Over the air*
- Assets tracking
- Capillary sensor network
  - *Bluetooth mesh*
  - *Gateway*



## Wireless connectivity options for IoT

- ZigBee

# WPAN: ZigBee

- ZigBee is a Wireless communication standard designed by the ZigBee Alliance.
  - *Designed for:*
    - simple implementation
    - low power consumption
  - *Target applications:*
    - Those that require secure communications, low data transmission rates and that maximize battery lifetime
  - *Designed for industrial as well as for residential applications*
    - Sensors and control devices

# WPAN: ZigBee

- ZigBee is defined from a protocol stack that allows a simple and efficient communication among different devices.
- *Low Layers: PHY and MAC, are defined by IEEE 802.15.4, (Low Rate – WPAN) standard.*
- *High Layers: NWK and APS, are defined by ZigBee Alliance.*
  - Network layer (NWK) manages routing tasks and the maintenance of network nodes
  - Application Support Sublayer (APS) establishes an interface between network layer and ZigBee Device Objects (defined by the standard or the manufactures)

# ZigBee

## Applications (1)

- Especially suitable for situations where power consumption and / or implementation costs are critical
- It is designed to be useful in a wide variety of applications:
  - *Industrial control and monitoring*
  - *Hosting embedded sensors*
  - *Collecting medical data*
  - *Public safety*
    - Sensing and location determination at disaster sites

# ZigBee

## Applications (2)

- Precision agriculture
  - *sensing of soil moisture, pesticide, herbicide, or pH levels*
  - *sensing of soil humidity (drip systems, using just the amount of water required to keep plants green)*
- Fleet control
  - *Automotive sensing*
    - Tire pressure monitoring
    - Oil levels
  - *Mileage*
- Smart badges and tags
- Smoke or intruder detection

# ZigBee

## Applications (3)

- Home automation and networking
  - *Network interconnection*
  - *Peripherals connection (transmission rate 115 kbps, latency 15 ms)*
    - Wireless mice, keyboard, joysticks, PDAS, games
  - *Control of consumer electronics (10 kbps, 100 ms)*
    - Radio, televisions, CDs, VCRs, DVDs and so on
    - Truly universal remote control to remote control all of them
  - *Home automation (10 kbps, 100 ms)*
    - heating, ventilation, and air conditioning (HVAC)
    - security
    - lighting
    - control of objects such as curtains, windows, doors, and locks
  - *Health monitoring (10 kbps, 100 ms)*
    - medical monitoring of the elderly people living alone
  - *Interactive toys and games (115 kbps, 15 ms)*

# ZigBee Protocol Stack

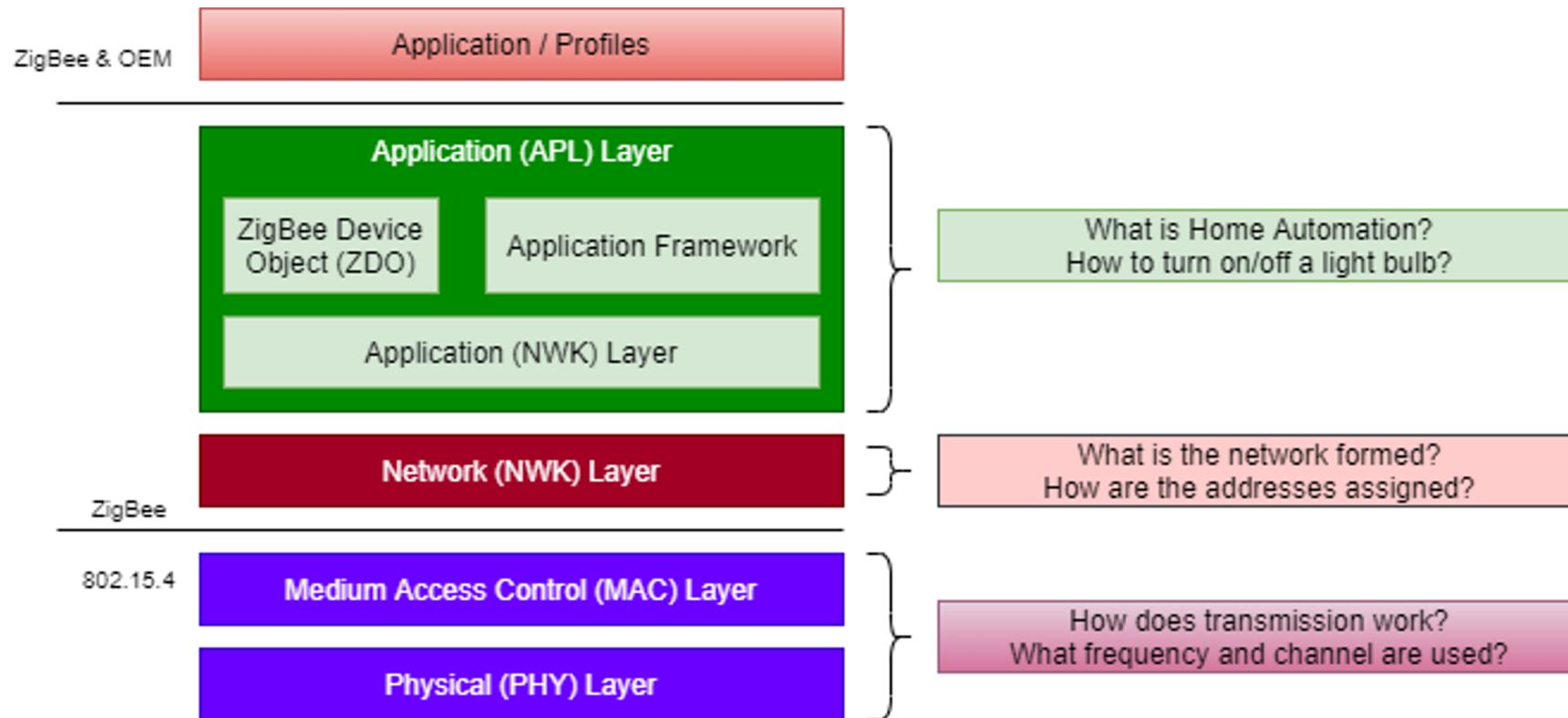


Figure 15: ZigBee stack

Source: XBee/XBee-PRO S2C Zigbee

# ZigBee: PHY Layer (1)

## Frequency Bands & Transmission rates

- The higher rate in the 2.4 GHz PHY is attributed largely to a higher-order modulation scheme, in which each data symbol represents multiple bits.
- The different transmission rates can be exploited to achieve a variety of different goals
  - *The low rate of the 868/915 MHz PHY can be translated into better sensitivity and larger coverage area, thus reducing the number of nodes required to cover a given physical area*
  - *The higher rate of the 2.4 GHz PHY can be used to attain higher throughput, lower latency, or lower duty cycle.*

# ZigBee: PHY Layer (2)

## Channelization

- Multiple wireless networks compete for the same frequency bands, and interferences:
  - *Need to be able to relocate within the spectrum*
- ZigBee can change channel through the crosslayer cooperation between PHY, MAC and NWK
  - *PHY layer: channel assessment and frequency agility:*
    - Receiver energy detection, link quality indication, channel switching
  - *MAC layer:*
    - Scan function
  - *NWK layer:*
    - Establish initial operation channel, change in response to a breakdown or prolonged failure.

# ZigBee: PHY Layer (3)

## Coverage range

- Each device shall be capable of transmitting at least 1 mW
- Typical devices (1 mW) are expected to cover a 10–20 m range
  - *With good sensitivity and a moderate increase in transmit power, a star network topology can cover the application scenario.*
  - *For applications allowing more latency, mesh network topologies can cover larger areas*
- 2,4 GHz band
  - *10 m indoor*
  - *200 m outdoor*
- 868/915 MHz band
  - *30 m indoor*
  - *1000 m outdoor*

# ZigBee: Network Layer (1)

## Device roles (1)

- To simplify some devices, they can be classified according to their capabilities:
  - **Reduced Function Device (RFD)**
    - *Suitable for simple applications (light switches and infrared sensors) that do not need to send or receive large amounts of data*
    - *Reduced stack size*
    - *Limited to being "leafs" in star topology*
    - *Talks only to Full Function Devices (FFD)*
  - **Full Function Device (FFD)**
    - *Capable of being network coordinator, link coordinator or a simple communication node*
    - *Discovery and routing capabilities*
    - *Implement a full stack*
    - *Higher consumption*

# ZigBee: Network Layer (2)

## Device roles (2)

- ZigBee Network Coordinator (ZC)
  - *Sets up a network (selecting the channel and PAN ID).*
  - *Transmits network beacons.*
  - *Manages network nodes.*
  - *Distributes addresses, allowing routers and end devices to join the network. Assists in routing data.*
  - *Buffers data packets for sleeping end device children.*
  - *Cannot sleep and must be powered on at all times.*

# ZigBee: Network Layer (3)

## Functions

- Functions of the NWK layer (implemented in the different devices of the network)
  - Starting/creating a network
  - Joining and leaving a network
  - Assign addresses to devices
  - Synchronization within a network through tracking beacons or by polling.
  - Security
  - Routing
    - *Logical address (16 bits long)*
    - *Multi-hop network*

# ZigBee: Application Layer (1)

## Application Support Sublayer (APS)

- Services provided by the application support sublayer:
  - Generation of the application level PDU (APDU) by adding the appropriate protocol overhead.
  - Binding: Devices can be bound, and then this layer is able to transfer a message from one to other
  - Group address filtering: The ability to filter group-addressed messages based on endpoint group membership.
  - Reliable transport: Increases the reliability of transactions above that available from the NWK .
  - Duplicate rejection: Messages offered for transmission will not be received more than once.
  - Fragmentation: Enables segmentation and reassembly of messages longer than the payload of a single NWK layer frame.
  - AIB (APS Information Base) management: Manages the information about bounded devices.
  - Security: The ability to set up authentic relationships with other devices through the use of secure keys.
  - Group management: The ability to declare a single address shared by multiple devices, to add devices to the group, and to remove devices from the group

# ZigBee: Profiles (2)

- Profiles are the key element for the communication between ZigBee devices
- Define the offered services and describes a common language for exchanging data: type of messages, available commands and their responses, etc.
  - *Allow communication between separated devices to build a distributed application*
  - *Guarantee device interoperability across different manufactures*
- Mostly all ZigBee operation must be defined under a profile
  - *For example, tasks for joining the network or discovering devices and services are supported by ZigBee Device Objects (ZDO)*

# Wide area communications

Cellular technologies that provide Wide Area connectivity for IoT



## Section III

Cellular connectivity

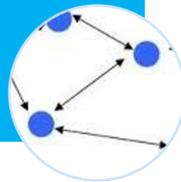
---

# Long Range IoT

## Alternatives

- ZigBee (802.15.4)
- Bluetooth LE
- WiFi HaLow (802.11ah)
- Z-Wave, ANT+
- ISA 100, WirelessHart

Multihop in short range technologies



- Traditional: GSM, GPRS, 3G, LTE, ...
- LTE-MTC (LTE-M), LTE-eMTC
- NB-CIoT, NB-LTE
- GSM Evolution

Cellular Networks (traditional or M2M)



- Sigfox
- LoRa, LoRaWAN
- NWAve (Weighless-N)
- Platanus (Weighless-P)
- Weighless-W
- Ingenu (OnRamp)

Low Power Wide Area Networks (LPWAN)



# Introducing LPWAN (Low Power WAN)

- LPWAN describes a category of wireless communications technologies designed to support IoT deployments
  - *Strong coverage over large areas*
    - Even when devices are underground or deep within buildings
  - *Great power efficiency*
    - Devices can run on batteries for 10 years or more
  - *Massive scale*
    - Connecting potentially millions of devices at once in a single deployment
    - Each cell or base station should be able to support more than 10k devices
  - *Low cost communications hardware (< 10 \$)*
  - *Low bandwidth*
    - Most of the use cases requires a few bytes of data to be transmitted per device per day

# Data Rate & Range

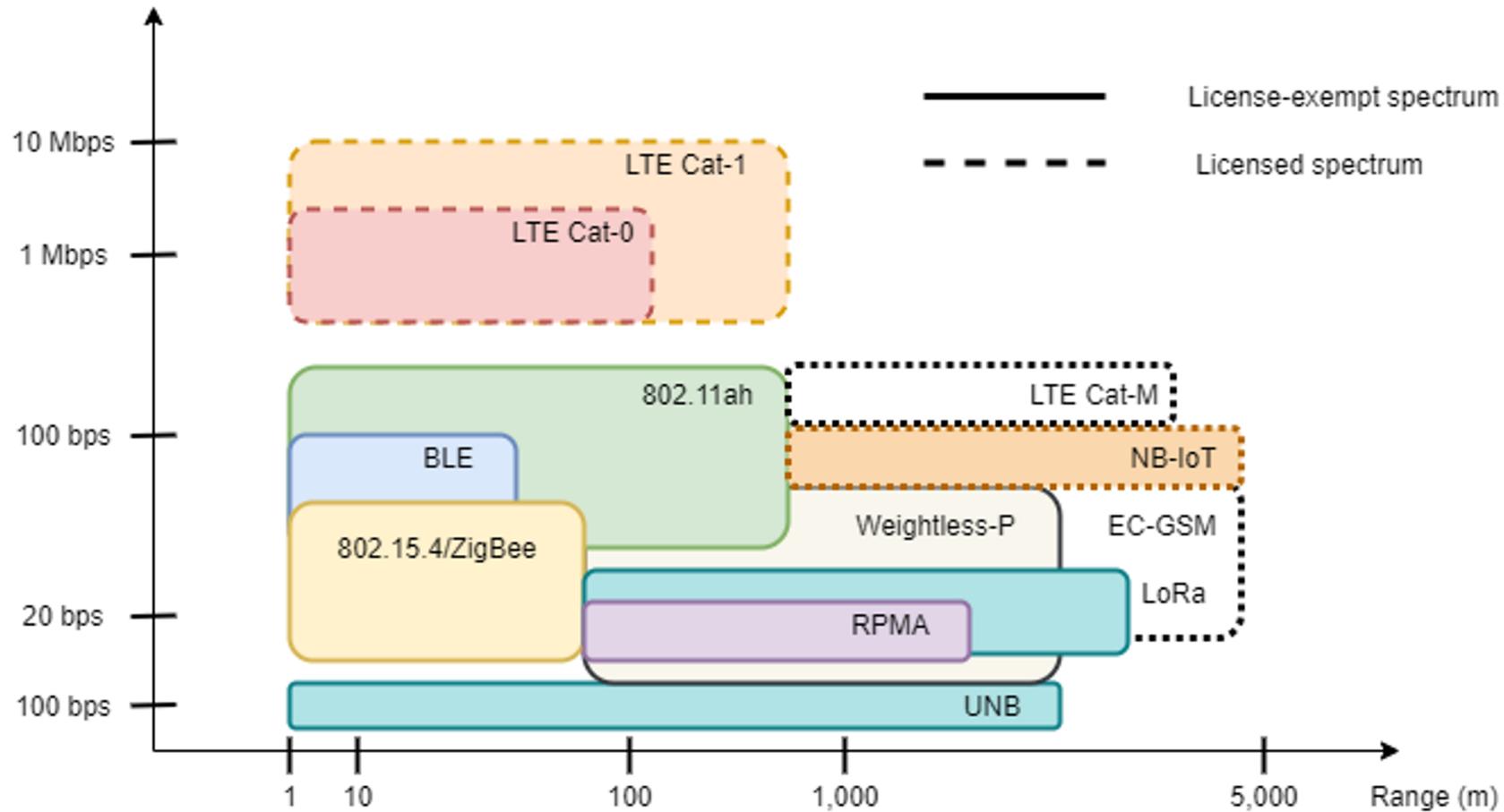


Figure 16: IoT technologies. Range vs data rates

# Comparison

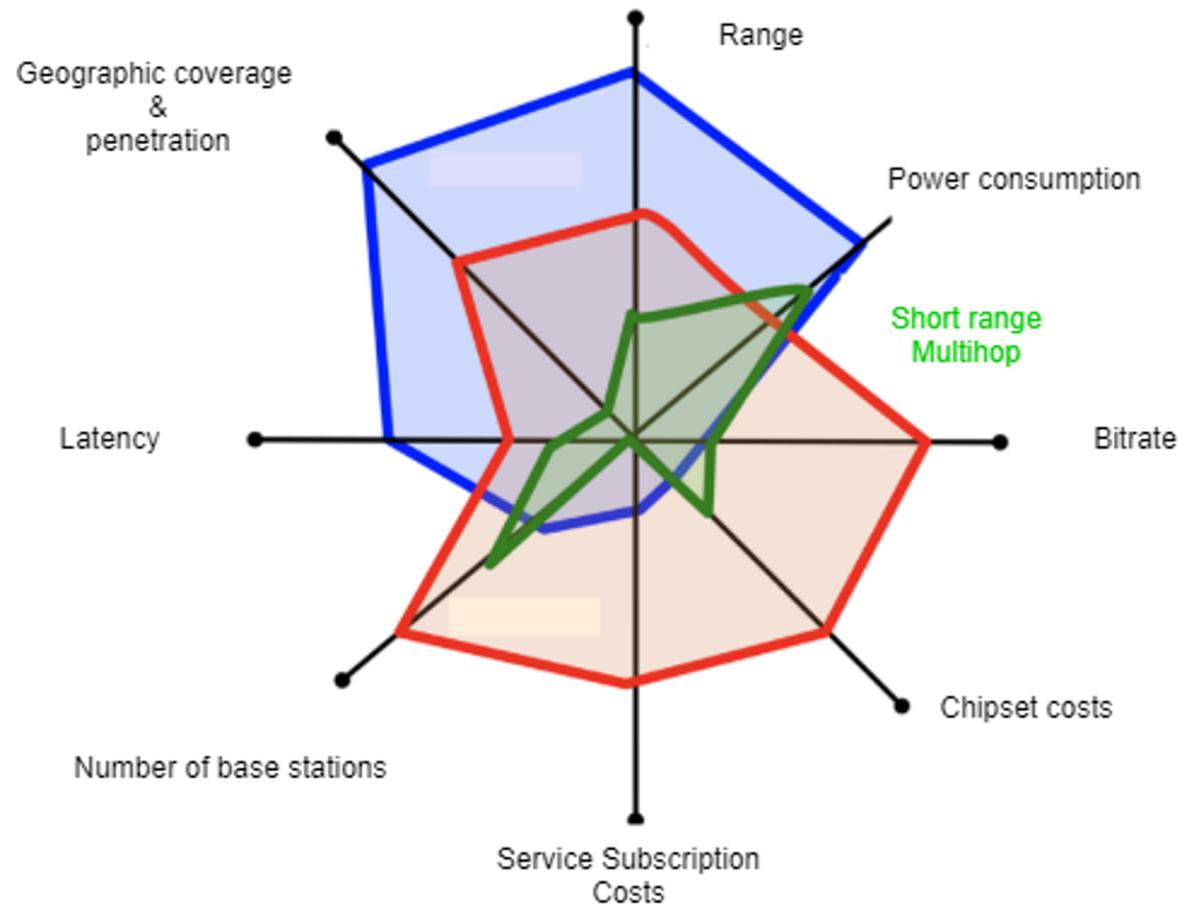


Figure 17: Comparison between IoT technologies for wide area coverage



## Cellular connectivity

- Sigfox

# Sigfox

- Sigfox (enterprise) is the owner of the whole technology (end-to-end)
  - *Licences the radio technology (with specific agreements)*
    - Devices from STMicroelectronics, Atmel, Texas instruments ...
    - They want that the production cost was as low as possible to have a bigger market
  - *Unidirectional communication*
    - To provide bidirectional communication requires higher network density
- **Business model**
  - *Pay per use*
    - Collaboration with operators to deploy their network
      - One operator per area
    - Offers cheaper hardware

# Sigfox

## Physical Layer (I)

- BPSK modulation with Ultra Narrow-Band (UNB) technology:
  - *Improves scalability, reduces interference, less power, ...*
- ISM bands: 868 & 900 MHz
- Topology: unidirectional link between devices and server
  - *Sigfox do not allow nodes to communicate to each other*
  - *Some fabricants implement the bidirectional communication (Telit, Star Network)*
  - *It is possible to configure ACK messages at the server side*
- Range
  - *LOS (Line-of-Sight): 1000 km, according to Sigfox*
  - *Rural areas: from 30 to 50 km*
  - *Urban areas: from 3 to 10 km*

# Sigfox

## Physical Layer (II)

- Data Rate
  - *100 bps*
  - *Cost: ~16€/year*
    - Uplink: 140 messages/day, 12 bytes payload
    - Downlink: 4 ACK messages/day, 8 bytes payload
    - Overpassing the limits will have a penalty
- Other:
  - *Frequency Hopping*
  - *Encryption is not used by default, but supports AES*
  - *Plug & Play devices, easy deployment*
  - *Frame format very simple*
  - *Low power*

# Sigfox

## Network & Service

- Protocols/Architecture:
  - *Proprietary (none or few public information)*
- Use interfaces:
  - *Web interface <http://backend.sigfox.com>*
  - *API REST*
    - “Pull” mode (queries) to get programmatically the same information as through the web interface
  - *Callbacks*
    - Subscription to an HTTP callback for each message received by a node

# Sigfox

## Use cases

- When is Sigfox recommended? (Source: Libelium)
  - *Sigfox is NOT advised for projects with a regular duty-cycle, which require sending one frame every few minutes*
    - Exceeding the 140 packets per day limit may lead to extra fee charges or to Sigfox license cancellation.
  - *Sigfox is recommended for long-range device communications in cities where their base stations are deploy.*
  - *Sigfox is NOT recommended for bi-directional communications.*
    - There is no a real data downlink. Although, packet acknowledges and callbacks are provided.
  - *Sigfox is NOT recommended for real-time streaming.*
    - Transmissions are not performed in real time as there is a minimum delay for packet arrival
  - *Sigfox is NOT recommended for large amount of data transmissions*
    - Maximum payload is 12 bytes.



## Cellular connectivity

- LoRa

# LoRa

- Follows the opposite strategy: only one enterprise builds the transceivers (Semtech)
  - *LoRa only defines the physical layer (LoRaWAN defines the network)*
  - *Possible licences for third parties*
- The higher levels of the protocol stack are managed by the LoRa Alliance
  - *Alliance participated by Alianza (\$50K), contributors (\$20K) and adopters (\$3K – sellers of certificated products).*
  - *An adopter can build certificated modules or gateways, and use the alliance logo.*
- LoRaAlliance also specifies LoRaWAN, a network architecture similar to Sigfox, indicating how to build the network of an operator
- Star topology
  - *The gateways send the information gathered by the devices to a central infrastructure (using the Internet)*

# LoRa

## Physical Layer

- Uses its own spread spectrum modulation scheme (LoRa Spread Spectrum), variant of Chirp Spread Spectrum.
  - *Robustness, low cost, low power*
- ISM bands: 868 (Europe), 915 (USA) and 433 MHz (Asia).
- Bidirectional communication:
  - *Uplink: 0.3 to 50 kbps (adaptive) C*
  - *Downlink: replies to previous transmission, the sending node remains listening for a while before going to sleep*
- Range:
  - *Rural areas: up to 22 Km*
  - *Urban areas: from 2 to 5 km*
- Others:
  - *Scalable bandwidth*
  - *Low power*
  - *Robust against fading, multipath, Doppler effect*

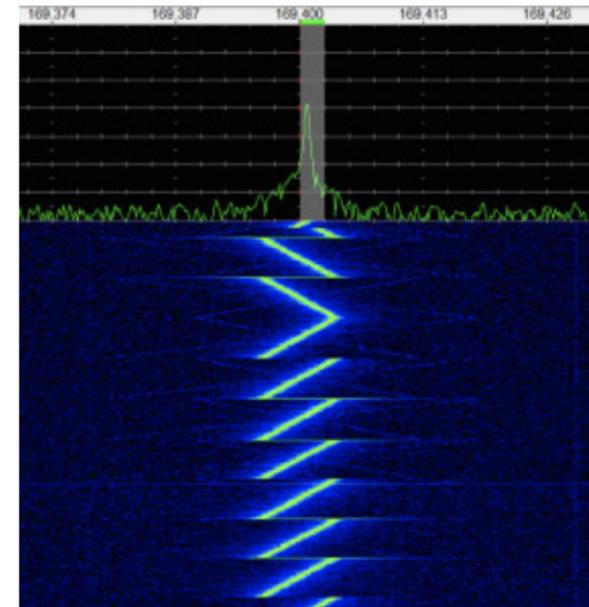


Figure 18: Chirp modulation

# LoRa

## Network & Service (I)

- LoRaMAC

- *Open source MAC protocol (IBM, Semtech, Actility, Microchip).*
- *Different node classes*
- *Unique identifier per node (allows to use IPv6 or 6LowPAN)*
- *Security: AES encryption to the end application*

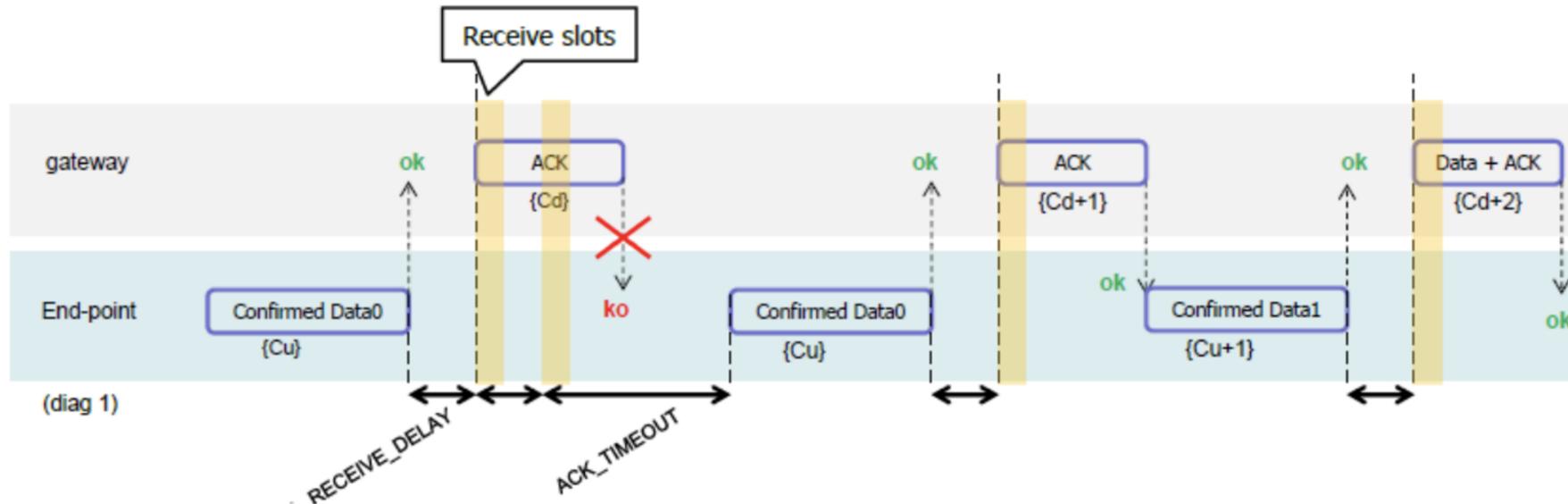


Figure 19: Lora MAC uplink timing diagram for confirmed data message

# LoRa

## Network & Service (II)

- Three device classes:
  - *Class A*
    - Must be support by all devices
    - Send packets to the network, and opens to reception windows
      - Downlink available only after sensor TX
    - Adaptive modulation
    - Battery powered sensors
  - *Class B*
    - Slotted communication synchronized with a beacon
      - Receive window opens in planned periodic intervals
    - Battery Powered actuators
      - Latency in the order of seconds
  - *Class C*
    - Devices which can afford to listen continuously (except when transmitting)
      - Medium/high energy consumption at end device
    - No latency for downlink communication
      - Suitable for actuators with strict time constrains
    - Main power actuators (higher power consumption)
      - Automotion, control

# LoRa

## Network & Service (III)

- LoRaWAN: the LoRa Alliance defines how to build a public or private network

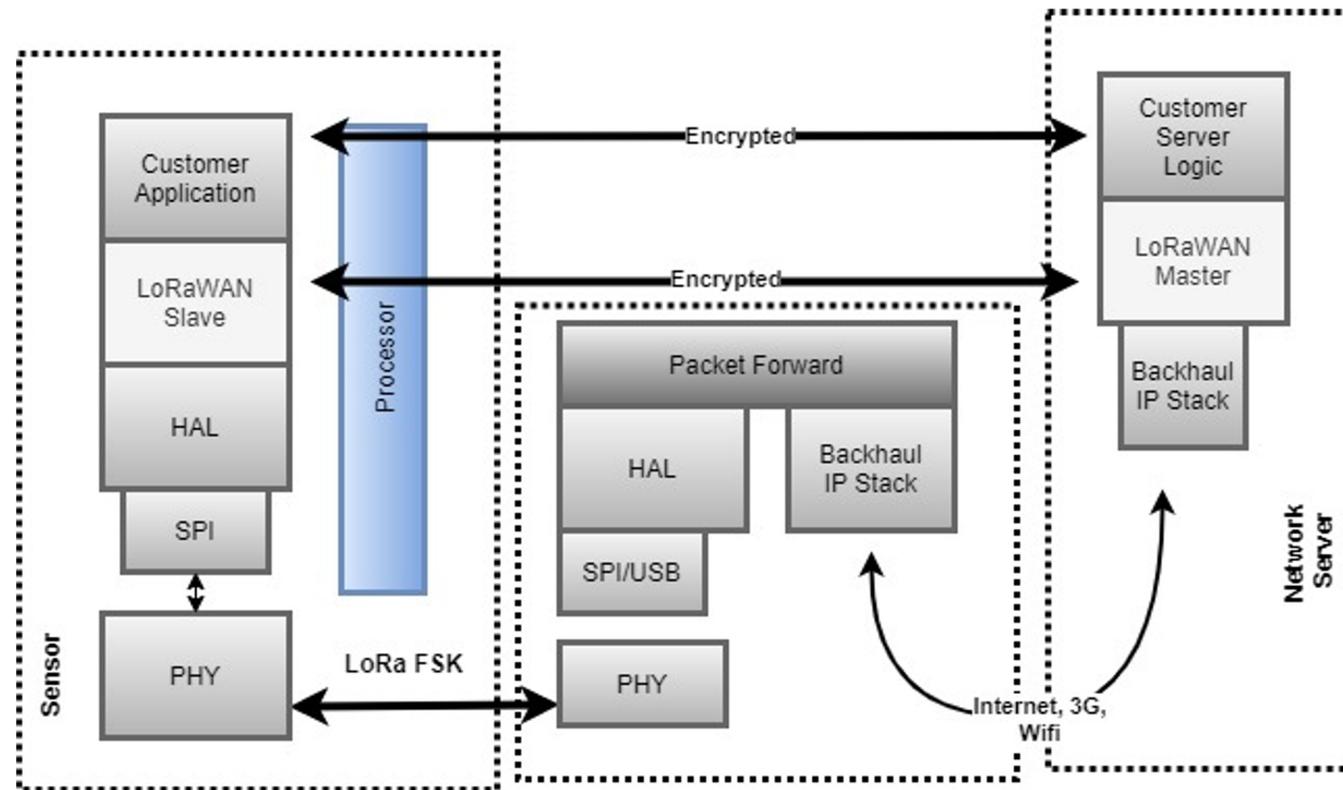


Figure 20: LoRa WAN architecture and communication stack (source: <https://lora-alliance.org>)

# LoRa

## Network & Service (IV)

- Architecture
  - *Star of stars topology*
    - End nodes send information to gateways
      - The gateways of the same network are synchronized
      - The gateways must listen all frequencies
      - The frequencies and the data rates depend on the distance and duration of the transmissions
    - The gateways resend the information to a central server (using IP)
  - *User applications are running in the cloud*
  - *Includes mechanisms for, but do not define:*
    - Roaming
    - OTA (over the air) update
    - QoS

# LoRa

## Private Networks (I)

- LoRa can be used as the physical layer, and implement any protocol over it:
  - *Custom protocols*
  - *Implementations of existing protocols: 802.15.4 MAC, 6LoWPAN*
  - *Using previous implemented solutions*
- **Simphony-Link**
  - *LoRa Physical Link*
  - *IEEE 802.15.4 MAC*
  - *Bidirectional communications*
  - *Low latency*
- **Virtual Extension VEMesh**
  - *Mesh topology*
  - *Multihop*
  - *Bidirectional communication*
  - *TDMA*



## Cellular connectivity

- NB-IoT (4G, 5G)

# Narrow Band - IoT

- NB-IoT is an open 3GPP standard based on LTE
  - *Release 13, June 2016*
- NB-IoT is a cellular technology (licensed spectrum)
  - *Compatible with existing cellular network infrastructure,*
  - *With the same level of security as LTE*
    - Mutually authentication of network and devices
    - Traffic between the device and the core network encrypted

# NB-IoT deployment scenarios

- Three possible types of deployment
  - *Standalone*
    - Re-farm GSM spectrum or use totally new spectrum
  - *Guard band of LTE spectrum*
  - *In band*
    - 1RB\* (180kHz) of LTE system
      - RB = Resource Block

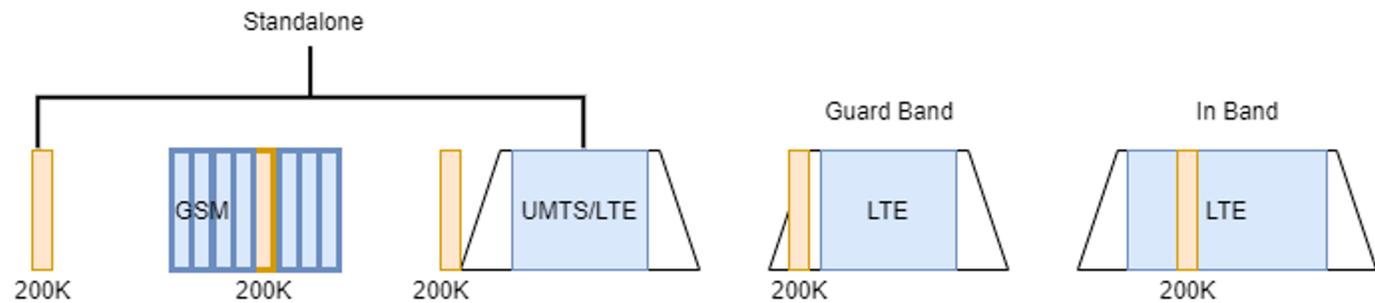


Figure 21: NB-IoT deployment alternatives

# NB-IoT

## Technical features

- NB-IoT = cellular communication + narrowband communication
  - *Narrowband: feature of GSM*
  - *OFDMA/SC-FDMA: feature of LTE*
- Multiple access/Modulation scheme
  - *Uplink: SC-FDMA (subcarrier spacing: 3.75kHz and 15kHz)*
    - Single tone transmission: Pi/2-BPSK and Pi/4-QPSK
    - Multi tone transmission: QPSK
  - *Downlink: OFDMA (subcarrier spacing: 15kHz) and QPSK*
- Duplex mode
  - *Half duplex FDD*
- UE maximum output power
  - *23dBm*
  - *20dBm*

# Networks of IoT devices

IoT devices can create networks to satisfy use cases where an area has to be covered with devices



## Section IV

Wireless Sensor Networks

---



## Wireless Sensor Networks

- Introduction: multi-hop networks

# Wireless Sensor Networks

- “A Wireless Sensor Network (WSN) is a self-configuring network of small sensor nodes communicating among them using radio signals, and deployed in quantity to sense the physical world” (H. Karl and A. Willig).
- Used when:
  - There is no infrastructure available (e.g. disaster zones)
  - Building an infrastructure is too expensive
  - There is no time to build an infrastructure (e.g. military operations)

# Roles of participants in WSN

- Sources of data: Measure data, report them “somewhere”
  - Typically equipped with different kinds of actual sensors
- Sinks of data: Interested in receiving data from WSN
  - May be part of the WSN or external entity, smartphone, gateway, ...
- Actuators: Control some device based on data, usually also a sink

# Wireless Sensor Networks

## Use cases (1)

- Disaster recovery
  - Deploy sensors in areas that have to be observed (e.g. wildfires)
  - Each node measures temperature/fire and transmit that information
- Biodiversity mapping
  - Use sensor nodes to observe wildlife
- Intelligent buildings (or bridges)
  - Reduce energy wastage by proper humidity, ventilation, air conditioning (HVAC) control
  - Needs measurements about room occupancy, temperature, air flow, ...
  - Monitor mechanical stress after earthquakes

# Wireless Sensor Networks

## Use cases (2)

- Facility management
  - Intrusion detection into facilities
  - Sensing dangerous products
- Machine surveillance and preventive maintenance
  - Embed sensing/control functions in places hard to connect with a wire
- Precision agriculture
  - Analyze the soil and apply fertilizer/pesticides/irrigation only where needed
- Medicine and health care
  - Post-operative or intensive care
  - Long-term surveillance of chronically ill patients or the elderly

# Wireless Sensor Networks

## Use cases (3)

### ○ Logistics

- Track goods with an IoT device
- Know the state of the good (temperature, impacts, etc.)

### ○ Telematics

- Detect traffic conditions (number of vehicles, speed, etc.)
- Intelligent roadside
- Cars as the sensor nodes

# Challenges for ad hoc networks

- Distributed protocols/algorithms are “harder” than centralized protocols.
- Limitations:
  - There is no central entity to organize/orchestrate. Participants must organize themselves
    - *No centralized MAC mechanism*
    - *No centralized routing*
  - Limited range of wireless communication
    - *Need for multi-hop communications*
  - Mobility of participants
    - *Mobile ad-hoc networks*
  - Battery-operated entities
    - *Protocols must be efficient*

# Mobile ad hoc Networks (MANET)

- Mobility of devices
  - There is no infrastructure that continues providing coverage when nodes move
- Mobile ad hoc Networks (MANET)
  - Mobility changes routes
  - Routing protocols (usually) have to be energetically efficient
    - *It is necessary to take into account the energy in the battery for routing mechanisms*

# Requirements for WSNs

## Requirements (1)

- Data-centric networks
  - It is not (really) relevant which node provides the information
- Quality of service
  - Traditional QoS metrics do not apply
  - But the right answers at the right time must be obtained
- Fault tolerance
  - Be robust against node failure (running out of energy, physical destruction, ...)
- Lifetime
  - The network should fulfill its task as long as possible
  - Lifetime of individual nodes relatively unimportant

# Requirements for WSNs

## Requirements (2)

- Scalability
  - Support large number of nodes
- Wide range of densities
  - Vast or small number of nodes per unit area
- Programmability
  - Re-programming of nodes in the field might be necessary, improve flexibility
- Maintainability
  - WSN has to adapt to changes, self-monitoring, adapt operation
  - Incorporate possible additional resources, e.g., newly deployed nodes



# Wireless Sensor Networks

- 6LowPAN

# 6LoWPAN

## Introduction

- WSN (Wireless Sensor Networks) require standard protocols to interoperate
- The IETF has different working groups (WGs) developing standards to be used by WSN, being the most known 6LoWPAN
  - **6LoWPAN:**
    - *IPv6 over Low-power Wireless Personal Area Networks*
    - *Standards for IPv6 communication over the IEEE 802.15.4*
    - *Adaptation layer between IPv6 and the low power and lossy wireless communications medium.*
    - *No IPv4 support is available.*

# LoWPANs

## Overview

- Low-power and lossy networks (LLNs):
  - Networks made of highly constrained nodes
    - *limited CPU, memory, power*
  - Interconnected by a variety of "lossy" links (low power radio links).
  - Characteristics
    - *Low speed, low performance, low cost, unstable connectivity.*
- LoWPAN: particular instance of an LLN
  - Formed by devices complying with the IEEE 802.15.4 standard.

# Overview of LoWPANs

## Characteristics of LoWPANs

- Small packet size: Maximum of 81 octets for data packets.
- IEEE 802.15.4 defines several addressing modes:
  - IEEE 64-bit extended addresses
  - 16-bit addresses unique within the PAN.
- Low bandwidth:
  - 250 kbps, 40 kbps, and 20 kbps.
- Topologies include star and mesh
- Large number of devices:
  - Location of the devices is typically not predefined.
  - They may move to new locations.
- Unreliable devices:
  - uncertain radio connectivity, battery drain, device lockups, physical tampering, etc.
- Sleeping mode

# Overview of LoWPANs

## Use of IP on LoWPANs

- IP networking provide the following benefits to LoWPANs:
  - Use of a pervasive technology (IP).
    - *Easier development of applications (well-known)*
    - *Interconnectable without protocol translation*
    - *Large amount of addresses available*
- There are also issues that make it difficult to use IP:
  - IP packets have to compressed to fit IEEE 802.15.1 frames
  - It is necessary to implement specific configuration and management protocols adapted to LoWPANs
    - *Service Discovery*
    - *Security*

# 6LoWPAN

## Protocol stack

- Lower layer
  - Physical and link layers: IEEE 802.15.4
  - Provide connectivity to devices
- 6LoWPAN
  - Deployed over the lower layer
  - Is an adaptation layer that solves issues between the lower layer and IPv6
- IPv6

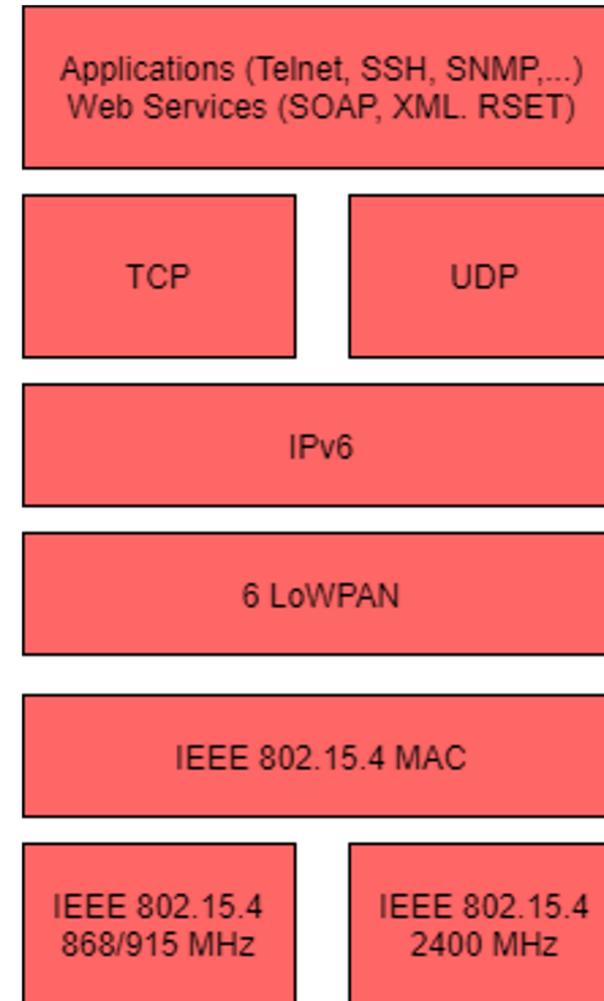


Figure 22: 6LoWPAN protocol stack (Source: IoT in 5 days)

# 6LoWPAN

## Main goals of 6LoWPAN

- Fragmentation and Reassembly layer
- Header Compression (otherwise IPv6 headers would use almost all the IEEE 802.15.4 frame payload)
- Address Autoconfiguration
- Mesh Routing Protocol
- Assign IPv6 frames in the correct IEEE 802.15.4 frame type
- Use link layer ACKs to help upper layers
- Suppress source or destination address (included in the IEEE 802.15.4 frame)
- Allow including source or destination PAN ID fields
- Support 64-bit extended addresses and 16-bit short addresses
- Multicast is not supported natively in IEEE 802.15.4, so they have to be adapted

# 6LoWPAN

## Solutions provided

- Adaptation format specified to carry IPv6 datagrams over constrained links (bandwidth, memory, or energy resources):
  - Mesh Addressing header to support sub-IP forwarding.
  - Fragmentation header to support the IPv6 minimum MTU requirement.
  - Broadcast Header for IPv6 multicast packets.
  - Stateless header compression for IPv6 datagrams to reduce the large IPv6 and UDP headers used as the LoWPAN encapsulation

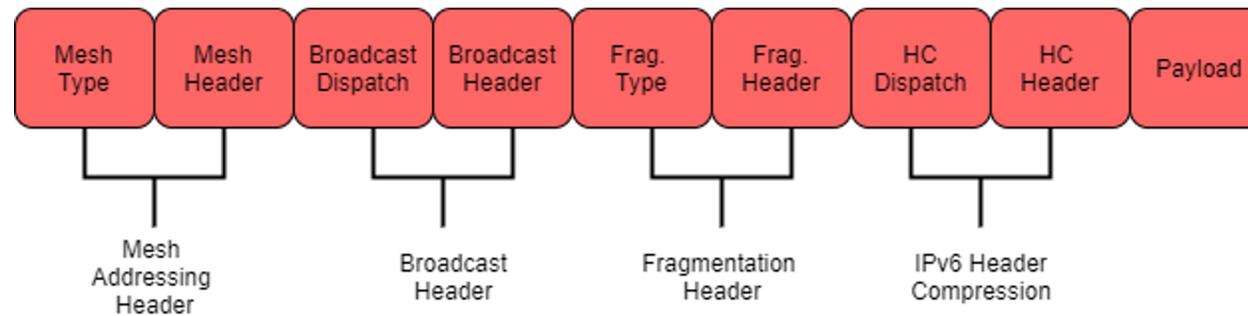


Figure 23: IPv6 frame (Source: IoT in 5 days)



This Photo by Unknown Author is licensed under [CC BY-NC](#)

# Summary

---

# Summary

- There is an overwhelming number of options for IoT connectivity
- They have to satisfy different requirements according to the application
- Different communication possibilities
  - Short range communication technologies (connect IoT nodes to the Internet through a gateway)
    - *Wi-Fi, Bluetooth, Zigbee, etc.*
  - Cellular networks (LPWAN)
    - *Sigfox, LoRa, NB-IoT (4G, 5G)*
  - Wireless Sensor Networks
    - *IoT devices interconnected*



This Photo by Unknown Author is licensed under [CC BY-NC](#)

# References

---

# References

- Schiller, Jochen H. *Mobile communications*. Pearson education, 2003.
- Wi-Fi Specifications
  - <https://www.wi-fi.org/discover-wi-fi/specifications>
  - <https://www.wi-fi.org/discover-wi-fi/white-papers>
- Bluetooth Specifications
  - <https://www.bluetooth.com/specifications>
  - <https://www.bluetooth.com/specifications/mesh-specifications/>
  - <https://www.bluetooth.com/bluetooth-technology/topology-options/le-mesh/mesh-glossary/>
  - <https://www.bluetooth.com/blog/introducing-bluetooth-mesh-networking/>
- Bluetooth Technology developer training videos
  - <https://www.bluetooth.com/develop-with-bluetooth/developer-resources-tools/developer-training-videos>

# References

- Kevin Twonsend, Carles Cufí, Akiba Davidson, Robert Davidson, "Getting Started with Bluetooth Low Energy", O'Reilly, 2015.
- Naresh Gupta, "Inside Bluetooth Low Energy", Artech House, 2013.
- Callaway, Gordav et al., "Home networking with IEEE 802.15.4: a developing standard for low-rate wireless personal area networks, " IEEE Communications Magazine, vol. 40, no. 8, pp. 70-77, ISSN: 0163-6804, August 2002.
- C. E. Perkins, E. M. Belding-Royer, and S. R. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," IETF Internet Draft, draft-ietf-manet-aodv-13.txt, Feb. 17, 2003.
- <http://www.nets.rwth-aachen.de/content/teaching/lectures/sub/mobil/WS07-08/>
- ZigBee Alliance: <http://www.zigbee.org>
- Patrick Kinney, "ZigBee Technology: Wireless Control that Simply Works"
- William C. Craig Zigbee: "Wireless Control That Simply Works"
- Colina, A. L., Vives, A., Bagula, A., Zennaro, M., & Pietrosevoli, E. (2016). Internet of Things (IoT) in 5 Days.

# References

- Lorenzo Vangelista, Andrea Zanella , and Michele Zorzi, “Long-range IoT technologies: the dawn of LoRa”, Proceedings of the 1st EAI International Conference on Future access enablers of ubiquitous and intelligent infrastructures (Fabulous 2015) Sep. 23-25, 2015, Ohrid, Republic of Macedonia.
- Sigfox: <https://www.sigfox.com/es>
- LoRA Alliance: <https://www.lora-alliance.org/>
- Rachel Wang. “Wireless IoT Technologies and Applications -Narrowband IoT Introduction”. May 2016.
- “NB-IOT – Enabling New Business Opportunities”. Huawei Technologies
- Holger Karl and Andreas Willig. Protocols and Architectures for Wireless Sensor Networks. Wiley, 2005
- Antonio Liñán Colina, et al., "Internet of things in 5 days". 2016.
- Sabine Roessel & Stefania Sesia. “Cellular Internet-of-Things –Explained”. Globecom 2016.



**Thank  
You**

---

**PROUDER**  
Introducing Recent Electrical Engineering  
Developments Into undergraduate curriculum



# This week's topics...

- IoT Connectivity Paradigms
- Application Layer Protocols for the IoT
- Integrating IoT with Current Networks
- Test Cases



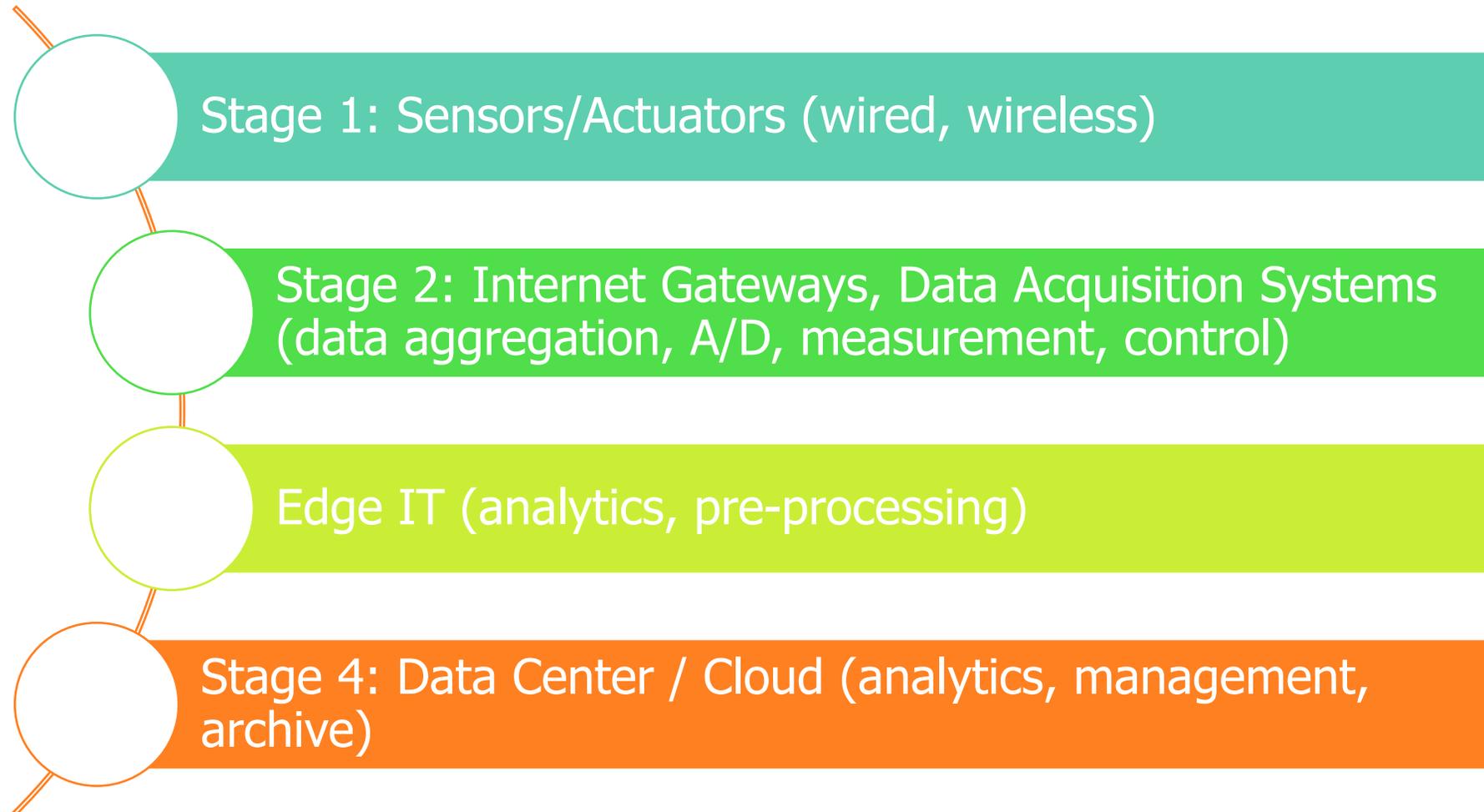
# Internet of Things Connectivity

## A General View

- The Internet of Things is a complex system-of-systems.
- A general view of the IoT architecture is as follows:
  1. Sensors and actuators (providing access to the outer world via sensing and actuating)
  2. Internet gateways and Data Acquisition Systems (processing the enormous amount of information collected on the previous stage and converting)
  3. Edge IT (performing enhanced analytics and data pre-processing)
  4. Data center and cloud (providing in-depth processing and service delivery)

# Internet of Things Connectivity

## A General View

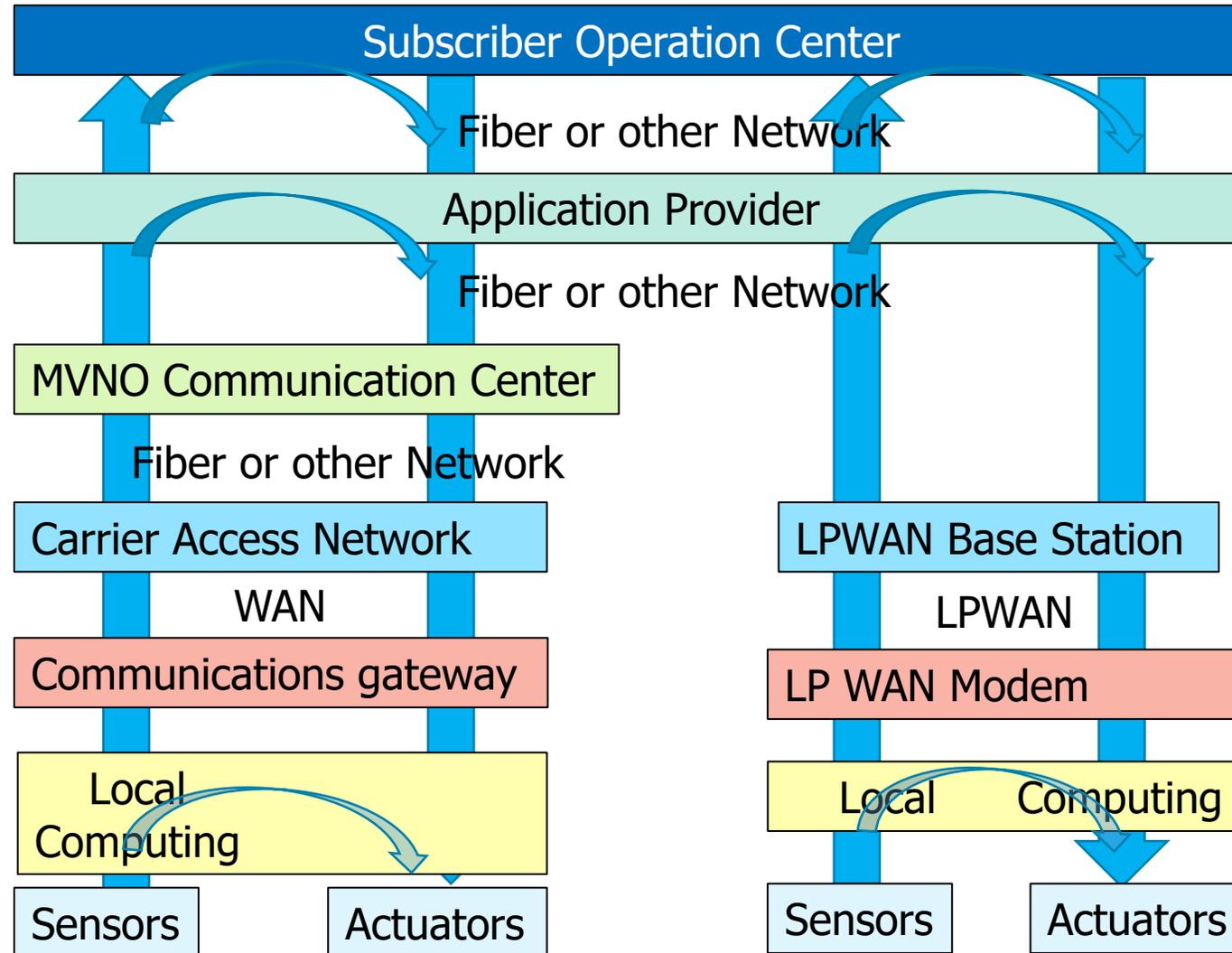


Source: P. Stokes, "4 Stages of IoT architecture explained in simple words," Medium.com

# Internet of Things Connectivity

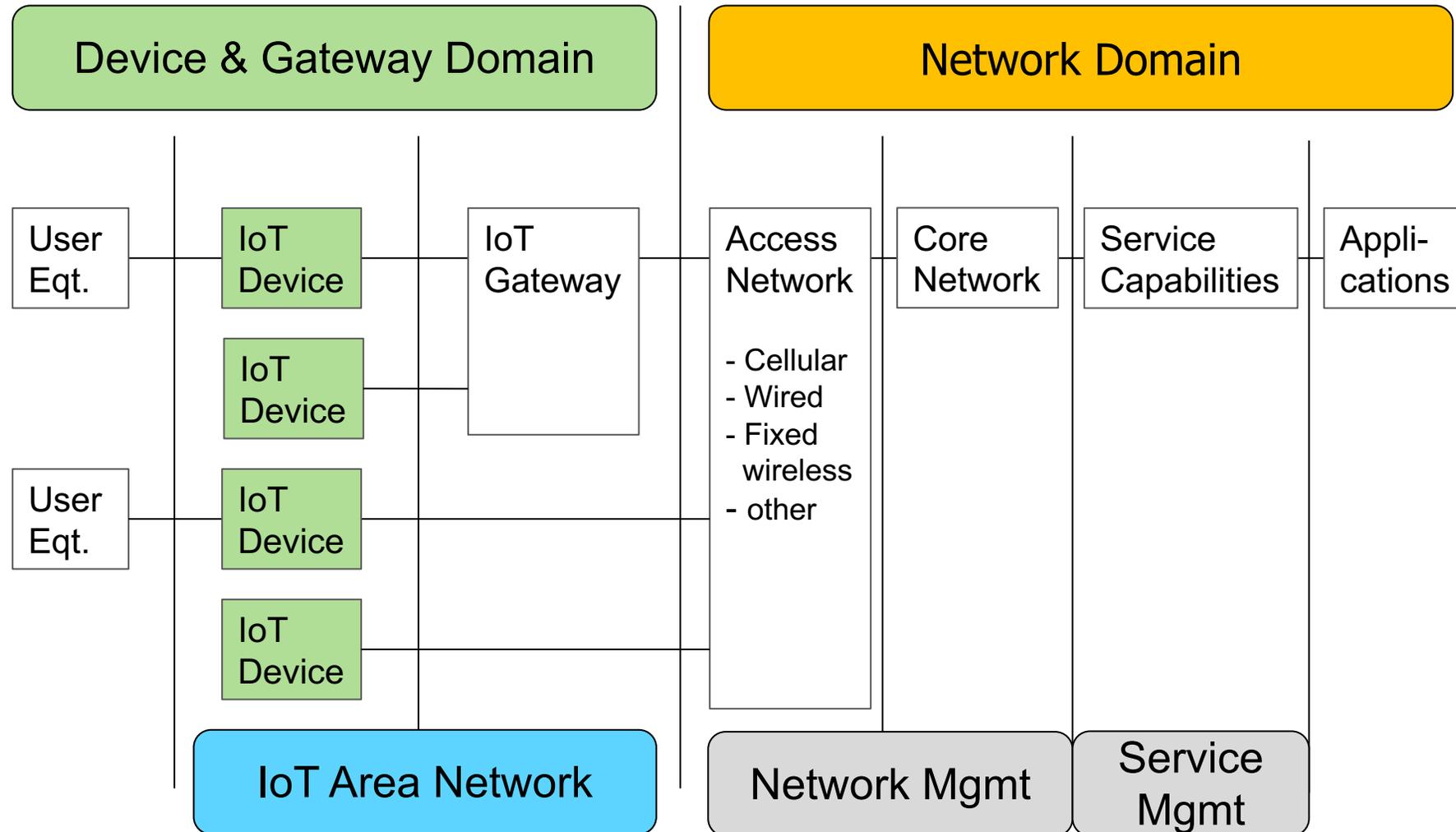
## Example Data Flows

Action may be taken at different levels



# Internet of Things Connectivity

## ETSI Model



# Internet of Things Connectivity

## IETF Model

- The IETF model represents an Internet-centric model, which is based on the 6LoWPAN architecture.
- The 6LoWPAN concept originated from the idea that "the Internet Protocol could and should be applied even to the smallest devices," and that low-power devices with limited processing capabilities should be able to participate in the Internet of Things.
- The base specification developed by the 6LoWPAN IETF group is RFC 4944 (updated by RFC 6282 with header compression, and by RFC 6775 with neighbor discovery optimizations).
- RFC 4944 proposes an IPv6 over Low-Power Wireless Area Networks approach based on direct end-to-end Internet integration.
- IPv6 over Bluetooth Low Energy (BLE) is defined in RFC 7668.

# Internet of Things Connectivity

- IETF Model

## Regular Web

- Usually ample bit rate
- Usually ample power
- Large processing power
- Large data flows
  - Streaming AV & Flash
  - Pretty web pages
- Near continuous connectivity

## IoT

- Bit rate constrained
- Power constrained
- Processor constrained
- Small data flows
  - Sensor data
  - Operational state
- Intermittent connectivity

TLS	=>	DTLS
IPv4 /x	=>	6LowPAN
HTTP	=>	CoAP
OSPF	=>	RPL

# Section Outline

HTTP

MQTT

CoAP

WebSocket

AMQP



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

## Section 2

Application Layer Protocols for the IoT

# Application Layer Protocols for IoT

## Introduction

- REST connectivity over the Internet is used as the communication architecture for the IoT devices.
- Typically, the IoT devices are resource constrained, and they may be subject to data loss or a high memory requirement
- HTTP can be used.
- Alternatively, a few protocols that are effective are MQTT, CoAP, XMPP, WebSocket, and AMQP.

# Application Layer Protocols for IoT

## HTTP

- Hypertext Transfer Protocol represents a possible alternative to support IoT services and maintain compatibility with the World Wide Web.
- HTTP is based on a client-server paradigm, where the client requests data from the server through a TCP connection.
- HTTP messages are text-based
- Both HTTP headers as well as the transported format (typically html text or binary data converted to text format) can be compressed.

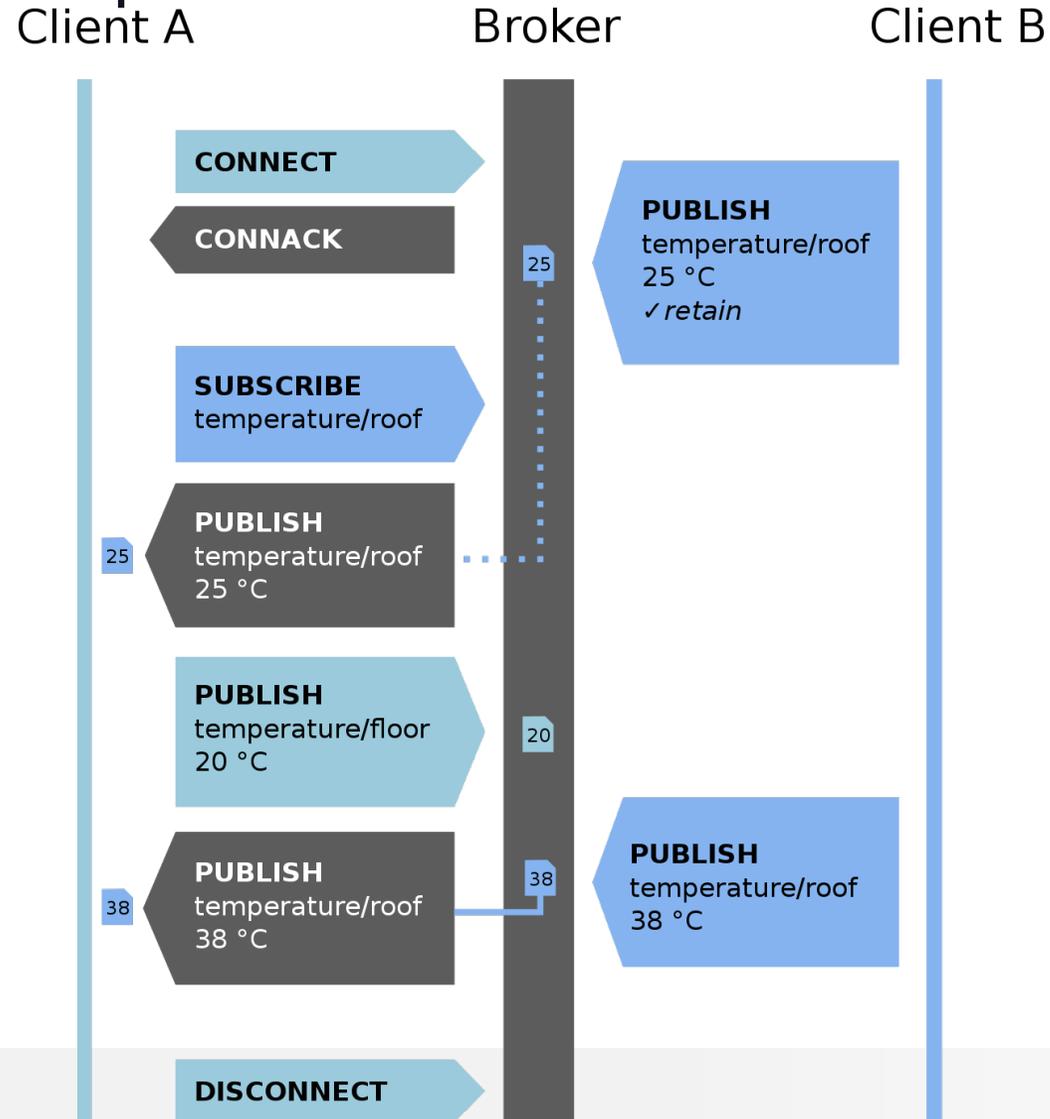
# Application Layer Protocols for IoT

## MQTT

- MQTT (Message Queuing Telemetry Transport) is an open ISO standard (ISO/IEC 20922)
- It offers a lightweight, publish-subscribe network protocol that transports messages between devices
- The protocol usually runs over TCP/IP
- The MQTT protocol defines two types of network entities: a message broker and a number of clients:
  - An MQTT broker is a server that receives all messages from the clients and then routes the messages to the appropriate destination clients.
  - An MQTT client is any device that runs an MQTT library and connects to an MQTT broker over a network

# Application Layer Protocols for IoT

## MQTT Connection Example



# Application Layer Protocols for IoT

## CoAP

- Constrained Application Protocol (CoAP) is a specialized Internet Application Protocol for constrained devices, as defined in RFC 7252.
- It enables those constrained devices called "nodes" to communicate with the wider Internet using similar protocols.
- CoAP is also being used via other mechanisms, such as SMS on mobile communication networks.
- CoAP is a service layer protocol that is intended for use in resource-constrained internet devices, such as wireless sensor network nodes.
- It is designed to easily translate to HTTP for simplified integration with the web, while also meeting specialized requirements such as multicast support, very low overhead, and simplicity.
- CoAP can run on most devices that support UDP.

# Application Layer Protocols for IoT

## WebSocket

- WebSocket is an IETF communications protocol, providing full-duplex communication channels over a single TCP connection (RFC 6455).
- WebSocket is distinct from HTTP. Both protocols are located at layer 7 in the OSI model and depend on TCP at layer 4.
- RFC 6455 states that WebSocket "is designed to work over HTTP ports 443 and 80 as well as to support HTTP proxies and intermediaries," thus making it compatible with the HTTP protocol.
- To achieve compatibility, the WebSocket handshake uses the HTTP Upgrade header to change from the HTTP protocol to the WebSocket protocol.

# Application Layer Protocols for IoT

## AMPQ

- The Advanced Message Queuing Protocol (AMQP) is an open standard application layer protocol for message-oriented middleware.
- The defining features of AMQP are message orientation, queuing, routing (including point-to-point and publish-and-subscribe), reliability and security.
- AMQP is a wire-level protocol. A wire-level protocol is a description of the format of the data that is sent across the network as a stream of bytes.

# Application Layer Protocols for IoT

## Summary

Name	Description	Use Case
<b>MQTT</b>	Simple and lightweight IoT protocol designed for constrained devices and low network bandwidth. See <a href="http://mqtt.org">mqtt.org</a> .	Small medical devices with limited network connectivity, mobile apps in mobile devices, sensors in remote locations that communicate with a gateway.
<b>CoAP</b>	Protocol based on the REST model and is suitable for constrained devices such as a microcontroller or a constrained network because it functions with minimum resources in the device or the network. See <a href="http://coap.technology/">http://coap.technology/</a> .	Smart energy applications and building automation applications.
<b>WebSocket</b>	A full-duplex communication channel over a TCP connection.	Implement WebSocket in runtime environments or libraries that act as servers or clients. You can apply WebSockets in an IoT network where chunks of data are transmitted continuously within multiple devices.
<b>XMPP</b>	Uses the XML text format for communication and runs over TCP. It's not fast and uses polling to check for updates when needed. See <a href="https://xmpp.org">https://xmpp.org</a>	Use XMPP to connect your home thermostat to a web server so that you can access it from your phone. It's used in consumer-oriented IoT applications.
<b>AMQP</b>	The message queue asynchronous protocol is for communication of transactional messages between servers. See <a href="https://www.amqp.org/">https://www.amqp.org/</a>	AMQP is best used in server-based analytical functions. It's effectively used in the banking industry.



# Integrating IoT with current networks

## Introduction

- After reviewing the application-level protocols, it is necessary to discuss how to interconnect the objects that we want to use for building IoT services.
- In this framework, we need to consider existing dominant technologies at the different layers of the protocol stack, from the network down to the physical layer of the TCP/IP protocol stack:
  - The Internet
  - Cellular / WAN Networks
  - Dedicated standards

# Integrating IoT with current networks

## The Internet

- Internet Protocol (IP) is the common layer 3 technology, for interoperability and interconnection of different underlying technologies.
- Two versions of IP are available: the “old” IPv4 and the “new” IPv6
- Local Area Network (and in some cases also Metropolitan Area Network) connectivity is dominated by the Ethernet standard and its extensions.
- Most diffused versions of Ethernet include:
  - Fast Ethernet (100Mbps)
  - Gigabit Ethernet (1Gbps), e.g. the Ethernet variant 1000BASE-T defined by the IEEE 802.3ab standard.

# Integrating IoT with current networks

## Cellular / WAN Networks

- Technologies to be used at this stage can be wired or wireless.
- Wired long range connectivity is provided by optical fibers. Optical fibers permit transmission over longer distances and at higher data transfer rates than electrical cables.
- Wireless WANs are represented by cellular networks. Most diffused cellular networks today belong to the 4G LTE standard, while 5G is being deployed.
- LTE networks are based on a star topology with a centralised EPC (Evolved Packet Core) for handling authentication, signalling and network traffic, with the average base station typically capable of handling signalling for up to 1000-1500 devices (low density in the IoT case).
- 5G is expected to address massive machine-type communications as one of the main deployment scenarios.

# Integrating IoT with current networks

## Dedicated L1/L2 Standards

- Standards that are specific to the Internet of Things domain.
- Typically dedicated to low-power communications and WLAN/WPAN scenarios, and they cover layers 1 (physical layer) and 2 (link layer) of the TCP/IP protocol stack:
  - IEEE 802.15.4
  - MIPI Standards
  - SuperSpeed USB Inter-Chip (SSIC)
  - Mobile PCIe
  - SPI
  - MODBUS
  - OBD

# Integrating IoT with current networks

## IEEE 802.15.4

- IEEE 802.15.4 is a technical standard which defines the operation of low-rate wireless personal area networks (LR-WPANs).
- It specifies the physical layer and media access control for LR-WPANs.
- The basic framework conceives a 10-meter communications range with a transfer rate of 250 kbit/s.
- Lower transfer rates of 20 and 40 kbit/s were initially defined, with the 100 kbit/s rate being added in the current revision.
- Important features include real-time suitability by reservation of Guaranteed Time Slots (GTS), collision avoidance through CSMA/CA and integrated support for secure communications.

# Integrating IoT with current networks

## IEEE 802.15.4

- IEEE 802.15.4 devices include power management functions such as link quality and energy detection.
- The standard operates in pure CSMA/CA or TDMA access modes.
- The TDMA mode of operation is supported via the GTS feature of the standard.
- IEEE 802.15.4-conformant devices may use one of three possible frequency bands for operation (868/915/2450 MHz).

# Integrating IoT with current networks

## MIPI Standards

- MIPI (Mobile Industry Processor Interface) Alliance is a global business alliance that develops technical specifications for the mobile ecosystem.
- M-PHY is a high speed data communications physical layer protocol standard developed by the MIPI Alliance, PHY Working group, and targeted at the needs of mobile multimedia devices.
- M-PHY supports a scalable variety of signaling speeds, ranging from 10 kbit/s to over 11.6 Gbit/s per lane:
  - two different major signaling/speed modes, a simple low-speed (using PWM) mode and high speed (using 8b10b).
- Communications goes on in bursts.

# Integrating IoT with current networks

## MIPI Standards

- UniPro (or Unified Protocol) is a high-speed interface technology for interconnecting integrated circuits in mobile and mobile-influenced electronics.
- The UniPro technology aims to provide:
  - high-speed data communication (gigabits/second)
  - low-power operation (low swing signaling, standby modes)
  - low pin count (serial signaling, multiplexing)
  - small silicon area (small packet sizes)
  - data reliability (differential signaling, error recovery)
  - robustness (proven networking concepts, including congestion management).

# Integrating IoT with current networks

## MIPI Standards – UniPro Protocol Stack

Layer #	Layer name	Functionality	Data unit name
LA	Application	Payload and transaction semantics	Message
DME	Layer 4	Transport	Ports, multiplexing, <a href="#">flow control</a>
	Layer 3	Network	Addressing, routing
	Layer 2	Data link	Single-hop reliability and priority-based arbitration
	Layer 1.5	PHY adapter	Physical layer abstraction and multi-lane support
Layer 1	Physical layer (PHY)	Signaling, clocking, line encoding, power modes	PHY symbol

# Integrating IoT with current networks

## MIPI Standards

- The UniPro specification itself covers Layers 1.5, 2, 3, 4 and the DME (Device Management Entity).
- The Application Layer (LA) is out of scope because different uses of UniPro will require different LA protocols.
- The Physical Layer (L1) is covered in separate MIPI specifications in order to allow the PHY to be reused by other (less generic) protocols if needed.

# Integrating IoT with current networks

## MIPI Standards

- System Power Management Interface (SPMI) is a high-speed, low-latency, bi-directional, two-wire serial bus suitable for real-time control of voltage and frequency scaled multi-core application processors and its power management of auxiliary components.
- SPMI obsoletes a number of legacy, custom point-to-point interfaces and provides a low pin count, high-speed control bus for up to 4 master and 16 slave devices

# Integrating IoT with current networks

## SuperSpeed USB Inter-Chip (SSIC)

- InterChip USB is an addendum to the USB Implementer Forum's USB 2.0 specification. The USB 3.0 successor of HSIC is called SuperSpeed Inter-Chip (SSIC).
- IC-USB is intended as a low-power variant of the standard physical USB interface, intended for direct chip-to-chip communications.
- The IC-USB bus's maximum length of 10 cm results in a lower inductance and capacitance and therefore allows lower power requirements.
- IC-USB is being used primarily in embedded systems and standards.
- One of the most relevant areas of application is in mobile phones, where, for instance, ETSI (in specification TS 102 600) has standardized on IC-USB as the official high-speed interface for connections between the phone's main chipset and the SIM card or UICC card.
- USB 2.0 High-Speed Inter-Chip (HSIC) is a chip-to-chip variant of USB 2.0 that eliminates the conventional analog transceivers found in normal USB.

# Integrating IoT with current networks

## Mobile PCIe

- PCI Express (Peripheral Component Interconnect Express), officially abbreviated as PCIe or PCI-e, is a high-speed serial computer expansion bus standard, designed to replace the older PCI, PCI-X and AGP bus standards.
- It is the common motherboard interface for personal computers' graphics cards, hard drives, SSDs, Wi-Fi and Ethernet hardware connections.
- Mobile PCIe specification allows PCI Express architecture to operate over the MIPI Alliance's M-PHY physical layer technology.

# Integrating IoT with current networks

## SPI

- Serial Peripheral Interface (SPI) is a synchronous serial communication interface specification used for short-distance communication, primarily in embedded systems.
- Typical applications include Secure Digital cards and liquid crystal displays.
- SPI devices communicate in full duplex mode using a master-slave architecture with a single master. The master device originates the frame for reading and writing.
- Multiple slave-devices are supported through selection with individual slave select (SS) lines.

# Integrating IoT with current networks

## MODBUS

- Modbus has become a de facto standard communication protocol and is now a commonly available means of connecting industrial electronic devices.
- Modbus is popular in industrial environments because it is openly published and royalty-free. Modbus uses the RS485 or Ethernet as its wiring type.
- Modbus supports communication to and from multiple devices connected to the same cable or Ethernet network. For example, a device that measures temperature and a different device to measure humidity, both of which communicate the measurements to a computer.
- Modbus is often used to connect a plant/system supervisory computer with a remote terminal unit (RTU) in Supervisory Control and Data Acquisition (SCADA) systems in the electric power industry.

# Integrating IoT with current networks

## OBD

- On-board diagnostics (OBD) is an automotive term referring to a vehicle's self-diagnostic and reporting capability.
- OBD systems give the vehicle owner or repair technician access to the status of the various vehicle sub-systems.
- Modern OBD implementations use a standardized digital communications port to provide real-time data in addition to a standardized series of diagnostic trouble codes, or DTCs, which allow a person to rapidly identify and remedy malfunctions within the vehicle.

# Section Outline

Test cases of IoT connectivity



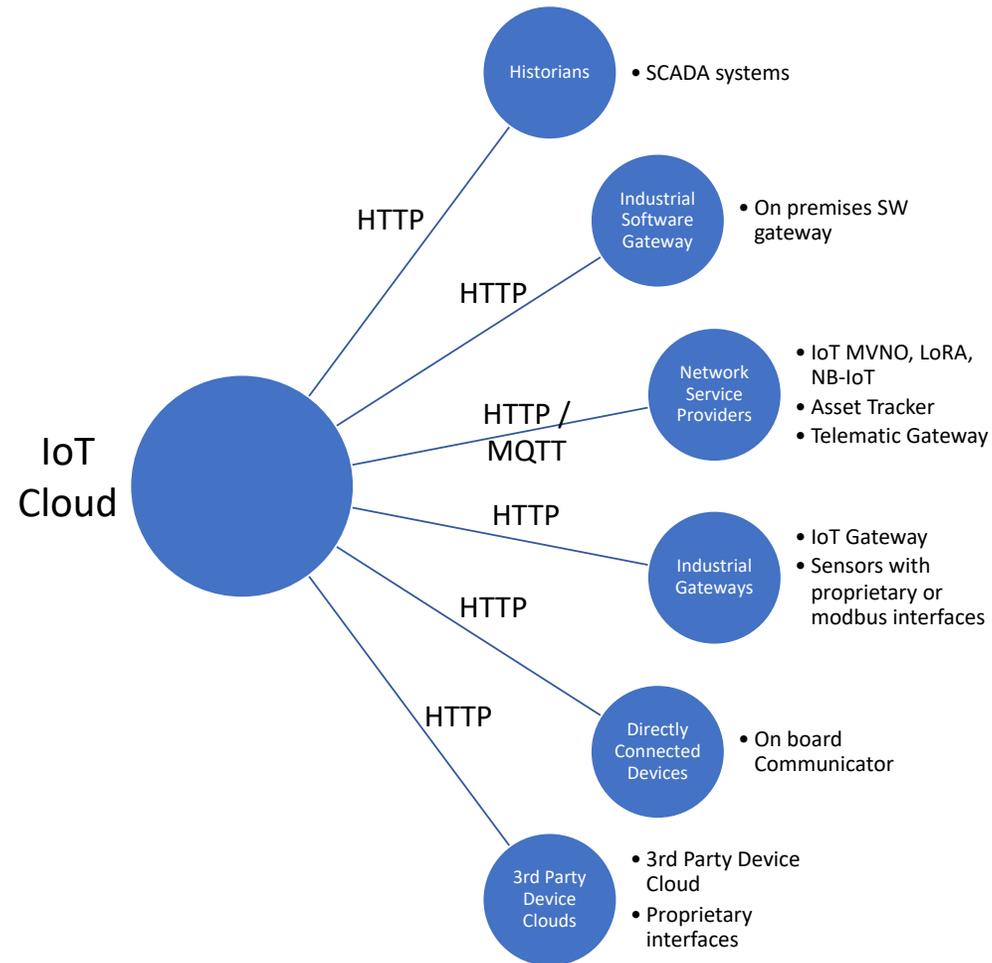
[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

## Section 4

Test Cases

# Test Cases in IoT Connectivity

## Example from Oracle Internet of Things Cloud Service



# Test Cases in IoT Connectivity

## Example from Oracle Internet of Things Cloud Service

Scenario	Option to Connect the Devices	Example
<b>Devices that support proprietary protocols.</b>	Sensors and machines connect indirectly to Internet of Things Cloud Service through a standardized industrial gateway using proprietary protocols.	A device uses the MODBUS protocol to connect with an industrial IoT gateway which, in turn, connects to Internet of Things Cloud Service using the HTTP protocol
<b>Machine with a powerful in-built sensor that supports HTTP protocol.</b>	Directly connect with Oracle Internet of Things Cloud Service using HTTP.	A device such as an on-board communicator directly exchanges messages with the Internet of Things Cloud Service by using the HTTP protocol.
<b>Devices that are already connected to an existing third-party cloud service using proprietary protocols.</b>	Third-party clouds can connect with the Internet of Things Cloud Service using HTTP. This results in the devices getting indirectly connected to Internet of Things Cloud Service.	A device such as an OBD II data logger connects to a third-party cloud service using its proprietary protocol. The third-party device cloud connects to Internet of Things Cloud Service using the HTTP protocol.
<b>Devices and gateways that are already connected to existing network providers that provide wireless connectivity to the devices.</b>	The network providers can transmit the device data to Internet of Things Cloud Service using HTTP or MQTT.	Network providers use LoRA, NB-IoT, or IoT MVNO for wireless connectivity with devices such as an asset tracker or a telematics gateway. The network provider can transmit the device messages to Internet of Things Cloud Service using HTTP or MQTT.
<b>Existing database with device data on events, time stamps, and alarms.</b>	The on-premises database can connect to Internet of Things Cloud Service using HTTP and transfer the device messages.	Machines using the Historian service save and store messages in a database or multiple databases over the supervisory control and data acquisition (SCADA) network. An on-premises Historian service connects to Internet of Things Cloud Service and provides the machine data for analysis.
<b>Existing on-premises industrial gateway software</b>	On-premises industrial gateway software applications manage machine data. The Internet of Things Cloud Service can connect to these gateway applications using HTTP to obtain the machine data.	Machines that use OPC Unified Architecture (UA), a machine-to-machine protocol to transfer machine data to industrial gateway software and that, in turn, can connect to the Internet of Things Cloud Service and send device messages received from the machines.





# Thank You

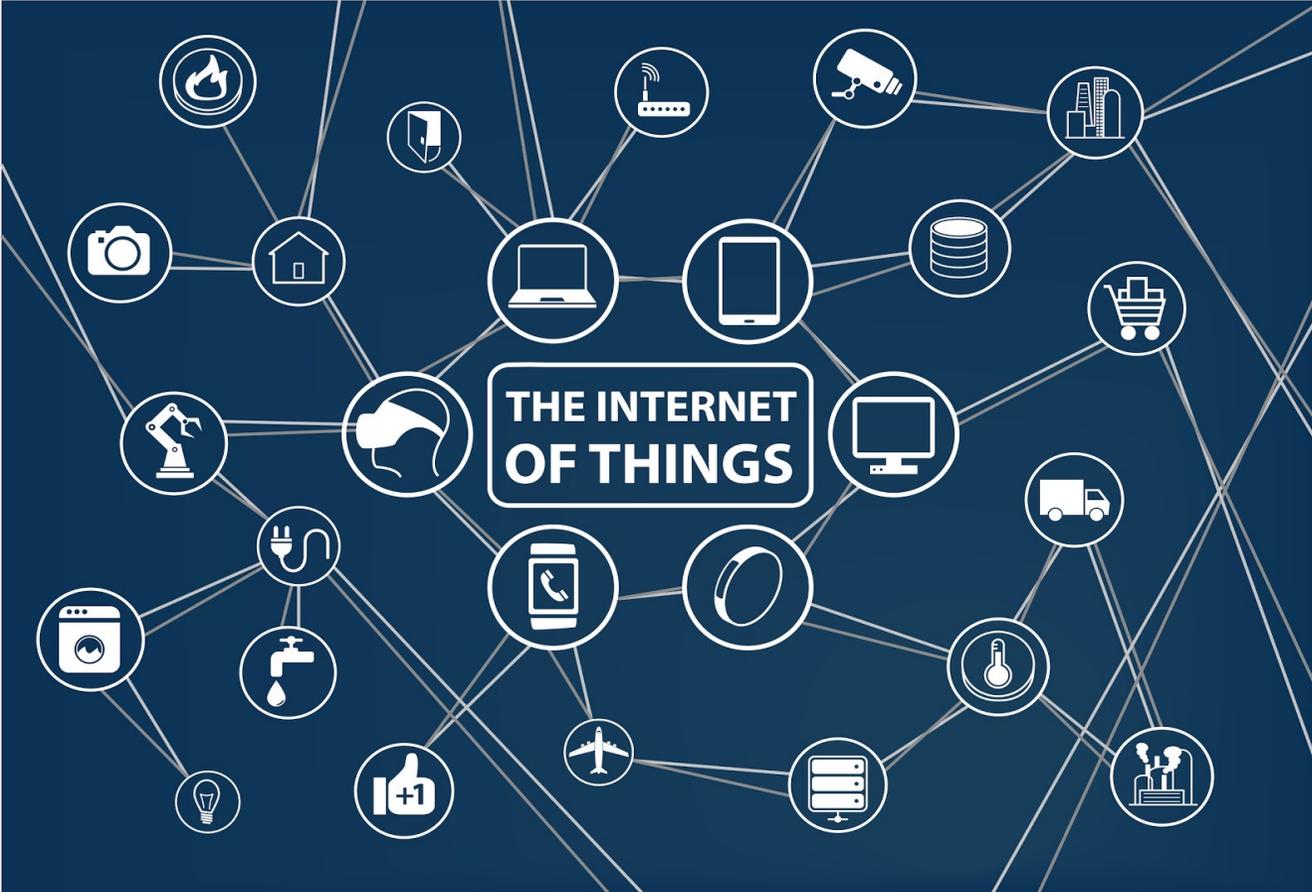
 Prof. Fabrizio Granelli

 +39 0461 282062

 [fabrizio.granelli@unitn.it](mailto:fabrizio.granelli@unitn.it)

 <http://www.unitn.it>

**PROCEED**  
Introducing Recent Electrical Engineering  
Developments Into Undergraduate Curriculum



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

*This publication was produced with the financial support of the European Union. Its contents are the sole responsibility of the partners of IREEDER project and do not necessarily reflect the views of the European Union*



Co-funded by the  
Erasmus+ Programme  
of the European Union

**Prof. Fabrizio Granelli**

**Dr. Claudio Sacchi**

# Introduction to the Internet of Things

Lecture 9: Data Storage and Cloud Systems

# This week's topics...

- Introduction
- Cloud Computing Basics
- Processing in Internet of Things Services
- Data Storage

# Section Outline

Introduction



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

# Section 1

Introduction

---

# Internet of Things Data Storage & The Cloud

## Introduction

- IoT and cloud are nowadays two related internet technologies, which go hand-in-hand in IoT deployments.
- Several modern massive IoT ecosystems are based on the cloud support.
- Nevertheless, before describing how IoT and cloud computing can be integrated, we will introduce some basic cloud computing concepts.

# Section Outline

Introduction

Cloud Computing Basics



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

## Section 2

Cloud Computing Basics

# Internet of Things Data Storage & The Cloud

## Cloud Computing Basics

- Cloud computing represented an evolutionary step in Internet-based computing, providing the means for delivering ICT resources as a service.
- The ICT resources that can be delivered through cloud computing model:
  - computing power
  - computing infrastructure (e.g., servers and/or storage resources)
  - Applications
  - business processes and more.
- Cloud services can be offered through:
  - infrastructures (clouds) that are publicly accessible (i.e. public cloud services)
  - privately owned infrastructures (i.e. private cloud services)
  - hybrid cloud services (i.e. both public and private clouds)

# Internet of Things Data Storage & The Cloud

## Cloud Computing Characteristics

- Elasticity and scalability
  - Cloud computing services can scale upwards during high periods of demand and downward during periods of lighter demand
- Self-service provisioning and automatic deprovisioning
  - In cloud computing, both provisioning and de-provisioning of resources represented automated functionalities.
- Application programming interfaces (APIs)
  - Cloud services are accessible via APIs, which enable applications and data sources to communicate with each other.

# Internet of Things Data Storage & The Cloud

## Cloud Computing Characteristics

- Billing and metering of service usage in a pay-as-you-go model
  - They provide the means for metering resource usage and subsequently issuing bills
- Performance monitoring and measuring
  - Cloud computing infrastructures provide a service management environment along with an integrated approach for managing physical environments and IT systems
- Security
  - Cloud computing infrastructures offer security functionalities towards safeguarding critical data and fulfilling customers' compliance requirements

# Internet of Things Data Storage & The Cloud

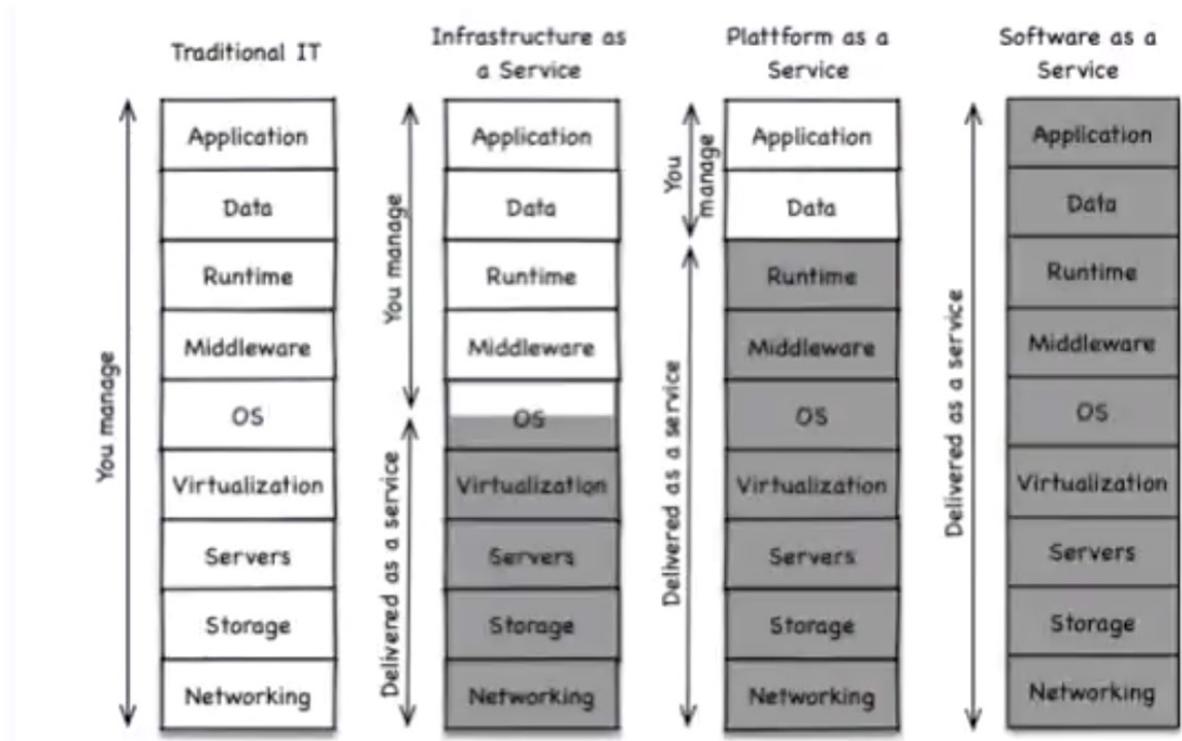
## Cloud Computing Service Delivery Models

- Infrastructure as a Service (IaaS)
  - IaaS deals with the delivery of storage and computing resources towards supporting custom business solutions
  - E.g. Amazon's Elastic Compute Cloud (EC2) w/ Xen hypervisor to create and manage virtual machines.
- Platform as a Service (PaaS)
  - PaaS provides development environments for creating cloud-ready business applications
  - It provides a deeper set of capabilities comparing to IaaS, including development, middleware, and deployment capabilities
  - E.g. Google's App Engine and Microsoft's Azure cloud environment
- Software as a Service (SaaS)
  - SaaS services enable access to purpose-built business applications in the cloud

# Internet of Things Data Storage & The Cloud

## Cloud Computing Service Delivery Models

### Cloud Service Models



# Section Outline

On-device Processing

Gateway Processing

Cloud Processing



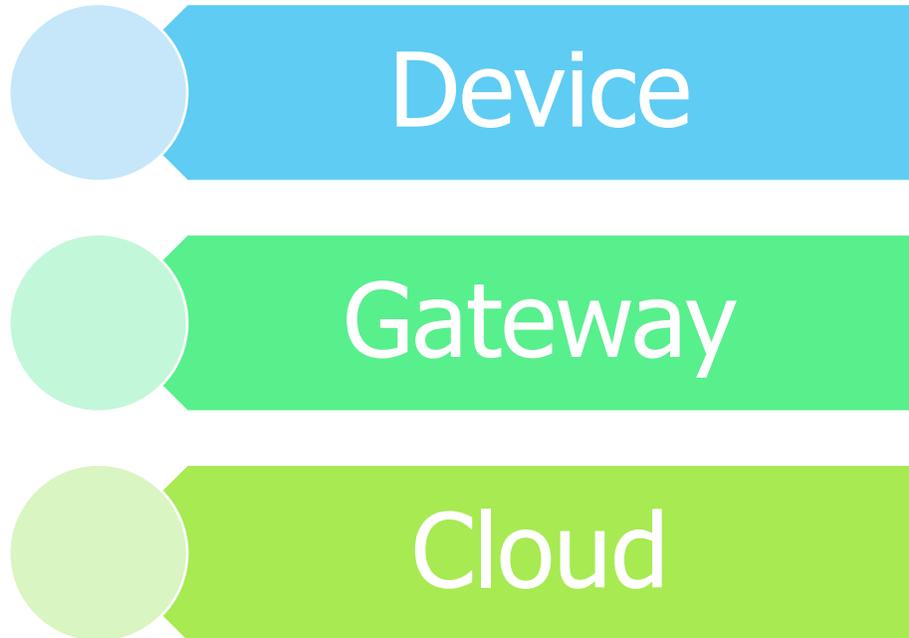
[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

## Section 3

Processing in IoT Services

# Internet of Things Data Storage & The Cloud

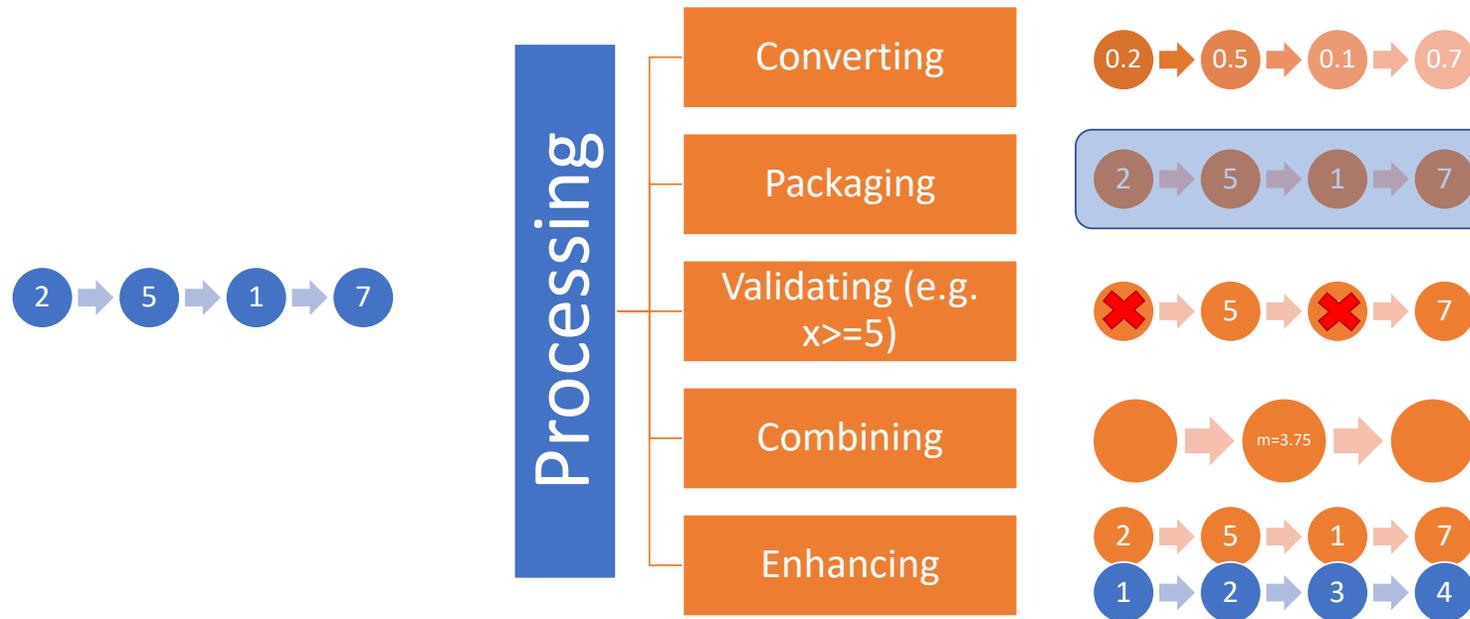
## Processing - Introduction



- Devices connect to a network to communicate with each other, or to centralized applications. Devices might be directly or indirectly connected to the internet.
- A gateway enables devices that are not directly connected to the Internet to reach cloud services.
- The data from each device is sent to the *cloud*, where it is processed and combined with data from other devices.

# Internet of Things Data Storage & The Cloud

## On-device Processing



# Internet of Things Data Storage & The Cloud

## Gateway Processing

- A gateway manages traffic between networks that use different protocols.
- It is responsible for protocol translation and other interoperability tasks.
- An IoT gateway device is sometimes employed to provide the connection and translation between devices and the cloud.
- A gateway device might act as a proxy, receiving data from devices and packaging it for transmission over TCP/IP.

# Internet of Things Data Storage & The Cloud

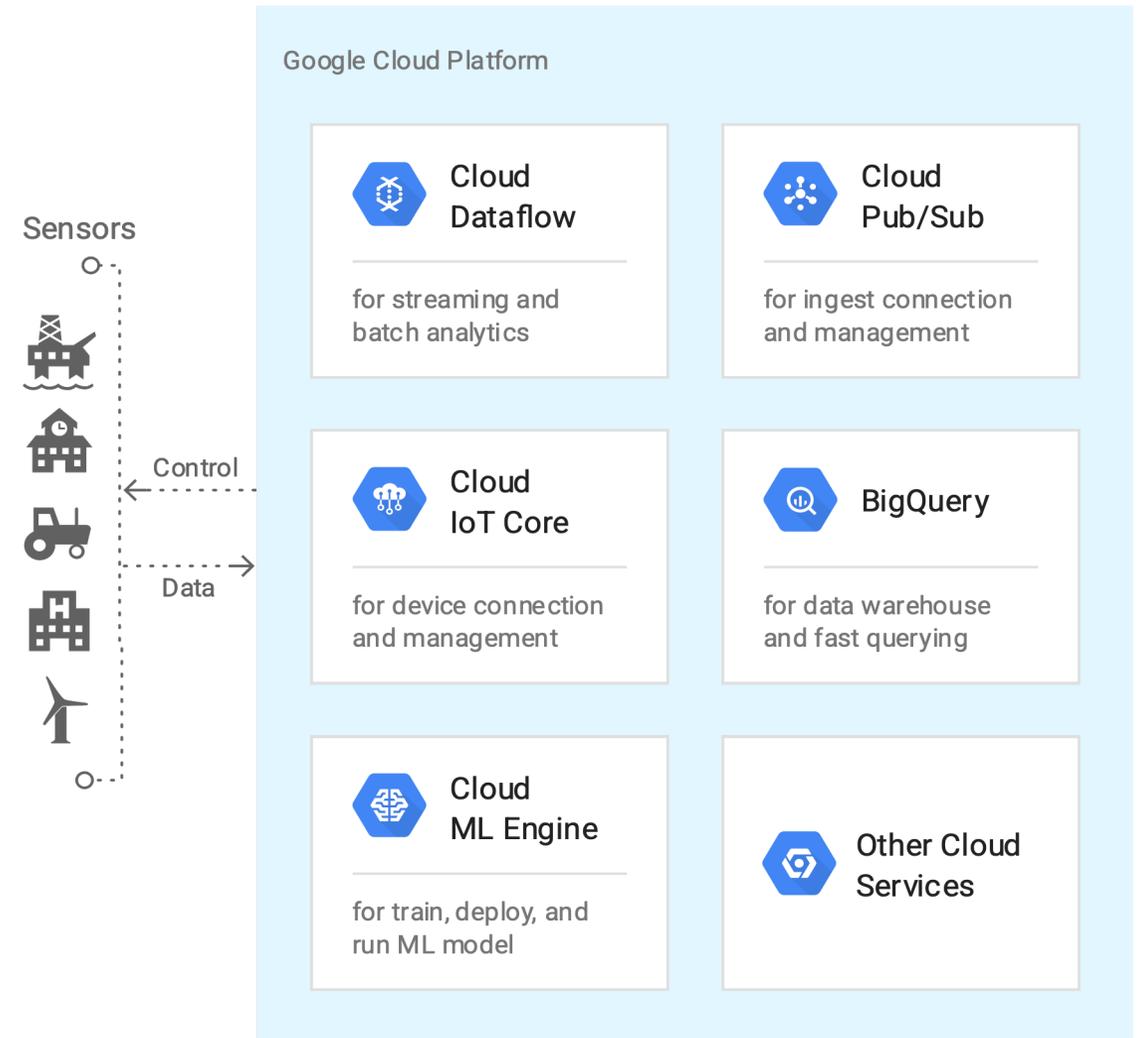
## Gateway Processing

- The gateway provides processing of the data across multiple devices before it is sent to the cloud.
- The following tasks would likely be relegated to a gateway device:
  - Condensing data to maximize the amount that can be sent to the cloud over a single link
  - Storing data in a local database, and then forwarding it on – when the connection to cloud is intermittent
  - Providing a real-time clock, with a battery backup, used to provide a consistent timestamp for devices that can't manage timestamps well or keep them well synchronized
  - Performing IPV6 to IPV4 translation
  - Ingesting and uploading other flat-file-based data from the local network that is relevant and associated with the IoT data
  - Acting as a local cache for firmware updates

# Internet of Things Data Storage & The Cloud

## Cloud Processing

- When it comes to storing, processing, and analyzing data, especially big data, the cloud represents the most appropriate location
- Example by Google Cloud



Build & train ML models in the cloud

# Internet of Things Data Storage & The Cloud

## Cloud Processing

### ○ Dataflow:

- Dataflow provides a programming model as a managed service for processing data in multiple ways, including batch operations, extract-transform-load (ETL) patterns, and continuous, streaming computation.
- Dataflow can be particularly useful for managing the high-volume data processing pipelines required for IoT scenarios.

# Internet of Things Data Storage & The Cloud

## Cloud Processing

### ○ IoT Core:

- The cloud will typically offer a native service for collecting and managing IoT data
- The Cloud IoT Core provides a secure MQTT (Message Queue Telemetry Transport) broker for devices managed by the IoT Core
- The IoT Core MQTT broker directly connects with the Pub/Sub service

# Internet of Things Data Storage & The Cloud

## Cloud Processing

### ○ Pub/Sub:

- Pub/Sub (Publish/Subscribe) provides a globally durable message ingestion service
- By creating topics for streams or channels, it is possible to enable different components of the IoT application to subscribe to specific streams of data without needing to construct subscriber-specific channels on each device
- Pub/Sub can act like a shock absorber and rate leveller for both incoming data streams and application architecture changes

# Internet of Things Data Storage & The Cloud

## Cloud Processing

- Cloud monitoring and logging
- Pipeline processing tasks:
  - Transforming data
  - Aggregating data and computing
  - Enriching data
  - Moving data
- Analytics
  - Performing analytics on data obtained through IoT sources is often the entire purpose of instrumenting the physical world

# Section Outline

IoT and Big Data

SQL Databases

NoSQL Databases

Time Series Databases



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

## Section 4

Data Storage

# Internet of Things Data Storage & The Cloud

## Introduction to Data Storage

- Managing and organizing access to a massive amount of data represents one of the key issues in the design of IoT applications
- IoT is about data, services, connectivity: data are gathered by objects, transferred, analyzed, and translated into services
- Large-scale IoT deployments can produce huge amounts of data, leading to the concept of *big data*.
- The word *big-data* is currently used to identify: (i) data sources with specific characteristics, as well as (ii) novel technologies to manage huge amounts of data.

# Internet of Things Data Storage & The Cloud

## Big Data Features

- Volume
  - in the order of Petabytes
- Velocity
  - data is produced at high rate
- Variety
  - big data is heterogeneous (text, image, video, etc)
- Value
  - valuable information is stored within such huge “container” and it can be extracted using proper IT techniques

# Internet of Things Data Storage & The Cloud

Example of Big Data / IoT Application

- Energy@Home project by Telecom Italia
- <http://www.energy-home.it/>
- Aimed at collecting data by smart plugs installed in customers' houses for
  - implementing data classification (i.e. identify the appliance consuming power)
  - profiling (i.e. identify users' habits)
  - prediction (i.e. predict energy consumption)
  - scheduling (i.e. intelligent ON/OFF schedule)

# Internet of Things Data Storage & The Cloud

## IoT Data as Time Series

- The unifying characteristic of IoT data is that they represent a time-series: a sequence of timestamp plus values
- This has the following implications:
  - Data are immutable.
  - Writing data is done by “appending” to previous values.
  - Reading is performed for contiguous sequences of samples data.
  - Data is highly compressible.
  - Deleting usually happens across large time period.
  - High precision might be desired for a short period of time, but single values are not so important.

# Internet of Things Data Storage & The Cloud

## SQL Databases

- Relational Database Management Systems (RDBMS) are a family of databases based on a relational model
- They are also called SQL databases, since they employ the SQL querying language.

Time	Time series ID	Value
<b>16:17:00</b>	101	10.1
<b>16:17:05</b>	102	0.5
<b>16:17:10</b>	103	2.3
<b>16:17:15</b>	104	1.2

# Internet of Things Data Storage & The Cloud

## SQL Databases

- The main problems of SQL databases for IoT are:
  - Scalability (i.e. need to store large amount of time-series data)
  - Performance (e.g. support for range-based operations)

# Internet of Things Data Storage & The Cloud

## NoSQL Databases

- NoSQL databases represent a set of tools and logic models, alternative or complementary to RDBMS
- They do not employ the SQL language and provide a scheme-less (un/semi-structured) database
- Most popular NoSQL databases for IoT services are: Redis, Cassandra, MongoDB, Couchbase and Neo4j

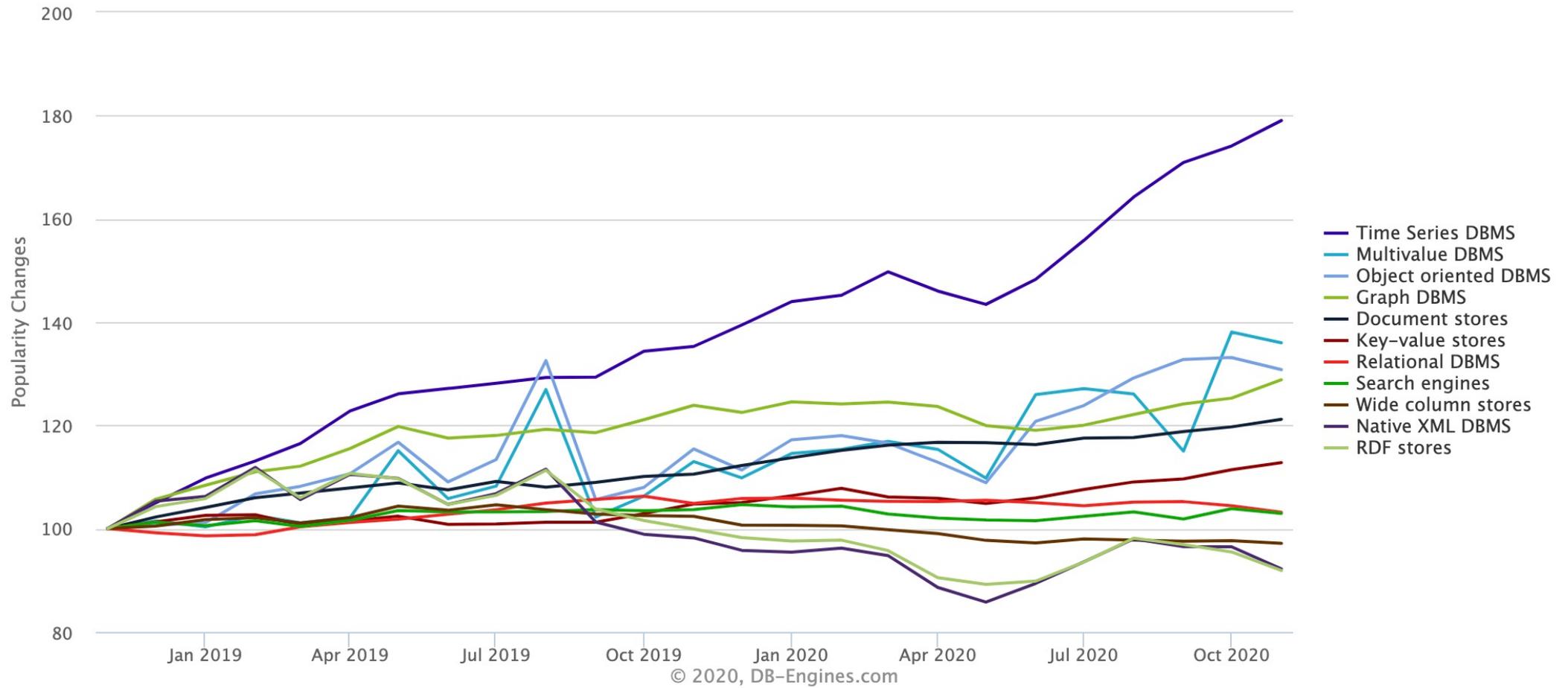
# Internet of Things Data Storage & The Cloud

## Time Series Databases

- Time-series Databases are dedicated DBMS optimized for managing large volumes of time-series data
- They provide:
  - Optimized data storage and sharding
  - Operational support (e.g. range-based queries)
  - Time-granularity management
  - Time-series analytics and mining

# Internet of Things Data Storage & The Cloud

## Time Series Databases



# Internet of Things Data Storage & The Cloud

## Time Series Databases - InfluxDB

- InfluxDB is an open-source time-series database (InfluxData)
  - It supports an SQL-like query language (InfluxQL) (<1.8) and a JS-like query language (Flux) (>=1.8)
  - It provides GUI, Command Line Interface (CLI) and HTTP APIs
  - It supports distributed deployments
  - Easy integration with time-series tools for data acquisition, analytics and visualization (e.g. Telegraf, Grafana).

# Internet of Things Data Storage & The Cloud

## Time Series Databases – InfluxDB parameters

- Time-Structured Merge Tree (TSM): data structure used to contain sorted, compressed series data
- Time Series Index (TSI): it can address millions of unique time series, regardless of the amount of memory on the server hardware
- Automatic downsampling and data retention procedures.
- Timestamp: in RFC3339 UTC format (yyyy-mm-ddThh:mm:ssZ)
- Field keys: string metadata, similar to column name
- Field values: actual measured data
- Tag-sets: optimal, extra-information about the measurements
  - Tag keys: string meta-data, similar to field keys
  - Tag values: string values

# Internet of Things Data Storage & The Cloud

## Time Series Databases

- Measurement: Container to hold the timestamps, fields and tags (similar to a table in a RDBMS)
- Buckets: collection of data-points, containing:
  - Measurement
  - Tag-sets
  - Retention policy

## ○ What we learned:

- Cloud Computing Basics
- Processing in Internet of Things Services
- Data Storage



[This Photo](#) by Unknown Author is licensed under [CC BY-NC](#)

# Summary

## Data Storage and Cloud Systems

---



# Thank You

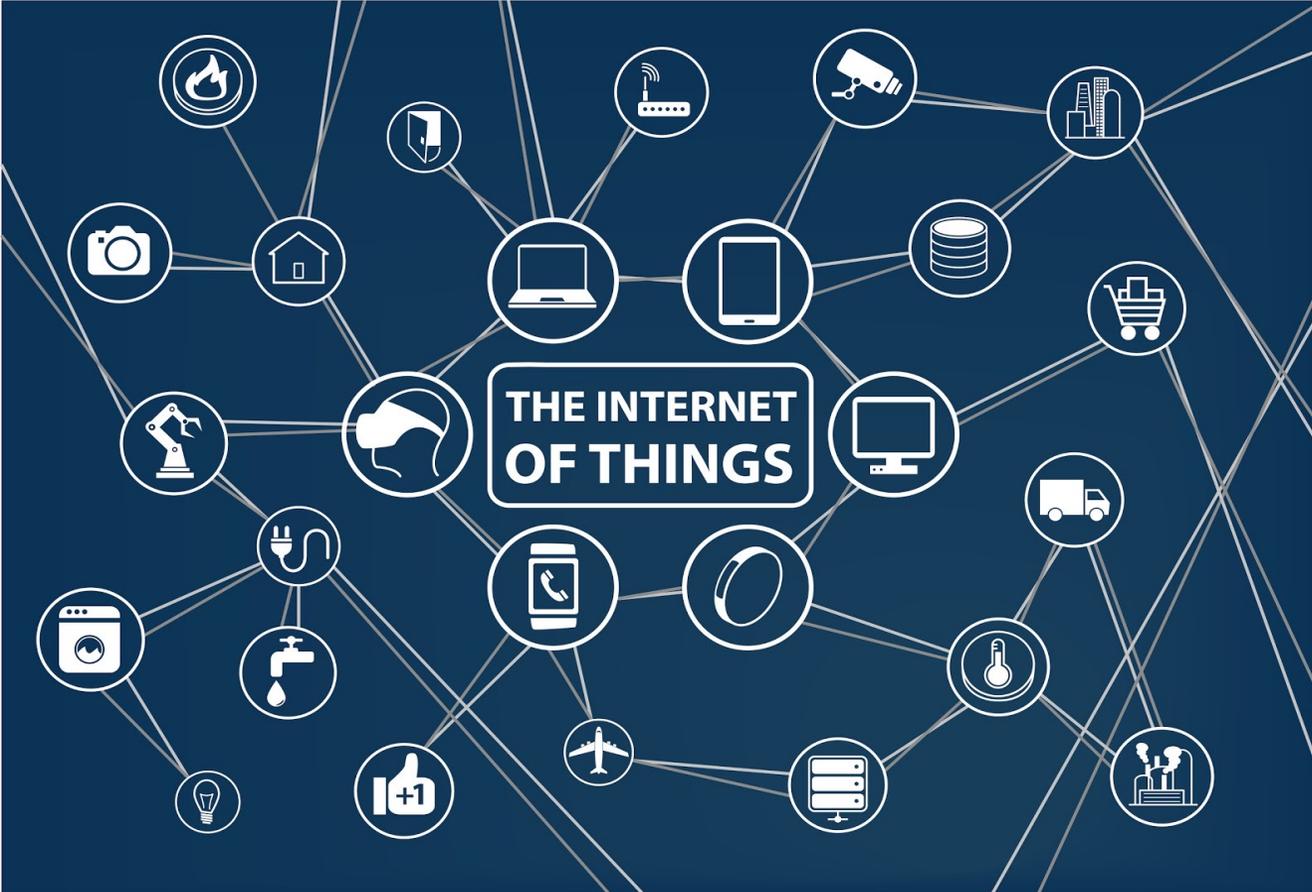
 Prof. Fabrizio Granelli

 +39 0461 282062

 [fabrizio.granelli@unitn.it](mailto:fabrizio.granelli@unitn.it)

 <http://www.unitn.it>

**PROCEED**  
Introducing Recent Electrical Engineering  
Developments Into Undergraduate Curriculum



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

*This publication was produced with the financial support of the European Union. Its contents are the sole responsibility of the partners of IREEDER project and do not necessarily reflect the views of the European Union*



Co-funded by the Erasmus+ Programme of the European Union

**Prof. Fabrizio Granelli**

**Dr. Claudio Sacchi**

# Introduction to the Internet of Things

Lecture 10: Data Analytics and Applications

# This week's topics...

- Introduction
- Interpretation of IoT Data
- Visualization of Data
- A Case Study

# Section Outline

Introduction



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

# Section 1

Introduction

---

# Internet of Things Data Analytics and Applications

## Introduction

- There is no specific methodology to solve Data Science for IoT (IoT Analytics) problems:
  - A Data Science for IoT problem is a typical Data Science problem.
  - IoT is unique in the use of hardware, high data volumes, impact of verticals (see later), impact of streaming data etc.
- A methodology for IoT analytics (Data Science for IoT) should cover the unique aspects of each step in Data Science.
- It is more than the choice of the model family.

# Internet of Things Data Analytics and Applications

## Introduction

- The choices include:
  - Choice of the model structure - optimisation methodology (Bootstrap, etc)
  - Choice of the model parameter optimisation algorithm (joint gradients vs. conjugate gradients)
  - Preprocessing of the data (reduction, functional reduction, log-transform, etc.)
  - How to deal with missing data (case deletion, interpolation, etc.)
  - How to detect and deal with suspect data (distance-based outlier detection, density-based, etc.)
  - How to choose relevant features (filters, wrappers, etc.)
  - How to measure prediction performance (mean square error, mean absolute error, misclassification rate, precision/recall, etc.)

# Internet of Things Data Analytics and Applications

## Introduction

- A proper methodology would need to consider the unique aspects of IoT:
  - Complex event processing
  - Deep learning
  - Anomaly Detection
- In addition, IoT vertical domains (Smart Grid, Smart cities, Smart Energy, Automotive, Smart factory, Mobile, Wearables, Smart Home etc.) can introduce further choices or constraints on the data analytics process.

# Section Outline

Interpretation of data



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

## Section 2

Interpretation of data

---

# Interpretation of IoT Data

## Introduction

- Interpretation of the data is an extremely relevant issue in IoT applications.
- IoT applications will typically generate huge amounts of data.
- It is necessary to identify anomaly or extract information to make the application significant and useful to the clients.

# Interpretation of IoT Data

## Kalman Filters and Sensors Fusion

- Kalman filters and sensor fusion are common methodologies that can be used to enable interpretation of IoT data.
- Kalman filtering is an algorithm that uses a series of measurements observed over time, and it produces estimates of unknown variables that tend to be more precise than those based on a single measurement alone.
- Kalman filters are used in Sensor fusion.
- Sensor fusion helps to determine the state (and also the overall context) of an IoT based computing system which relies on inferring the combined meaning from different sensors.

# Interpretation of IoT Data

## Kalman Filters and Sensors Fusion

- In IoT applications:
  - The system state cannot be measured directly
  - The application can collect multiple measurements – each of which might not be 100% accurate. Then, those need to be fused/combined.
  - Fusion across different sensors can mean different things: e.g. several temperature sensors, fusion across different attributes, fusion across different domains, fusion across different time (sampling over space and time)
- Note: sensor fusion is not merely 'adding' values, i.e. not just adding temperatures.
  - It is more about understanding the overall 'State' of a system based on multiple sensors.

# Interpretation of IoT Data

## Kalman Filters and Sensors Fusion

- Kalman filters are used to predict the state of dynamic systems.
- Dynamic systems are systems that change or evolve in time according to a fixed rule.
  - For many physical systems, this rule can be stated as a set of first-order differential equations.
  - This set of input, output and state variables related by first-order differential equations is called a state-space representation of a physical system.
- In some cases, we might be interested in estimating the covariance between the two measurements.

# Interpretation of IoT Data

## Kalman Filters and Sensors Fusion

### ○ To summarise:

- Sensors can generate fragments of context information with varying degrees of confidence that feed into the overall context estimation of the state.
- Typical sensed information from individual sensors will include the sensor's confidence level and timestamp.
- The Kalman filter averages a prediction of a system's state with a new measurement using a weighted average. Values with smaller estimated uncertainty are trusted more. The relative weights between the sensor inputs and their impact on the overall state are calculated using covariance. This gives a new state of the system.
- This process is repeated every time step, with the new estimate and its covariance informing the prediction used in the following iteration.

# Section Outline

Visualization of data

Principles

Sample tools



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

## Section 3

Visualization of data

# Visualization of IoT Data

## Introduction

- Efficient visualization of data might be relevant in an IoT scenario for the following reasons:
  - It helps to make real-time decisions with the combination of multiple data sources into a single insightful dashboard with multi-layered visual data.
  - It combines the new IoT data transmitted from data sensors with the existing data to analyze and bring light to new business opportunities.
  - It supports to monitor IoT devices and infrastructure for better performance on IoT data flow.
  - It helps to analyze multiple data correlations in real-time.

# Visualization of IoT Data

## Grafana

- Grafana, an open-source data visualization tool, is built to consume time-series data.
- Grafana tool provides a visual dashboard which covers multiple functionalities all under single and powerful interface.
- Several panels in the grid are responsible for analyzing the data and presents the data in different visual formats like:
  - heat maps
  - geo maps
  - Histograms
  - charts – pie/bar
  - graphs.

# Visualization of IoT Data

## Grafana



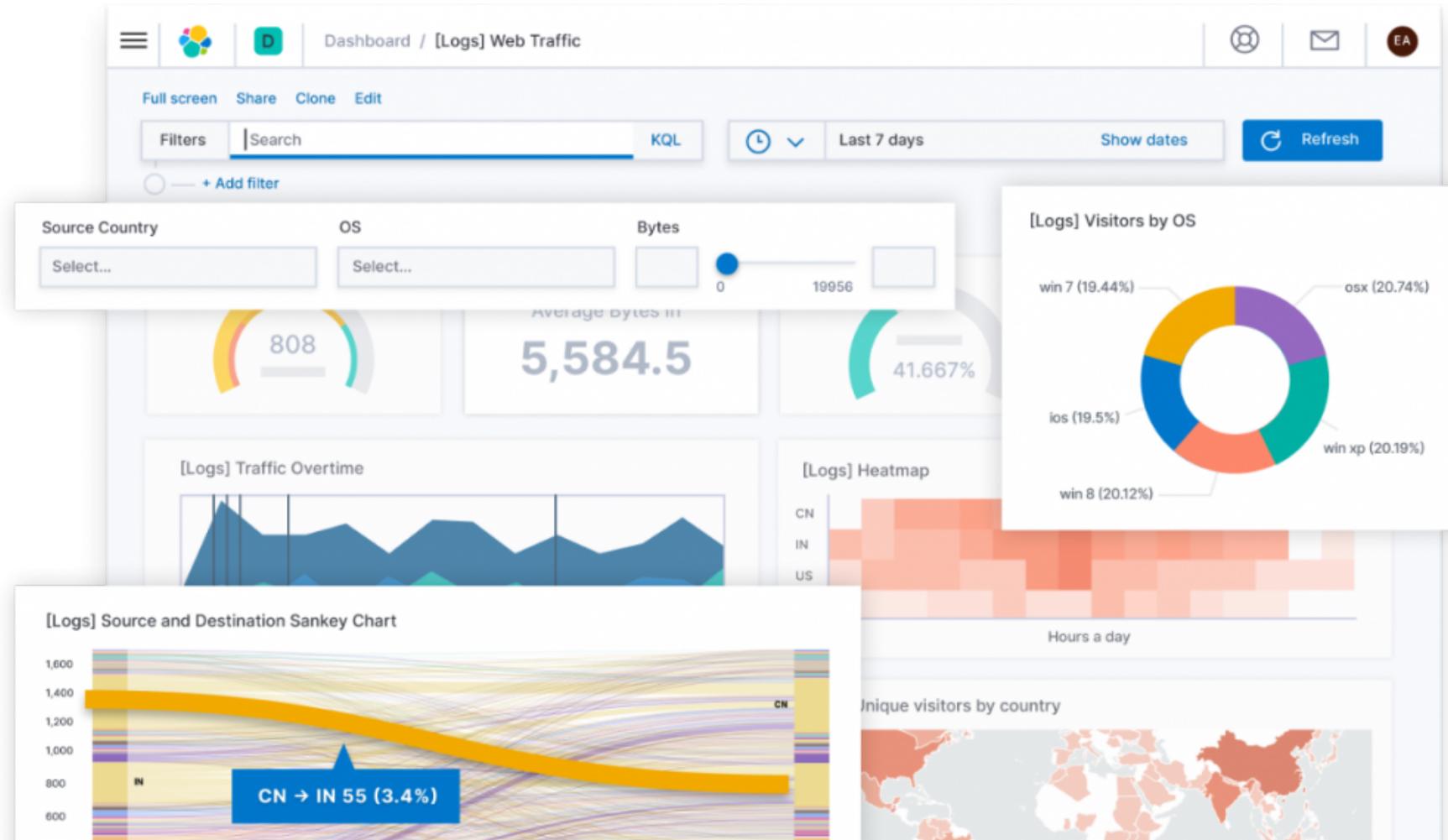
# Visualization of IoT Data

## Kibana

- Kibana is an open source data visualization tool for analyzing large volumes of log data.
- In the Kibana workflow:
  - the logstash is responsible to collect all the data from the various remote sources
  - these data logs are then pushed and sent to the Elasticsearch
  - Elasticsearch acts as the database to the kibana tool with all the log information
  - Kibana tool presents these log data in the form of pie charts, bar or line graphs to the user.

# Visualization of IoT Data

## Kibana



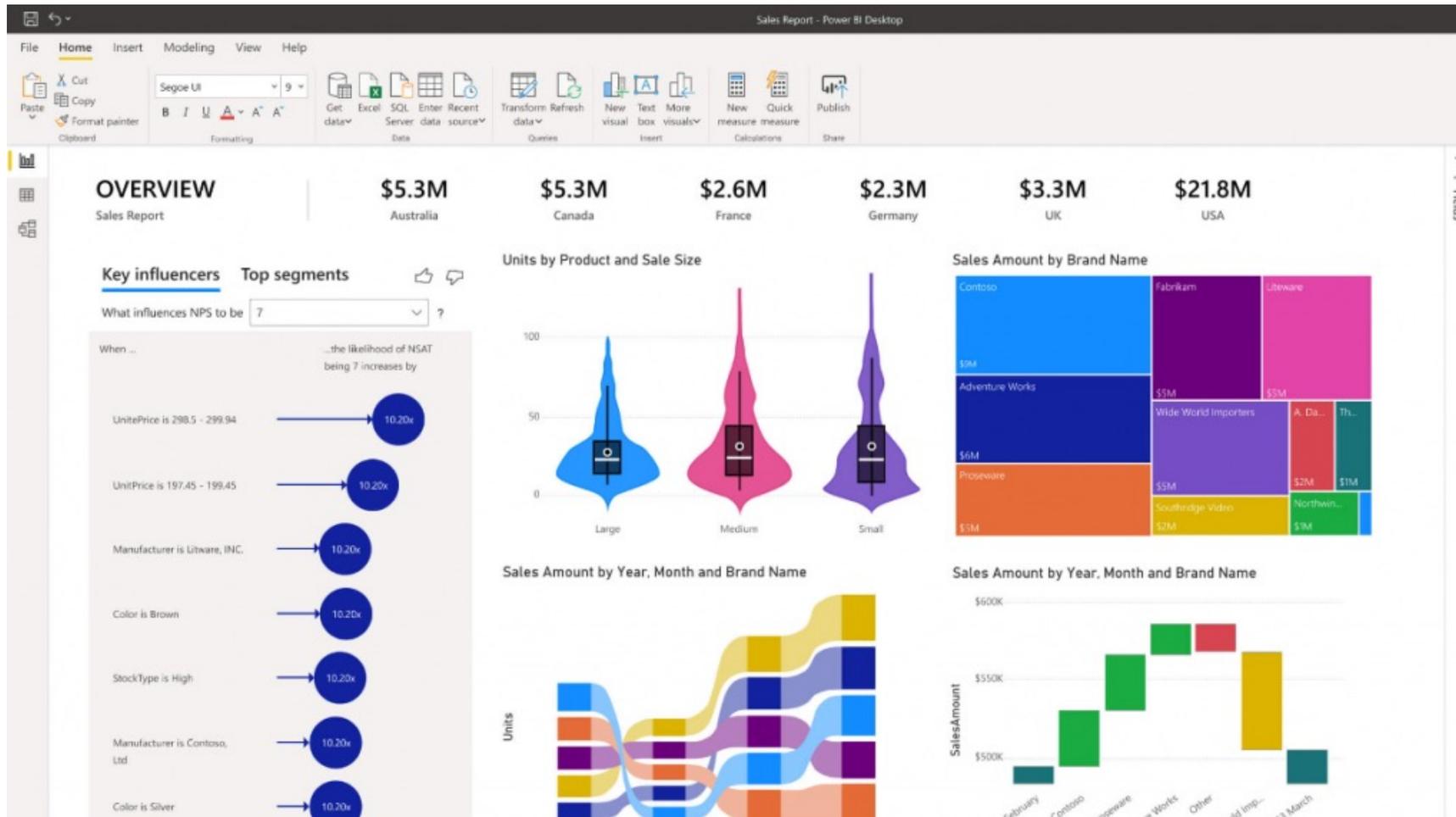
# Visualization of IoT Data

## Power BI

- Microsoft PowerBI is a popular Business Intelligence Tool.
- The workflow is as follows:
  - First, the data is collected from the external data sources (Facebook, Google Analytics, Azure Cloud, Salesforce, etc.)
  - Using Power BI, you can create visualization in 2 ways
    - *one is by adding from the right-side panel to the report canvas which is in a table type visualization format, or*
    - *by simple drag and drop of value axis under visualization.*
  - Once the report is developed, it can be published to web portal with the help of Power BI service.
  - The report can be exported it in pdf, excel or any preferred format.

# Visualization of IoT Data

## Power BI



# Section Outline

A simple sensors -> broker -> apps IoT scenario



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

## Section 4

A Test Case

# An IoT Test Case

## Scenario Description

- We have one or many sensors, an MQTT broker and one or more applications.
- Sensors can measure any relevant information from the environment (e.g. temperature, humidity, etc.) and provide data in input to the broker.
- IoT applications subscribe to the data flows of interest and then analyze and process the corresponding streams of data.

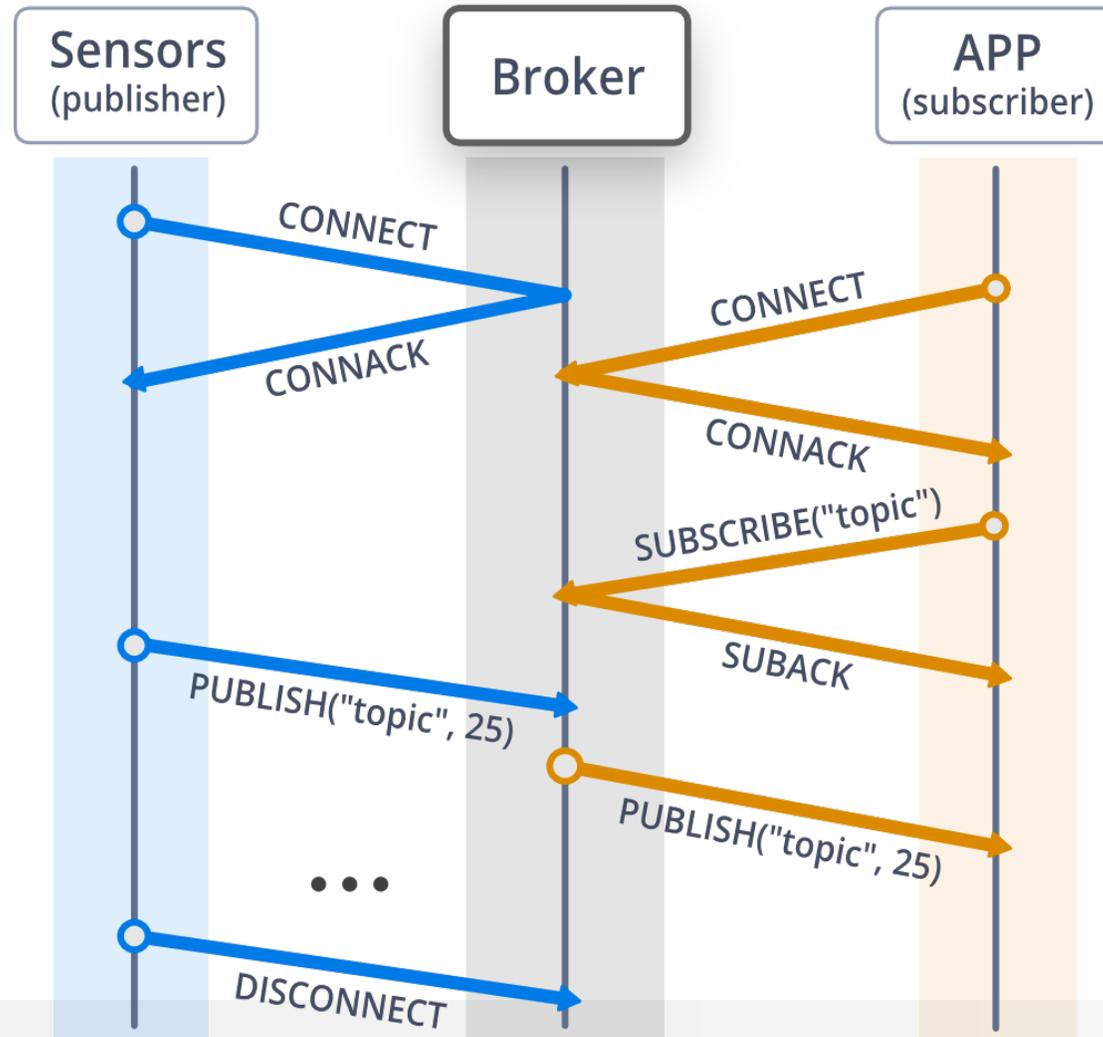
# An IoT Test Case

## The MQTT Broker

- The MQTT Broker collects data from the sensors (publishers) and provides them to subscribers.
- Subscriber apps can subscribe to multiple topics and receive the corresponding data flows.
- The MQTT protocol runs over TCP/IP and typically on port 1883, which is assigned by the Internet Assigned Numbers Authority (IANA). For using MQTT over SSL, port 8883 is used.

# An IoT Test Case

## Messages Exchange







# Thank You

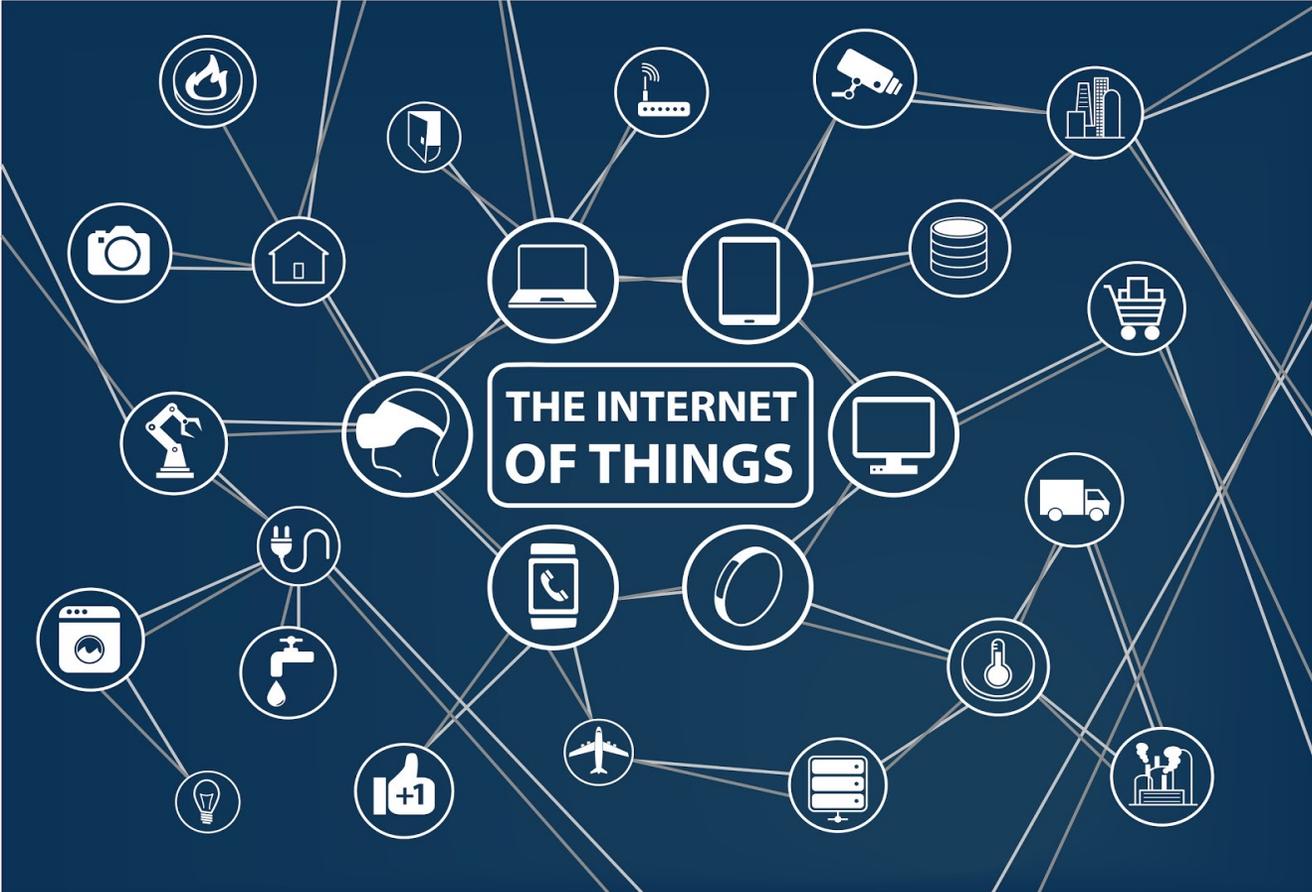
 Prof. Fabrizio Granelli

 +39 0461 282062

 [fabrizio.granelli@unitn.it](mailto:fabrizio.granelli@unitn.it)

 <http://www.unitn.it>

**PROCEED**  
Introducing Recent Electrical Engineering  
Developments Into Undergraduate Curriculum



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

*This publication was produced with the financial support of the European Union. Its contents are the sole responsibility of the partners of IREEDER project and do not necessarily reflect the views of the European Union*



Co-funded by the Erasmus+ Programme of the European Union

**Ms. Maria Papaioannou**

# Introduction to the Internet of Things

Lecture 11: IoT Security and security standards

# This week's topics...

- The Internet of Things (IoT) – An Overview
- Baseline Security Recommendations for IoT
- Guidelines for Securing the Internet of Things: Secure supply chain for IoT
- Good Practices for Security of IoT: Secure Software Development Lifecycle

Based on:

- "Computer Security: Principles and Practice", 4/e, by William Stallings and Lawrie Brown
- "Baseline Security Recommendations for IoT in the context of CII\_FINAL", ENISA, 2017
- "ENISA Report - Guidelines for Securing the Internet of Things", ENISA, 2020
- "Good practices for security of IoT", ENISA, 2019

# Section Outline

- Things on the Internet of Things
- Evolution
- Components of IoT-enabled things
- IoT and cloud context
- IoT security
  - *The patching vulnerability*
  - *IoT security and privacy requirements defined by ITU-T*
  - *An IoT security framework*



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

## Section 1

The Internet of Things (IoT)

---

# The Internet of Things (IoT)

## Definition (1/2)

- IoT is a term that refers to the expanding interconnection of smart devices, ranging from appliances to tiny sensors
  - In particular, the IoT enables innovative communication forms between
    - *(i) persons and things, and*
    - *(ii) things themselves,*by embedding short-range mobile transceivers into multiple gadgets and everyday items.
  - These days, the Internet, through advanced computing and the cloud systems, supports the interconnection of billions of devices - both industrial and personal objects.

# The Internet of Things (IoT)

## Definition (2/2)

- These objects receive, handle, and deliver sensing data (usually coming from sensor devices), act on their environment, and may alter themselves, in order to overall manage a larger environment or system, such as a factory or a town.
- The IoT is primarily driven by deeply embedded devices
  - These devices are low-bandwidth, low-repetition data capture, and low-bandwidth data-usage appliances that communicate with each other and provide data via user interfaces

# The Internet of Things (IoT)

## Evolution

With reference to the end systems supported, the Internet has gone through roughly four generations of deployment culminating in the IoT:

### Information technology (IT)

PCs, servers, routers, firewalls, and so on, bought as IT devices by enterprise IT people, primarily using wired connectivity

### Operational technology (OT)

Machines/appliances with embedded IT built by non-IT companies, such as medical machinery, SCADA, process control, and kiosks, bought as appliances by enterprise OT people, primarily using wired connectivity

### Personal technology

Smartphones, tablets, and eBook readers bought as IT devices by consumers (employees) exclusively using wireless connectivity and often multiple forms of wireless connectivity

### Sensor/actuator technology

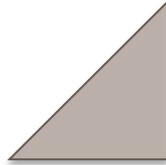
Single-purpose devices bought by consumers, IT and OT people exclusively using wireless connectivity, generally of a single form, as part of larger systems

The fourth generation is marked by using billions of interconnected embedded devices, and thus is typically considered as the IoT itself.

# IoT Components

- Sensor
- Actuator
- Microcontroller
- Transceiver
- Radio-Frequency Identification (RFID)

# Edge



The edge of a classic enterprise network includes a number of IoT-enabled devices. These devices can be sensors and maybe actuators.

- These devices may communicate with one another
- An example would be a cluster of sensors that send their data to another sensor.

A gateway interconnects the IoT-enabled devices with the higher-level communication networks

- Since the communication networks and the IoT devices may use different protocols, the gateway performs the necessary transformations and may also execute a basic data aggregation function.

# Fog (1/4)

- In many IoT deployments, a distributed network of sensors may be capable of producing a large amount of data.
- One method for these data is to keep it permanently (or at least for a long time) in central storage, ensuring that it can be accessed by IoT applications. However, it is sometimes preferred to perform as much data processing close to the sensors as possible.
  - This type of processing is often referred to as processing done at the edge computing level.
- The purpose is the transformation of the sensor data into information suitable for storage and higher-level processing.

# Fog (2/4)

- Processing elements at these levels may deal with high volumes of data and perform data transformation operations, resulting in the storage of much lower volumes of data
- Some of the fog computing operations are described below:

Evaluation

Formatting

Expanding/decoding

Distillation/reduction

Assessment

# Fog (3/4)

- In general, fog computing devices can be found near the edge of the network of IoT sensors and other data-generating devices.
- Fog computing and fog services are some of IoT `s distinct characteristics.
- As a philosophy, fog computing and cloud computing stand in opposite sides.
  - On the one hand, cloud computing is synonymous to massive, centralized storage and processing resources using cloud networking facilities, the storage and resources are available to distributed customers.
  - On the other hand, fog computing entails massive numbers of individual smart objects that communicate with each other through fog networking facilities.

# Fog (4/4)

- Fog computing is capable of addressing those issues using thousands or millions of smart devices, including security, privacy, network capacity constraints, and latency requirements.
- The term “fog computing” originates from the observation that fog tends to hover low to the ground, whereas clouds are high in the sky.

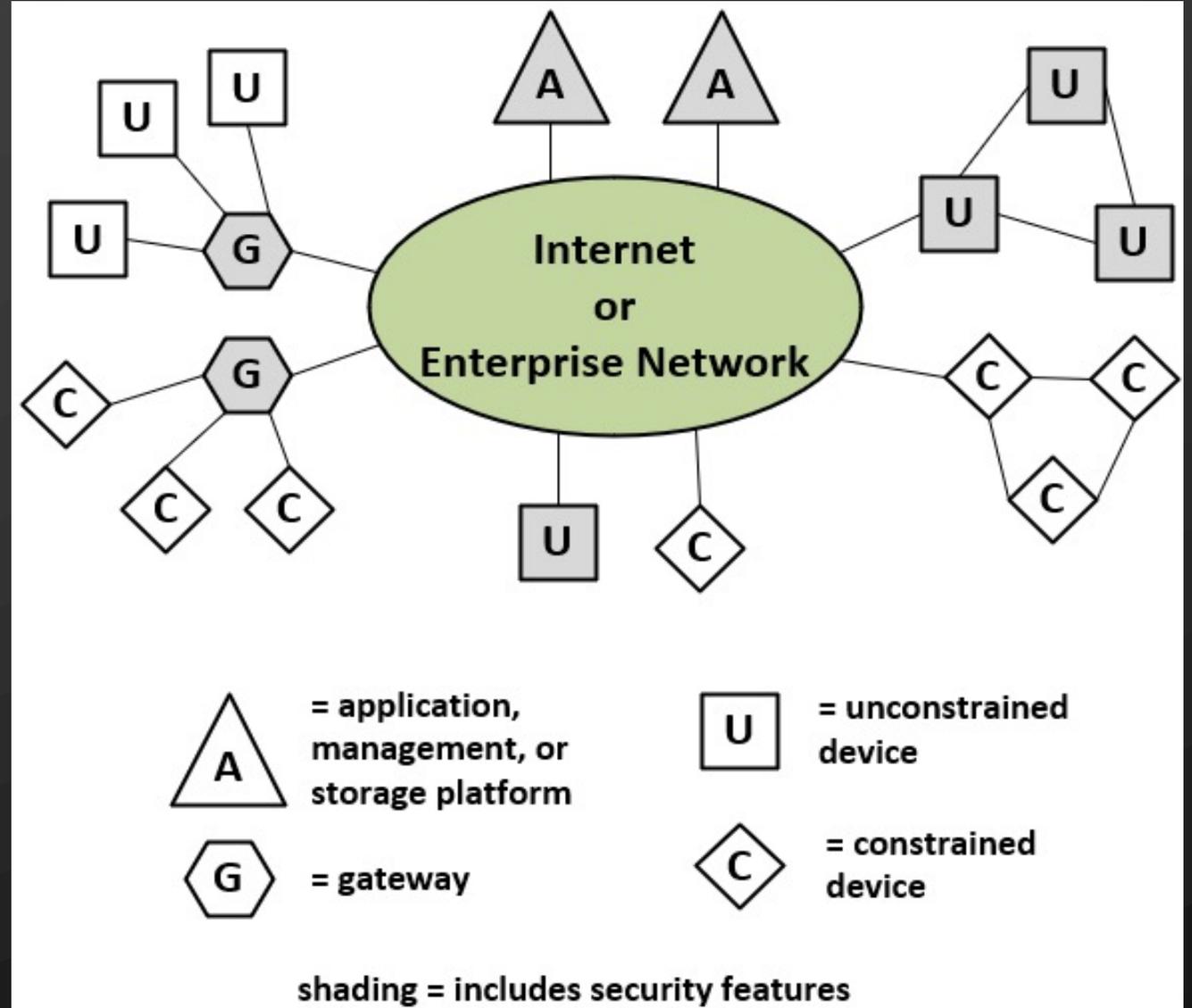
# Core (1/2)

- The core network is often called as a backbone network.
  - Its purpose is the interconnection of geographically dispersed fog networks.
  - Furthermore, it can provide access to networks outside of the enterprise network.
- Typically, in order to achieve its purpose, the core network utilizes very high-performance routers, high-capacity transmission lines, and multiple interconnected routers to increase redundancy and capacity.

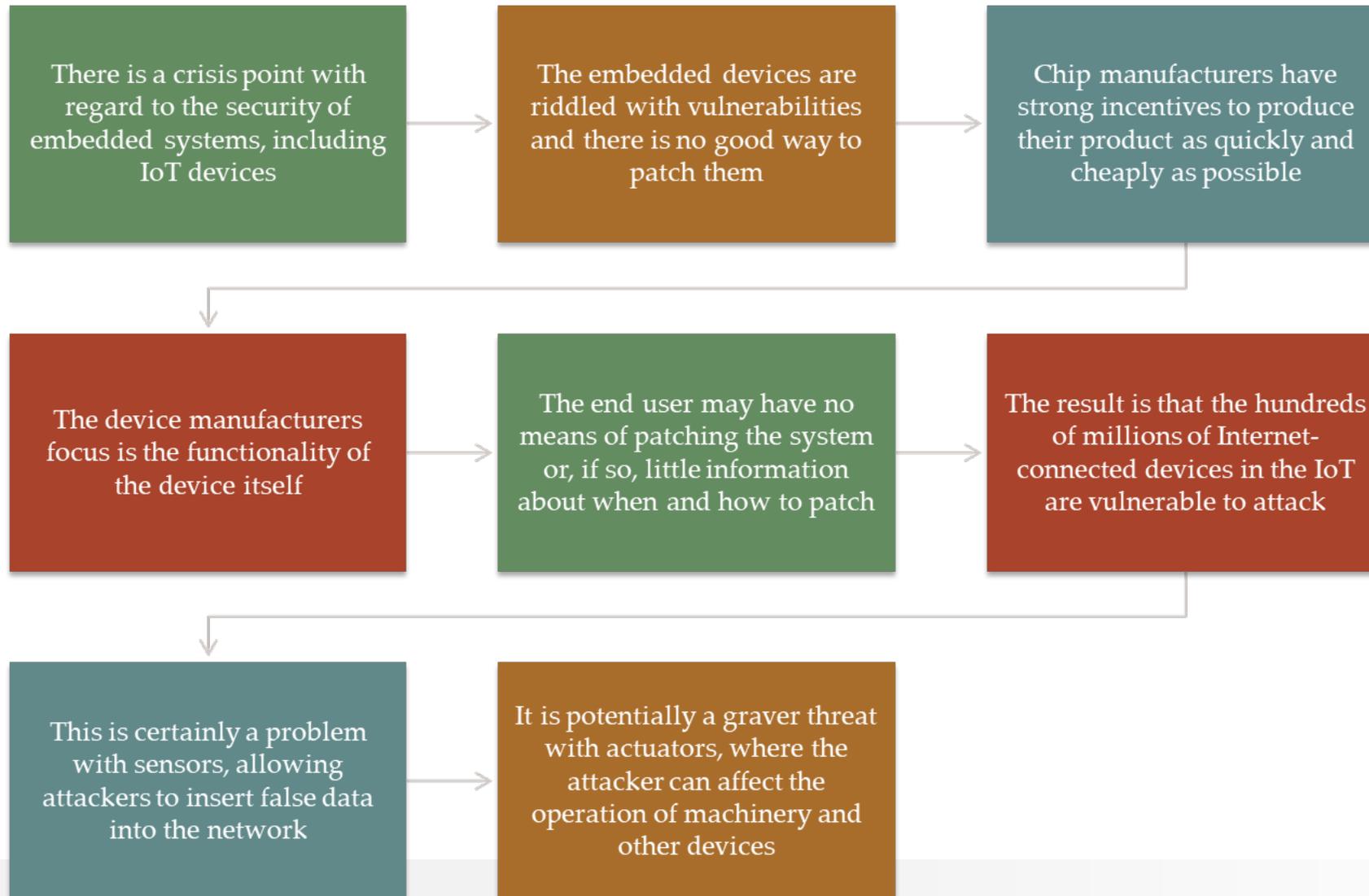
## Core (2/2)

- The increased redundancy is also achieved if a part of the core routers is purely internal without acting as edge routers.
- Additionally, the core network may communicate with high-performance, high-capacity servers such as large database servers and private cloud facilities.

# IoT Security: Elements of Interest



# Patching Vulnerability



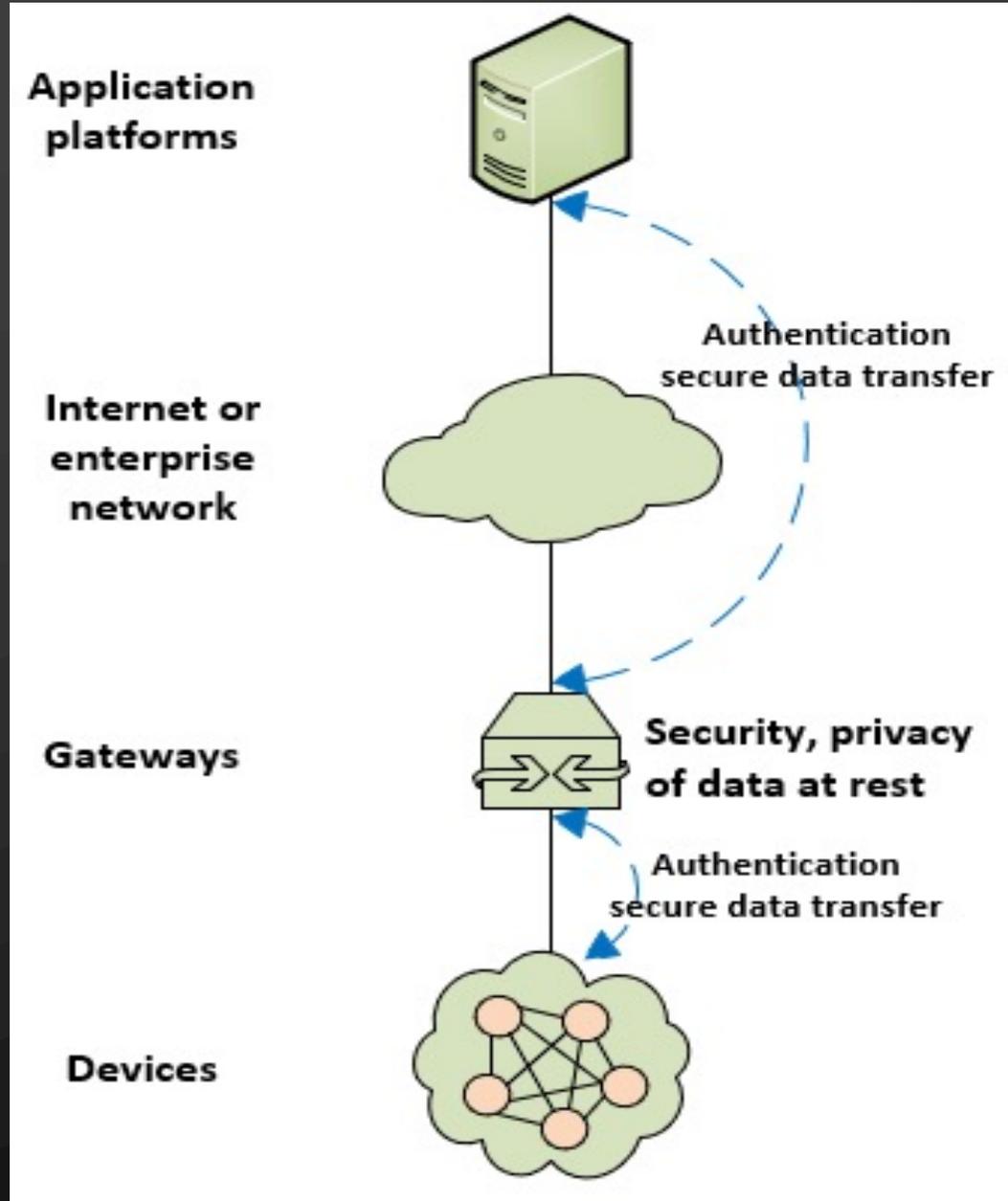
# IoT Security and Privacy Requirements (1/2)

- ITU-T Recommendation Y.2066 includes a list of security requirements for the IoT.
- This list constitutes a useful baseline for understanding the scope of a suitable and efficient security implementation for an IoT deployment.
- These requirements are covering all the functional operations of an IoT system that are capturing, storing, transferring, aggregating, processing the data of things, as well as providing services which involve things.

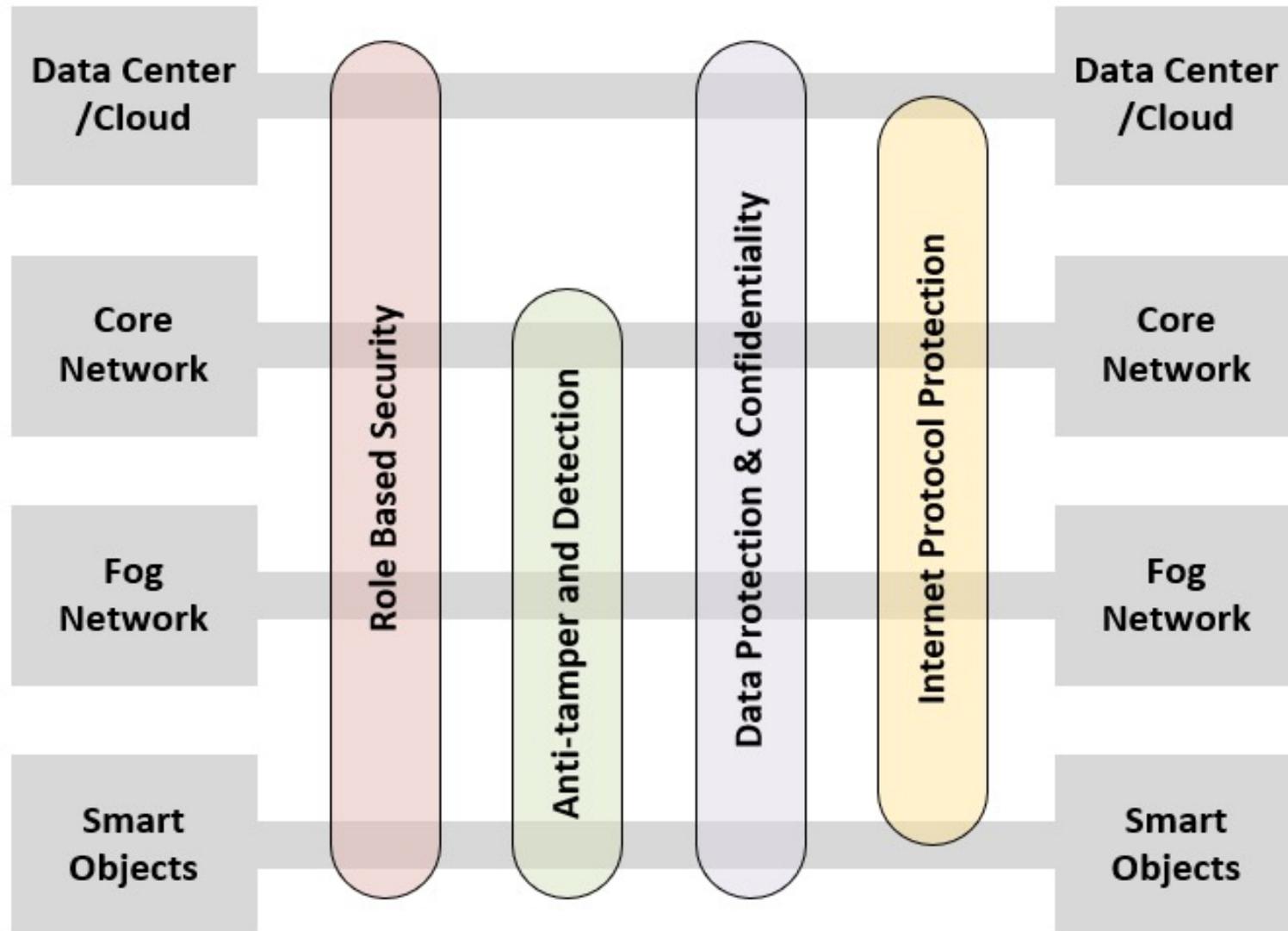
# IoT Security and Privacy Requirements (2/2)

- The requirements are:
  - Communication security
  - Data management security
  - Service provision security
  - Integration of security policies and techniques
  - Mutual authentication and authorization
  - Security audit

# IoT Gateway Security Functions



# IoT Security Environment



# Secure IoT Framework

- A secure IoT framework is also proposed by the Cisco model. The framework describes the components of a IoT security facility and encompasses all the IoT levels. The four components are presented below:
  - Authentication
  - Authorization
  - Network enforced policy
  - Secure analytics, including visibility and control



# Security considerations (1/3)

- As time passes, we are becoming increasingly dependent on smart, interconnected devices for a lot of tasks in our everyday lives. Nevertheless, the same devices or “things” can be the target of attacks and intrusions.
- Therefore, security is one of the main concerns regarding IoT, which needs to be addressed along with the paramount need for safety, since both matters are tightly intertwined with the physical world.
- Another important aspect involves administration of IoT devices, namely who is going to be responsible for this especially considering the inherent complexity and heterogeneity of the IoT ecosystem, as well as taking into account scalability concerns.

# Security considerations (2/3)

- There are a lot of different concerns that limit the consolidation of secure IoT ecosystems. Below, some of these concerns are presented:
  - Very large attack surface
  - Limited device resources
  - Complex ecosystem
  - Fragmentation of standards and regulations
  - Widespread deployment
  - Security integration
  - Safety aspects

# Security considerations (3/3)

- Low cost
- Lack of expertise
- Security updates
- Insecure programming
- Unclear liabilities

# Challenge of defining horizontal baseline security measures (1/3)

- High complexity of studying IoT security in a horizontal way, due to the security measures and the impact of the threats being determined by the criticality of the different assets, which differs depending on the use case, the application use and the use scenario.

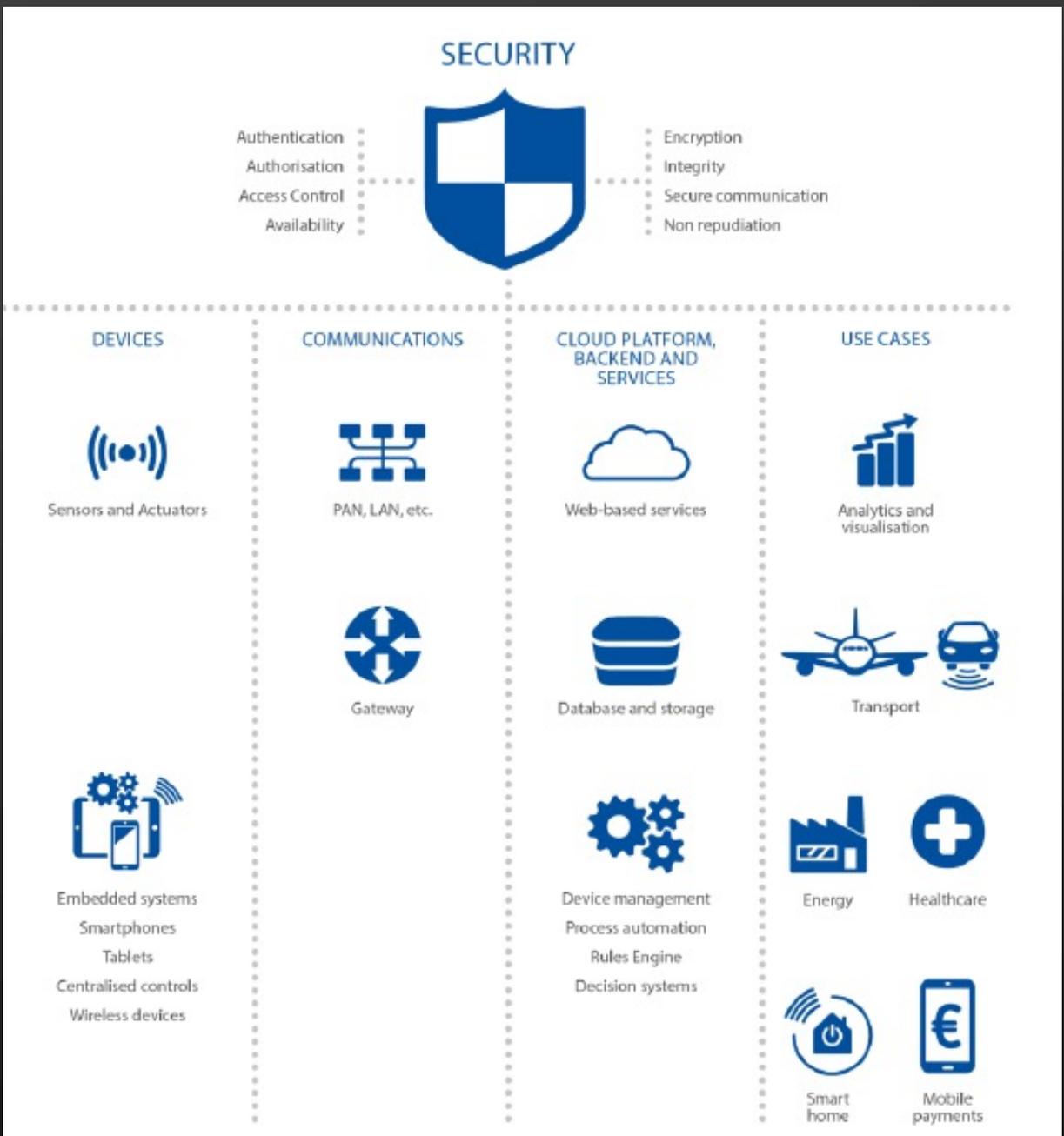
# Challenge of defining horizontal baseline security measures (2/3)

- Each IoT environment requires a risk assessment to be carried out in order to understand the threats of the various assets, define the plausible attack scenarios and associate them with the context of the specific IoT service which will indicate the critical hazards and the ones that can be tackled.
- Therefore, it is easy to perceive the intricacy of approaching the IoT in a horizontal way, in contrast to dealing with a specific IoT problem vertically such as Smart Cars, Smart Airports, Smart Homes, Intelligent Public Transport etc.

# Challenge of defining horizontal baseline security measures (3/3)

- Despite the above-mentioned, this section considers the horizontal aspects of IoT that are observed across vertical sectors and focuses on defining baseline security measures for IoT across Critical Information Infrastructures.
  - Therefore, this section compliments the actions of ENISA in the vertical sectors and shows a holistic approach towards IoT security.

# IoT High-level reference model



# Security measures and good practices (1/5)

- In this section, security measures and good practices are listed in detail.
  - These measures and practices aim to tackle the threats, weaknesses and risks associated with IoT devices and environments.
- They were defined while taking into account the various IoT environments and deployments that they could be applied. Therefore, a wide range of security matters, such as security by design, data protection, risk assessment and others, is covered.

# Security measures and good practices (2/5)

- The security measures and good practices can be divided into several security domains. The purpose of this split is the coverage of every IoT environment horizontally and the classification of the security measures based on the IoT ecosystem that they apply. Below, the proposed security domains are presented:

# Security measures and good practices (3/5)

- **Information System Security Governance & Risk Management:** Includes security measures regarding information system security risk analysis, policy, accreditation, indicators and audit, and human resource security.
- **Ecosystem Management:** Includes security measures regarding ecosystem mapping and ecosystem relations.
- **IT Security Architecture:** Includes security measures regarding systems configuration, asset management, system segregation, traffic filtering and cryptography.
- **IT Security Administration:** Includes security measures regarding administration accounts and administration information systems.
- **Identity and access management:** Includes security measures regarding authentication, identification and access rights.

# Security measures and good practices (4/5)

- **IT security maintenance:** Includes security measures regarding IT security maintenance procedures and remote access.
- **Physical and environmental security**
- **Detection:** Includes security measures regarding detection, logging, and log correlation and analysis.
- **Computer security incident management:** Includes security measures regarding information system security incident analysis and response, and incident report.
- **Continuity of Operations:** Includes security measures regarding business continuity management and disaster recovery management.
- **Crisis Management:** Includes security measures regarding crisis management organization and process.

# Security measures and good practices (5/5)

○ As it was stated earlier, the proposed security domains arrange the security measures according to the area of IoT ecosystem that they apply to. Except the area of application, the security measures can be classified based on their nature. The first category is policies that must be considered during the devices' development. The next category consists of organisational business measures and employees that should be adopted by the organisation. The last category refers to the technical measures to limit the potential risks that target the IoT devices and other elements of the IoT ecosystem. Thus, the identified IoT baseline security measures are placed into the three categories and are presented below:

- Policies (PS)
- Organisational, People and Process measures (OP)
- Technical Measures (TM)

# Policies

- The first category of security measures is the policies relating to information security and how to make it more concrete and robust. The policies need to contain well documented information and they should cover all of the organisation's activity.
- Additionally, it should be mentioned that concerning the security and privacy by design, the security measures need to take into account the unique traits and context that the IoT device or system is deployed in. For example, different specifications are considered when an IoT device is used at a home environment, in comparison to a critical infrastructure. When applying specific security measures, it is always worth keeping in mind that the cyber risk associated to IoT depends highly on context.

# Organisational, People and Process measures

- All businesses must have organisational criteria for information security.
- Their personnel practices need to promote good security, ensure the management of processes and safely operate the information in the organisation practices.
- Organisations should ensure that contractors and suppliers are responsible and accountable for the functions considered.
- In the event of an incident in the safety of the organisation, the organisation must be prepared (responsibilities, evaluation and response).

# Technical Measures

- Evidently, the security measures and good practices need to consider and cover the technical elements, in order to diminish the vulnerabilities of IoT.
- Below we provide an overview of the necessary technical measures to preserve and protect the security of information in IoT.
- Since these are horizontal measures across vertical sectors/CII, given the particularities of each vertical, more concrete measures can be introduced for each vertical/CII.
- Applying these technical measures should take into account the particularities of the IoT ecosystem such as scalability, namely given the huge number of involved devices certain measures might need to be carried out at the level of specialised architectural components, e.g., gateways.

# Gaps analysis

- In this section, the main gaps of cyber security in IoT are analysed. Before addressing cyber security in IoT, it is necessary to identify and define the gaps, namely how much the present state deviates from the desired state. This deviation can be used to determine solutions to close those gaps.
  - Gap 1: Fragmentation in existing security approaches and regulations
  - Gap 2: Lack of awareness and knowledge
  - Gap 3: Insecure design and/or development
  - Gap 4: Lack of interoperability across different IoT devices, platforms and frameworks
  - Gap 5: Lack of economic incentives
  - Gap 6: Lack of proper product lifecycle management

# High-level recommendations to improve IoT cybersecurity

ID	DESCRIPTION
1	Promote harmonization of IoT security initiatives and regulations
2	Raise awareness for the need for IoT cybersecurity
3	Define secure software/hardware development lifecycle guidelines for IoT
4	Achieve consensus for interoperability across the IoT ecosystem
5	Foster economic and administrative incentives for IoT security
6	Establishment of secure IoT product/service lifecycle management
7	Clarify liability among IoT stakeholders

# Section Outline

- Overview of IoT supply chain
- Good practices for security of IoT supply chain
  - *Actors*
  - *Processes*
  - *Technologies*
- General guidelines for securing IoT supply chain



[www.enisa.europa.eu](http://www.enisa.europa.eu)

## Section 3

Guidelines for Securing the Internet of Things:

Secure supply chain for IoT

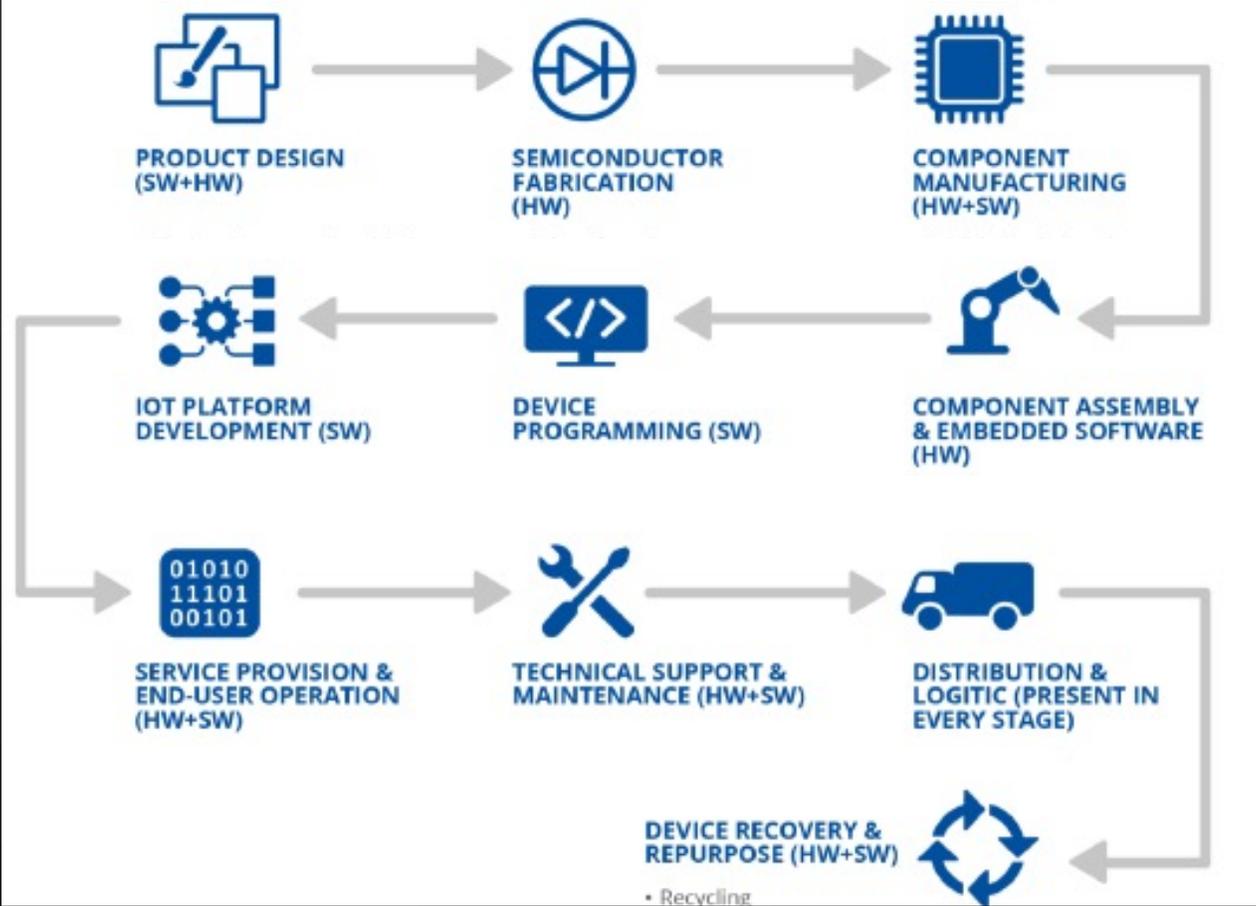
# Overview of the IoT Supply Chain

- The IoT supply chain includes the actors, processes and assets that participate in the realization (e.g., development, design, maintenance, patch management) of any IoT device.
- This study considers the supply chain for IoT is composed of two main aspects: the physical aspect and the logic aspect. The physical supply chain relates to all the physical objects (e.g. devices, electronic components, appliances) moved through the supply chain phases, as well as the associated manual processes (e.g., manual assembly, distribution processes).
- The logic aspect of the supply chain for IoT is associated with the software development and deployment, network-based communications, and virtual interactions between the IoT objects and the supply chain stakeholders.
- In this section, an overview of the IoT supply chain is provided. This aims to give an approximation of the stages sequence and the interactions between actors to identify where the security concerns might arise.

# Overview of IoT Supply Chain



# Mapping of the IoT supply Chain



# Good practices for security of IoT supply chain

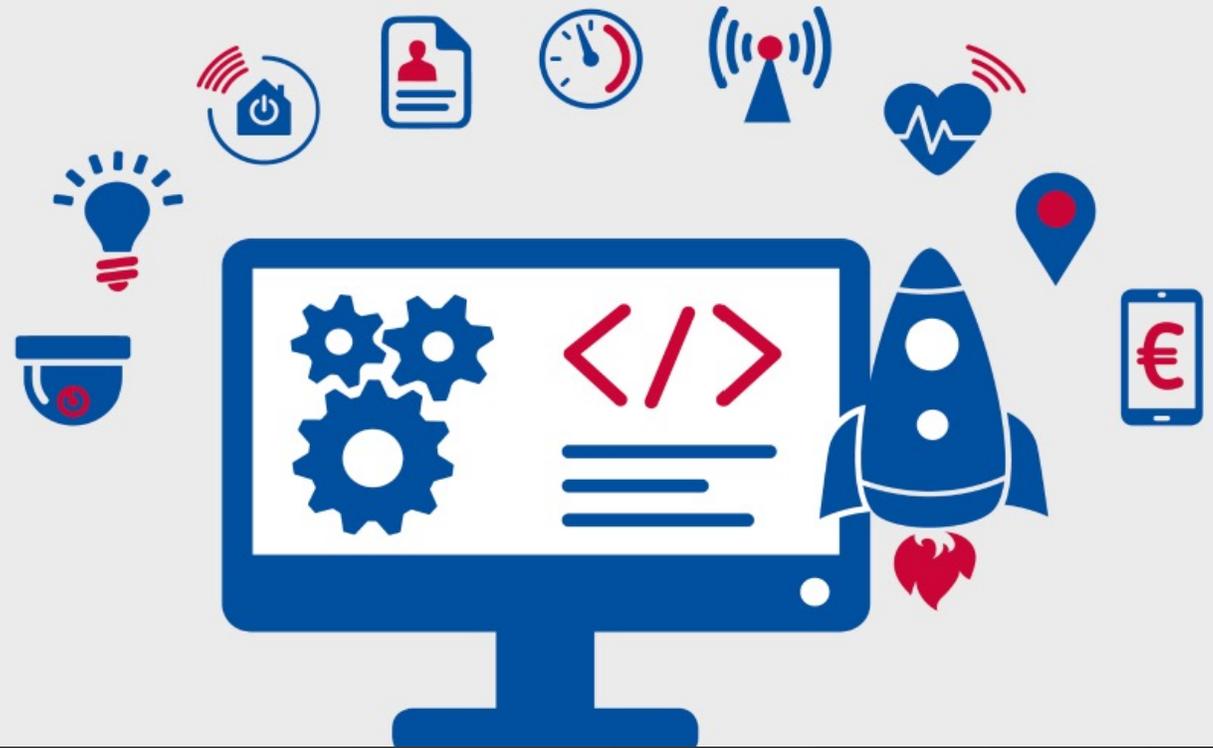
- Development of good practices for securing the supply chain for IoT is one of the key objectives of this study.
- The aim is to provide recommendations for the target audience to assist in countering and mitigating the threats that might impact the supply chain for IoT. Recommendations focus on covering the overlapping issues, as most practices are not effective across all industries and users.
- To organise the domains in a logical manner, good practices were classified into the following three main groups:
  - actors,
  - processes,
  - and technologies.

# General guidelines for securing IoT supply chain

- Forging better relationships between actors
- Cybersecurity expertise should be further cultivated
- Security by design
- Take a comprehensive and explicit approach to security
- Leverage existing standards and good practices

# Section Outline

- Introduction
- IoT Secure Software Development Lifecycle (SDLC)
- SDLC phases
- Security in SDLC



[www.enisa.europa.eu](http://www.enisa.europa.eu)

## Section 4

GOOD PRACTICES FOR SECURITY OF IOT

Secure Software Development Lifecycle

# Introduction

- ENISA establishes good practices regarding IoT security, focusing on software development guidelines to ensure the security of IoT products and services throughout their **lifetime**.
- Introducing secure development guidelines across the IoT ecosystem, is an absolute necessity for the improvement of IoT security.
- The good practices, provided in this section, can help to achieve **security by design**, a key recommendation that was highlighted in the Baseline Security Recommendations section which presented the **security of the IoT ecosystem from a horizontal point of view**.

# IoT Secure Software Development Lifecycle (SDLC)

- Making use of **secure Software Development Life Cycle (SDLC)** principles is an effective and proactive means to avoid vulnerabilities in IoT and thus assist in developing software applications and services in a secure manner.
- Several security challenges of the IoT can be addressed by establishing a baseline of secure development guidelines, such as checking for security vulnerabilities, secure deployment, ensuring continuity of secure development in cases of integrators, continuous delivery etc.

# IoT Secure Software Development Lifecycle (SDLC)

- ENISA strongly recommends **security and privacy by design and by default**.
- Accordingly, an effective and proactive means to reduce the number and severity of vulnerabilities in IoT is to develop applications in a secure manner, making use of **secure Software Development Life Cycle (sSDLC)** principles and developers trained in secure coding.
- Several security challenges of the IoT can be addressed by establishing a set of secure development guidelines, such as checking for security vulnerabilities, secure deployment, ensuring continuity of secure development in cases of integrators, continuous delivery etc.

# IoT Secure Software Development Lifecycle (SDLC)

- In this regard, the aim of this section is to define a **set of good practices and guidelines** to be applied in the different **phases of the secure SDLC of IoT solutions**.
- During this study, experts were asked what the main phases of the SDLC were. The vast majority of them considered that the SDLC comprises up to six phases, as shown in the following figure.

# SDLC Phases

1. *Requirements*
2. *Software design*
3. *Development/implementation*
4. *Testing and acceptance*
5. *Deployment and integration*
6. *Maintenance and disposal*



# Security in SDLC (1/3)

- When integrating **security** in a process (like the SDLC), one significant factor that is usually overlooked, refers to **assessment** and **evaluation**.
- The current cybersecurity state must be first understood before creating a plan to sustain this state and ameliorate it.
- In this respect, Security Maturity Models (SMM) are really useful because they guide organisations to recognise their level of security based on their intended requirements.
- There are numerous standards that serve as tools to evaluate the security of a software project. Some of the most widely recognised industry standards are:
  - Common Criteria (CC),
  - Capability Maturity Model Integration (CMMI),
  - Building Security in Maturity Model (BSIMM),
  - Security for industrial automation and control systems Part 4-1 (IEC 62443-4-1),
  - Open Software Assurance Maturity Model (OpenSAMM), among others.

# Security in SDLC (2/3)

- All six phases of the IoT SDLC share the same **core principle of security**.
- **Relevant and applicable controls** are required in each phase to assess the security state (e.g., creating security gates and metrics).
- **Security gates** must be implemented to confirm that software covers all required security conditions before moving forward to next phases.
- In each phase, the completion is indicated by **severity thresholds** that are defined by means of metrics. Utilizing the metrics, vulnerabilities throughout the development process can be analysed, detected and corrected.

# Security in SDLC (3/3)

- Complementary to security, the **documentation process** is a cross-cutting activity that is often viewed as not necessary during the SDLC process.
- This is due to multiple reasons, such as the complexity of the IoT solutions, the number of resources involved in a development process, the module integrations, the IoT interconnectivity, the number of external and internal components, designs, configurations, requirements etc.
- Nevertheless, a **good documentation** and a **documentation management system**, that supports the SDLC process, are crucial to consider it understandable, traceable, and subject to auditing and monitoring.



# Thank You

 Ms. Maria Papaioannou

 +351 932732968

 [m.papaioannou@av.it.pt](mailto:m.papaioannou@av.it.pt)

 <https://www.it.pt/ITSites/Index/3>

**PRODER**  
Introducing Recent Electrical Engineering  
Developments Into Undergraduate Curriculum



# This week's topics...

- General Principles
  - General **Perceptions** of Ethics
  - **Why Ethics?** *Ethics in general, related to IoT types of interactions, etc.*
  - A **methodical approach** to resolution: *checkpoints, resolution theories, etc.*
- Focusing on IoT Ethics
  - Developing IoT Technology: ethics for developers
  - Using IoT technology: ethics for users
  - **Law** and Ethics: *The differences and GDPR considerations*
- Tutorial: 'The Internet of Things and Ethics' Case Study
  - About the **Case Study**
  - Overview of IoT-based **monitoring and tracking applications**
  - Focusing on the **user**: Quality of Experience
  - Focusing on the **ethical aspects**: access, discrimination and inequality, informed consent, potential for malicious use, privacy, responsibility, transparency and trust

# Section Outline

This section introduces general ethical principles and relevant approaches and perceptions to consider in subsequent sections, specifically, for the IoT networks and applications.



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

## Section 1

General Principles

---

# General Perceptions

The perspectives <sup>1</sup>

- On the one hand it can be argued that IoT technology:
  - May eliminate jobs 
  - May be intrusive
  - May disconnect us from tradition
  - May encourage us to become lazy/unhealthy
- On the other hand: 
  - It is exciting and new
  - It represents progress and better approach to aspects of life
  - It can help in addressing world issues
  - It will provide more leisure time by freeing us from tedious tasks
  - It will create more interesting rewarding jobs to replace the ones becoming obsolete

<sup>1</sup> McEwen, A. (2014). *Designing the Internet of Things*, Chichester, England: Wiley, 338 pages.

# Why Ethics?

## Ethics in general ...

- In general the idea of **ethics** is often viewed as a set of **moral principles** that can affect humans' quality of life, i.e. what actions are *right* or *wrong* in specific situations.
  - Many definitions of Ethics have been attempted that refer to '**quality of character**', '**moral duty or principle**', or '**proper behaviour**'.
  - The following is from the *Stanford Encyclopedia of Philosophy*<sup>2</sup> on **Computer and Information Ethics**:
    - *This is a branch of Applied Ethics that studies ethical problems aggravated, transformed or created by computer technology.*
    - *Computer ethics developments include new conferences, research centres, journals, textbooks, web sites, and courses.*
    - *Additional sub-topics in computer ethics continually emerge as information technology grows. Recent new topics include online ethics, "agent" ethics (robots, softbots), the "open source movement", electronic government, ethics and nanotechnology, etc.*
- Ethics that we will be discussing in this lecture <sup>3</sup>:
  - **The right to Privacy**
  - **The principle of informed consent**
  - **Data security and Safety**
  - **Transparency and Trust**
  - **Information/Power Asymmetries**

<sup>2</sup> plato.stanford.edu

<sup>3</sup> Tzafestas, S. G. (2018). *Ethics and Law in the Internet of Things*, Smart Cities, 2018, 1, 98–120; doi:10.3390/smartcities1010006

# The right to Privacy

## Overview and example

- Public data about Internet users exists, e.g. through social media
  - Massive data storage potential makes this more complicated
  - You have the right to decide that your data is not visible to everyone
- As the Internet of Things is about *Things*, data collected from these *Things* may also become available online
  - As sensor data is so ubiquitous, it inevitably detects more than just the data that you have chosen to make public. <sup>4</sup>
- Example
  - Consider the electricity smart meter:
    - *The aggregate data collected from a group of consumers can be very useful for the company (cost benefits) or the environment; but it may still "leak" personal data*
      - Private information about a household could be inferred from such data
    - *Who owns such sensor data?*
      - The electricity company? The householder? The other house residents? The Internet Service Provider?

<sup>4</sup> McEwen, A. (2014). *Designing the Internet of Things*, Chichester, England: Wiley, 338 pages.

# The principle of informed consent

Overview and example form <sup>5</sup>

- Informed Consent is an ethical and legal requirement
  - this is applied to participants in a specific research study, or customers of a specific service, users of a specific product provided by a company, etc.
    - *e.g. permission to collect or manipulate personal data or sensitive data, permission to join a clinical trial, etc.*
  - An example of a consent form should:
    - *Inform the subject about: his or her rights, the purpose of the study, the procedures to be undergone, the potential risks and/or benefits of participation*
    - *Be designed carefully according to the requirements of the regional regulatory body*
    - *Inform the participants of their right to withdraw at any time without penalty*

# Data Security and Safety

## Overview and example

- Data Security in the Internet of Things, is about safeguarding connected devices and networks, while practicing data security principles in unit devices, e.g. encrypting data.
- The connectivity of the “Things” in the IoT, opens them up to a number of vulnerabilities.
  - Although the issues of security are extensively covered elsewhere in the course, it is important to highlight the ethical issues that arise when security is compromised when it comes to personal and private data and basic safety.
- The idea of safety is about recognising and mitigating the risks of using the technology. Such risks could include:
  - Device Discovery Challenges
  - Loss of privacy
  - Physical security of devices
  - Lack of user awareness for device usage
  - Untimely patching and updating
- For example, a common IoT device, e.g. a home appliance or a wearable device, can be used by an attacker to infiltrate a larger network, or to obtain sensitive personal information.

# Transparency and Trust

## Overview and example

- Ideally data generating users should always be made aware of their rights (by using informed consent).
  - How will the data be used?
  - Who has access to it?
- For trust and transparency, it is important to explain to users their right to grant or withdraw consent at any time, as well as to request that their data is deleted or can be accessed in its entirety.
- For example, transparency and trust are important in dealing with large amounts of IoT data generated as part of a service based on Artificial Intelligence (AI) algorithms, e.g. machine learning.
  - Trust for the actual AI service is crucial but it is based on the trust in the generated data itself (i.e. the IoT generated data).

# Power/Information Asymmetries

## Overview and Example

- Asymmetry of Information, and consequently of power between users and providers, is a phenomenon frequently observed in IoT deployment:
  - Oftentimes, it is necessary for the service provider to have more information than the service user (even about the user) in order to be able to use the IoT to support the service itself.
- For example, consider a service that uses location information of customers and vehicles in order to match appropriately the customers to vehicles (e.g. if the customers need a ride or a delivery).
  - In order to complete the transaction the provider must collect information about the user and match to the information about the vehicles, resulting in information asymmetry.
  - It is important to ensure that the asymmetry is solely allowed for the use within the service and that it is beneficial to the user.

# Considering specific IoT application types

- eHealth
  - E.g. wearables
- Transportation
  - E.g. congestion control
- Smart Metering/Banking/Farming
- Industrial Automation
- Remote Monitoring
  - Environmental monitoring
  - Tracking, e.g. employees, vehicles, etc.



# Why Ethics?

Ethics related to IoT types of interactions

- Interaction between IoT providers and IoT users
  - The principle of informed consent; trust
  - The terms of use are often not fully understood (note data collection and data sharing are often included in such terms)
- Other Human to Human interaction
  - E.g. tracking of employees, e.g. distribution/delivery company
  - Right to Privacy; Information/Power Asymmetries
- Human to Object interactions
  - Data Usability; data security; safety; quality of user experience.
  - *Can you think of application examples?*

# Ethical Checkpoints: a methodical approach

According to Rushworth M. Kidder <sup>6</sup>:

- Recognise that there exists a **moral issue** in a given scenario
  - What are the ethical issues that need attention?
  - Avoid too much (hyper-moralist) or too little (cynic) diligence on the issue
- The second step is to **determine the actor**
  - Recognise who is involved (are we all involved?)
- Gather the **facts**: what happened, what can happen ...
- Test for ***Right versus Wrong*** paradigms.
  - What does the law say? What would the public say? What would a parent say?
- Test for ***Right versus Right*** paradigms.
  - Is it a “truth versus loyalty”, “individual versus community”, “short-term versus long-term” or a “justice versus mercy” dilemma?
- Apply resolution **principles**:
  - Utilitarian, Kantian, or the “Golden Rule”?
  - Is there a third way out of the dilemma?
- Make the **decision, revisit** it, and **reflect** on it.

<sup>6</sup> Kidder, R. M. (2003). *How Good People Make Tough Choices*, HarperCollins, 241 pages

# Resolution Principles

## Utilitarian Principles

- *...the morally right action is the action that produces the most good, i.e. aim to bring about the greatest amount of good for the greatest number<sup>7</sup>*
- *Utilitarian principles hold that everyone's happiness counts the same (no egoism in the decision-making)*



<sup>7</sup> Stanford Encyclopedia of Philosophy

# Resolution Principles

## Kantian Principles

- Based on *Kant's Theory of Judgement*<sup>8</sup>
- Kant's ethics are based on a deontological theory of ethics and in summary state that an action is good only if based in a principle of duty to the moral law, i.e. that all people should follow regardless of their preferences.



# Resolution principles

## The Golden Rule

- This principle is also known through the phrase "*Do unto others as you would have them do unto you*"
- Normally we interpret the golden rule as telling us how to *act*. But in practice its greater role may be psychological, alerting us to everyday self-absorption, and the failure to consider our impacts on others. The rule reminds us also that we are peers to others who deserve comparable consideration.<sup>9</sup>



<sup>9</sup> Internet Encyclopedia of Philosophy, [iep.utm.edu](http://iep.utm.edu)



# Developing IoT technology

## Ethics for developers <sup>10</sup>

- High level requirements:
  - Human agency, liberty and dignity;
  - Technical robustness and safety;
  - Privacy and data governance;
  - Transparency;
  - Diversity, non-discrimination and fairness;
  - Individual, societal and environmental wellbeing;
  - Accountability.
- Responsible and ethical development must be based on:
  - Responsible and Ethical development methods, which we will go into more detail
  - A responsible and ethical policy/governance framework to encourage such development
  - Responsible and ethical societal structures, e.g. reflected in the education system, reflected in business practices

# Developing IoT technology

## Ethics for users <sup>10</sup>

- High level requirements (as previously for development):
  - Human agency, liberty and dignity;
  - Technical robustness and safety;
  - Privacy and data governance;
  - Transparency;
  - Diversity, non-discrimination and fairness;
  - Individual, societal and environmental wellbeing;
  - Accountability.
- Ethics must be integrated in governance and management, such that responsible deployment and use of AI is achieved:
  - Responsible IT Management and IT Governance, which we will go into more detail
  - Support form other stakeholders and society at large:
    - *IT suppliers, governmental institutions, educational institutions, professional organisations, clients*

# Operational Ethics Requirements

For technology developers/users <sup>10</sup>

## Human Agency, Liberty and Dignity

- Ensure that stakeholders are informed about how to control the system without being deceived
- Ensure that the system does not indirectly affect autonomy or freedom of stakeholders
- Ensure that the system does not interfere with the stakeholders' ability to make decisions

## Technical Robustness and Safety

- Ensure that the system is secure and resilient against attacks
- Ensure that the system is safe in case of failure (safe mode, fallback plan)
- Ensure the accuracy, reliability and reproducibility of the system (proper logging and documentation, audit-ready)

## Privacy and Data Governance

- Ensure protection of stakeholders' privacy
- Ensure protection of quality and integrity of data
- Ensure protection of access to the data
- Ensure protection of data rights and ownership

# Operational Ethics Requirements (cont.)

For technology developers/users <sup>10</sup>

## Transparency

- Ensure that the system has a sufficient level of traceability
- Ensure that the system has a sufficient level of explainability
- Ensure that the relevant functions of the system are communicated to stakeholders

## Diversity, Non-discrimination and fairness

- Ensure the avoidance and reduction of harmful bias
- Ensure fairness and avoidance of discrimination
- Ensure the inclusion and engagement of stakeholders

## Accountability

- Ensure that the system and its design process is auditable
- Ensure that negative impacts are minimised and reported
- Ensure internal and external governance frameworks
- Ensure human oversight

## Individual, Societal and Environmental Wellbeing

- Ensure a sustainable and environmentally friendly system
- Ensure the protection of democracy and democratic decision-making
- Ensure system evaluation for potential impact on individual well-being (vulnerable groups)
- Ensure the protection of social relationships/cohesion

# Law and Ethics

## The differences

### Law

---

- A set of rules produced by the government
  - The judicial system of the country to provide protection to the public
- Aim: maintain social order, peace and justice for society
- Compulsory for all citizens
  - What a citizen can do and cannot do
  - Penalties and legal consequences are defined for violations

### Ethics

---

- Ethics move beyond law
  - Based on people's awareness of right/wrong
- A system of moral principles
  - What is good or bad for individuals and for society
  - Agreed code of conduct
- Not compulsory
  - May be adopted by citizens
- Suggest how a person should live and interact with other people

# Law and Ethics

## The General Data Protection Regulation (GDPR)

### GDPR and potential ethical issues

- On 25th of May 2018, the European Union Regulation 2016/679 on data protection (the General Data Protection Regulation), came into effect
  - replacing the European legislation on data protection (Directive 95/46/EC)
- The example of monitoring under GDPR (enabled by IoT technology), may pose ethical concerns even though consent is provided.
  - It is clear that for any employee monitoring to take place (and hence for any personal data to be collected), **consent** must be given by the employee
- 
- Why is this type of consent of ethical concern?
  - Even though the seeking of consent between two parties demonstrates mutual **respect**, reinforces **autonomy** and generally assures **fairness**, often this is not the case, since there are at least three ways in which consensual transactions might be invalidated, and they include **fraud, exploitation and coercion** <sup>7</sup>.

<sup>7</sup> Macnish, K. (2018) *The Ethics of Surveillance: an introduction*. Routledge: London.

## Section Outline

This section will be a tutorial that makes use of a case study published in *The Orbit Journal* in 2019, and in particular in the special issue entitled *Case Studies of Ethics and Human Rights in Smart Information Systems*<sup>9</sup>, which visits the ethical implications of an IoT tracking application from a software designer's point of view, through interviews with the software development company.



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

## Section 3

Tutorial: 'The Internet of Things and Ethics' case study

<sup>9</sup> J. Antoniou, A. Andreou (2019). *Case Study: The Internet of Things and Ethics*, The Orbit Journal Special Issue – Case Studies of Ethics and Human Rights in Smart Information Systems, vol.2 no. 2, pp. 1 - 29

# About the Case Study

## The 'Internet of Things' and Ethics

- The ORBIT journal <sup>9</sup>
  - The ORBIT project is funded by the **UK Engineering and Physical Sciences Research Council**. Its purpose is to provide services to promote Responsible Research and Innovation (**RRI**) across the ICT research community.
    - *RRI aims to ensure the sustainability, acceptability and desirability of research processes and outputs.*
- The SHERPA project <sup>10</sup>
  - The Case Study took place within a European funded project (acronym: SHERPA), as one of several case studies that investigated the **ethical implications of developing and using emerging technologies**, like the IoT, AI, Big Data, etc.
  - SHERPA: Shaping the Ethical Dimensions of Smart Information Systems – a European perspective
- The Interviews
  - Two software designers of IoT-powered tracking applications
  - The company provides tracking software as a service nationally and internationally
  - The case study considers both the design and usage of such software in relation to ethics
    - *In addition to ethical implications, any relevant social, economic and legal implications are also identified*

<sup>9</sup> J. Antoniou, A. Andreou (2019). *Case Study: The Internet of Things and Ethics*, The Orbit Journal Special Issue – Case Studies of Ethics and Human Rights in Smart Information Systems, vol.2 no. 2, pp. 1 - 29

<sup>10</sup> [www.project-sherpa.eu](http://www.project-sherpa.eu)

# IoT-based monitoring and tracking applications

## Overview

- Purpose: to assist and enhance specific business processes
  - The principle of **informed consent**
  - The **terms of use** are often not fully understood (note: **data collection** and **data sharing** are often included in such terms)
- IoT and Big Data are used to create a Smart Information System
  - Data collection must be considered
  - Data manipulation to generate useful information must be considered
  - Decision making based on generated information must be considered

# IoT-based monitoring and tracking applications

## Issues Arising from Literature Review

### Effect on employee-employer relationship <sup>11</sup>

- The employee perspective: beliefs of full-time working adults according to an empirical study
  - Employees who do not perceive much privacy, tend to view their organization's policies as less fair, trust upper management less, and demonstrate less commitment to their organizations.

### Employer Motivation <sup>12</sup>

- prevention of related image damage,
- defense of corporate espionage,
- a general intended protection of corporate assets,
- detection of illegal software and missing data,
- increase of productivity, detection of reasons for a disciplinary warning,
- significantly reduced costs and increased availability of surveillance technologies

### Need for guidelines <sup>13</sup>

- Since monitoring is a practice which can result in a violation of rights, the need for guidelines and policies on monitoring
- Avoid discrimination claims by applying a policy consistently
  - employers must ensure that their employees understand the circumstances under which monitoring may take place

<sup>11</sup> Chory, R., et. al. (2016). Organizational Surveillance of Computer-Mediated Workplace Communication: Employee Privacy Concerns and Responses. *Employee Responsibilities and Rights*, 28(1), pp.23-43.

<sup>12</sup> Hugel, U. (2013). Workplace surveillance: examining current instruments, limitations and legal background issues. *Tourism & Management Studies*, 9(1), pp.58-63.

<sup>13</sup> Edwards, G. (2015). Employee Monitoring. What are the Legal Issues?. *Credit Management*, p.49.

## In Class Task:

- *What are the advantages and disadvantages of Business Use of IoT for Surveillance according to the Case Study?*
- Given we have discussed some of the issues arising from literature review:
  - *Identify some additional important issues that may arise, either from the case study or from your own investigation*



# Focusing on the technology user

## Quality of Experience (QoE)

- What is QoE? <sup>10</sup>
  - *QoE is the degree of **delight** or **annoyance** of the user of an application or service.*
  - *It results from the fulfillment of his/her **expectations** with respect to the **utility** and/or **enjoyment** of the application or service in the light of the user's personality and current state.*
- How can QoE be measured? According to the International Telecommunications Union,
  - Either by a subjective QoE assessment, typically based on Mean Opinion Score (MOS) of a service according to user perception,
  - Or by a objective QoE assessment, typically involving Quality of Service (QoS) parameters like latency, traffic volume density, reliability and cost etc.
- How is QoE related to Ethics?
  - Ethics refers to a preferred action or state based on aforementioned paradigms, and so does QoE
  - QoE is more closely related to a Utilitarian approach as it literally evaluates a utility function to quantify the user QoE (using either of the quantification methods mentioned above)

<sup>10</sup> Brunstrom, K., et. al. (2013). *Qualinet White Paper on Definition of Quality of Experience*, hal-00977812

# Subscription and Billing Management

Based on IoT

## Why?

---

- The software remotely tracks and monitors how assets are used:
  - So the company can bill according to usage
  - So that the company can identify usage fraud

## How?

---

- In two modes:
  - The service can host the software on a cloud server controlled by the developing company, and offer customers a software subscription as a service.
    - *The customer cannot completely control or have access to the software.*
    - *The developing company can monitor and maintain the software.*
  - the software can be offered as a stand-alone solution, where the software is installed at the customer's site.
    - *The customer has complete control over the management and use of the software, unless otherwise decided by the customer.*
    - *Even if the developing company supports the customer with maintenance tasks, they cannot control the software usage.*

# In-class Task

- Study the selected quotes.
- What are the ethical issues that can be identified in each case?
  - Consider the case study context for your answer
  - Do not simply echo the main issue given by the case study, but be inventive

*'have included specific clauses in [their] contract with [their] clients so that they know that we have access and if they have their own log enabled they will check where our users logged into and what they've seen.'*

*'not everyone has access to everything - it's up to the customers.'*

*'if the customer has not consented, it is not possible to allow the customer to use the system in a full functionality'*

*'if they abuse the system to target specific cases then we provide a full audit log that can be used to trace those cases.'*

*'we provide a full audit log of which users did what and when reporting of those actions.'*

# In-class Task

- Inspect the list of ethical issues.
- What are some real-life scenarios that could result in each of the issues?
  - Try to keep the same context as the case study
  - Can you offer possible resolutions?

PRIVACY

TRUST

DISCRIMINATION/INEQUALITY

TRANSPARENCY

CONSENT

ACCESS TO TECHNOLOGY

RESPONSIBILITY

## ○ What we learned

- What are ethics
- How to identify issues
- Why it is important to consider ethics for IoT
- How to incorporate them into technology
  - *Considering development and use perspectives*
- What is the difference between ethics and law
- IoT ethics currently
  - *Case Study*



[This Photo](#) by Unknown Author is licensed under [CC BY-NC](#)

## Summary

What we know now about IoT Ethics



# Thank You

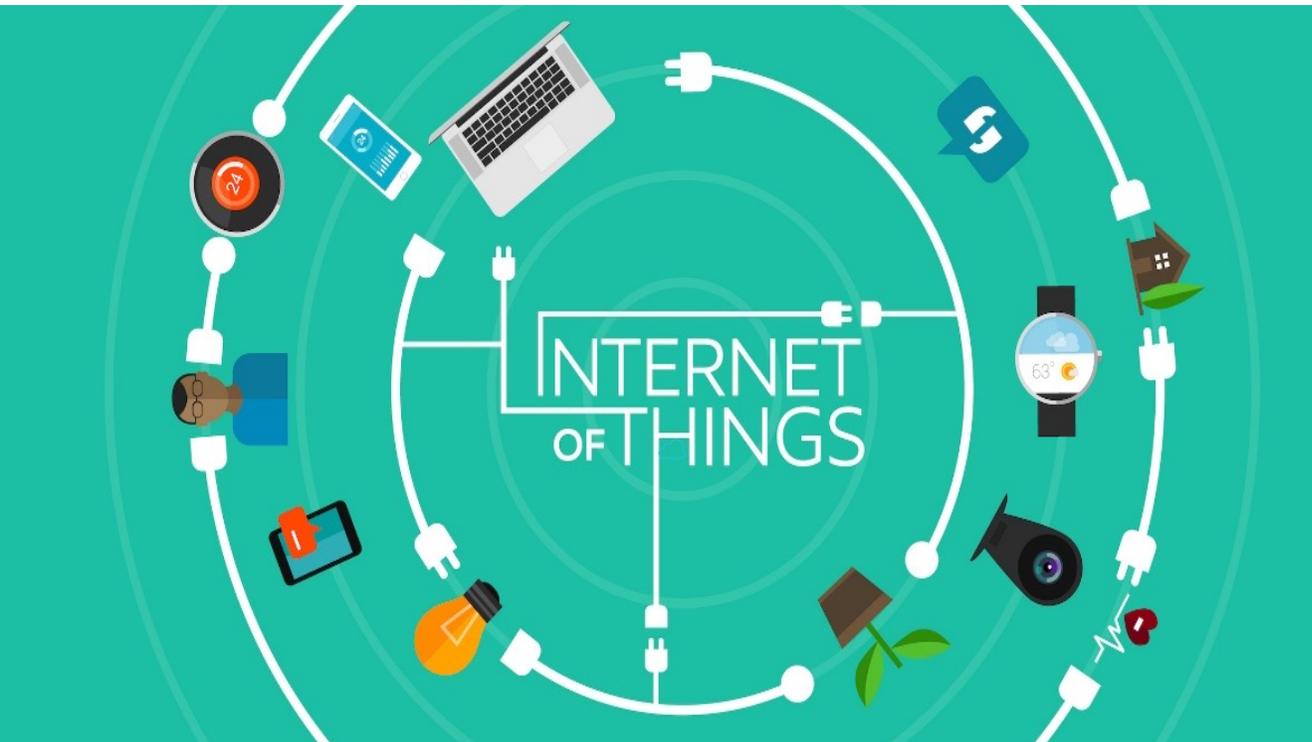
 Dr Josephina Antoniou

 +357 24 4000

 [jantoniou@uclan.ac.uk](mailto:jantoniou@uclan.ac.uk)

 <http://www.uclancyprus.ac.cy/>

**PROUDER**  
Introducing Recent Electrical Engineering  
Developments Into Undergraduate Curriculum



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)

*This publication was produced with the financial support of the European Union. Its contents are the sole responsibility of the partners of IREEDER project and do not necessarily reflect the views of the European Union*



Co-funded by the  
Erasmus+ Programme  
of the European Union

**Samiha Falahat**  
**Moath Alsafasfeh**  
**Saud Althunibat**

# Introduction to the Internet of Things

Lecture 13: Key-Enabling  
Technologies and  
Applications in IoT (1 of 3)

Introducing Recent Electrical Engineering  
Developments into undergraduate curriculum

IREEDER

# This week's topics...

## Key Enabling Technologies and Applications in IoT

- ❖ *Identification*
- ❖ *Localization*
- ❖ *Power management*

# Section Outline

## □ Introduction

## □ Identification

- Radio Frequency Identification
- Barcode Identification Technique
- Biometric identification

## □ Localization

- Positioning Systems
- Ranging Techniques
- Range-free Techniques

## □ IoT Power management

- Energy harvesting
- Battery technology in IoT



# Section 1

Key Enabling Technologies and Applications in IoT

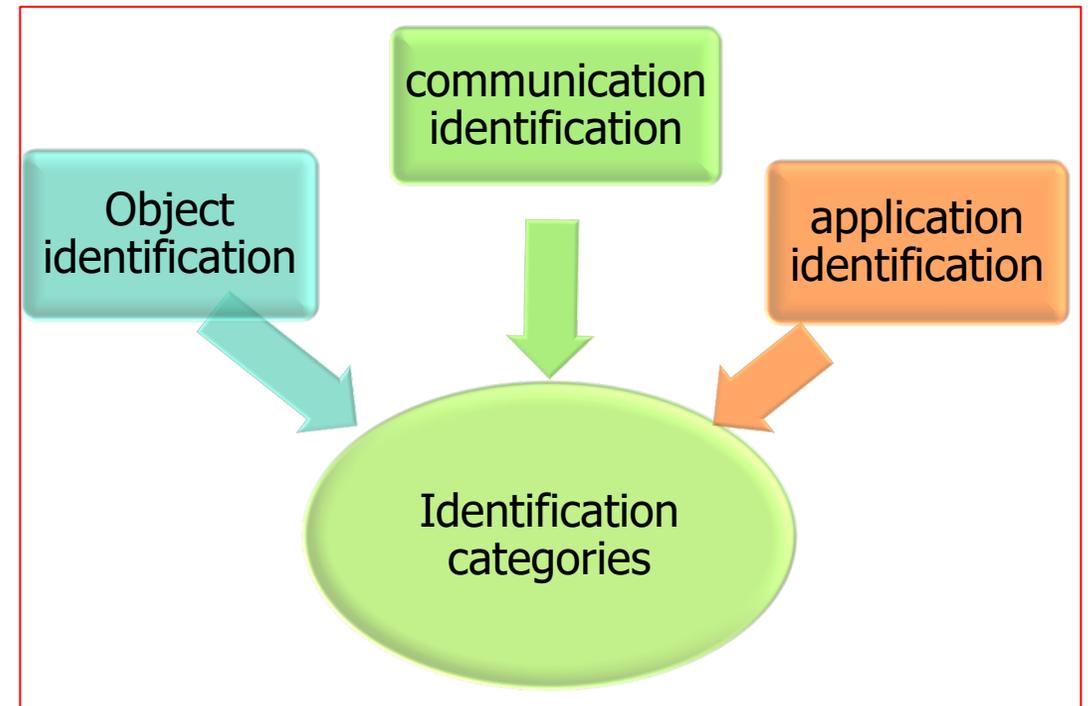
# Key Enabling Technologies in IoT

## Introduction

- Any IoT network requires a set of different technologies to enable/facilitate/accomplish the main task of the IoT networks.
- The enabling technologies for an IoT network vary depending on the scenario and the application.
- In this week, we address three of the key enabling technologies for IoT :
  - Identification
  - Localization
  - Powering Up IoT

# Identification

- When a node connects to the network, it is assigned an identifier that enables it to communicate with other nodes.
- Identification is important to control the excessive use of bandwidth in large networks (e.g., flooding) by ensuring that only identified nodes communicate.
- Identification can be divided into three categories (as shown).
- Our focus is on Object Identification



# Object Identification in IoT

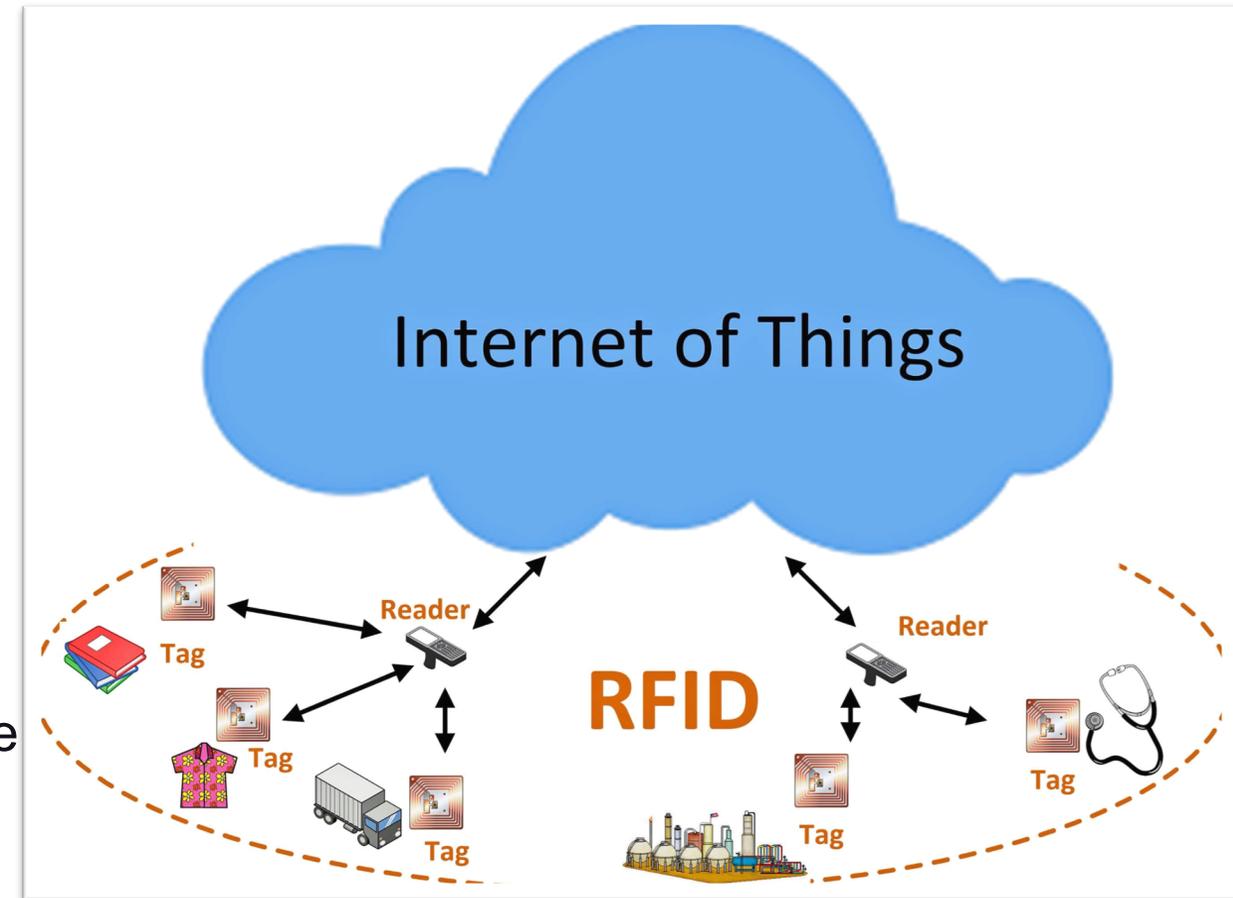
- To easily locate devices in the massive network, names and addresses are assigned to the devices.
- Object identification techniques provide an effective method of fast and accurate data collection.
- Solving slow speed and high error rate while taking the keyboard for manual input .

# Object Identification Techniques

- **Many object identification techniques exists:**
  - **Radio Frequency Identification (RFID):** Radio frequency identification systems use tiny, so-called *tags* with embedded microchips that typically contain a small amount of computer memory and transmit their content via radio signals over a short distance to specific RFID readers
  - **Biometric:** is a method for identifying and authenticating an individual using a collection of identifiable data that is unique and special to that person.
  - **Barcode:** the barcode is series of parallel printed lines (bars) these line may have variable width (mainly two widths available; thin and thick). The computer read bar in form of 0 or 1.

# RFID

- The specific role of RFID in the IoT can be observed from noting some fundamental components of the IoT, namely, sensing, communication, services, semantics, computation, and identification(Zhu et al., 2015).
- RFID technology can provide the link between the physical world and the virtual elements of the IoT.



Ref: *Qusay F. Hassan*, Internet of Things A to Z Technologies and Applications, 2018. IEEE Press Wiley

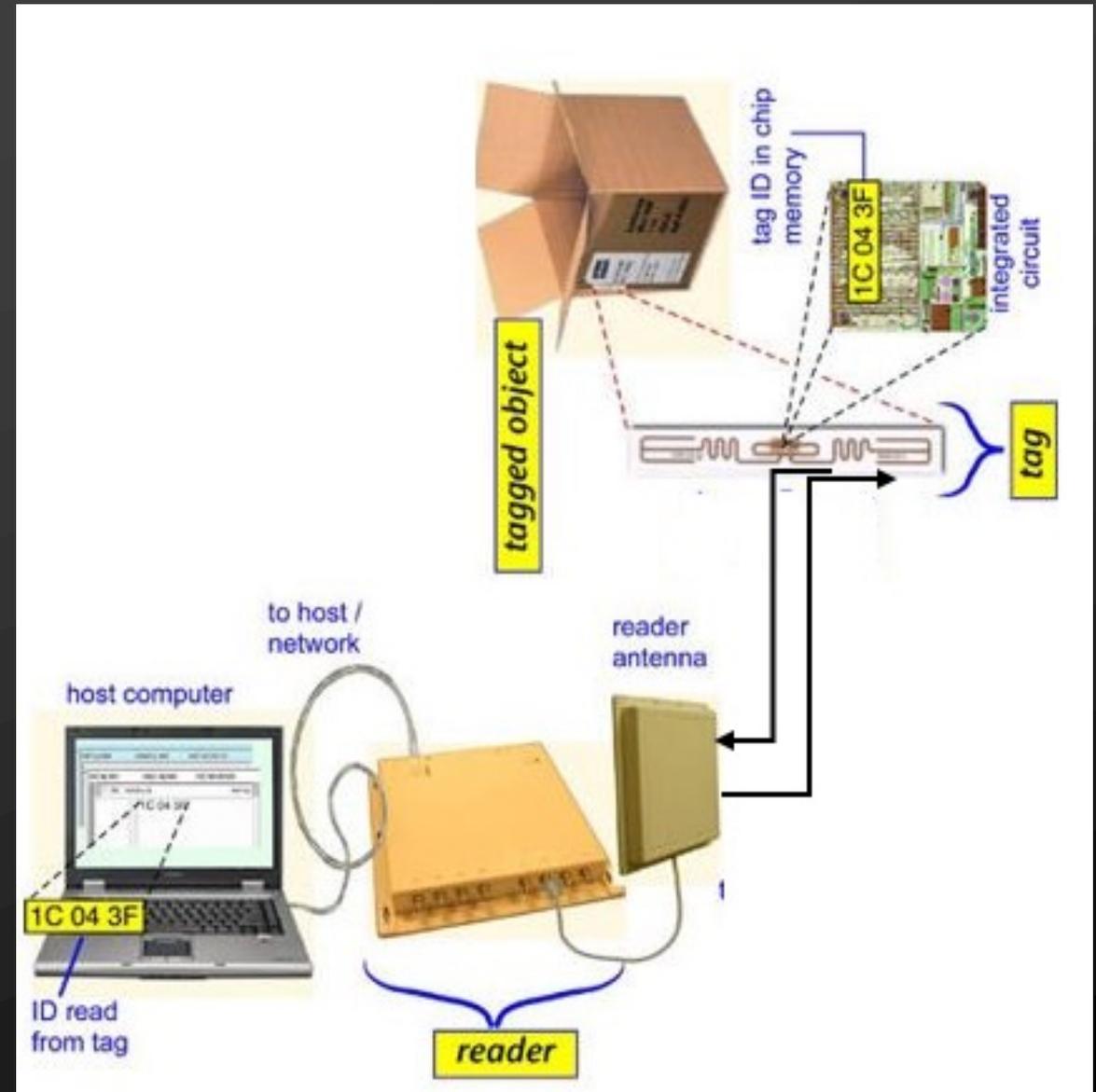
# Object Identification using RFID

- Readers and tags are the basic hardware used in RFID.
- Readers and tags are equipped with antennas for communication.
- A typical tag contains a chip that stores a unique identification number(ID)

# RFID System Components

- the main five components of A RFID system:

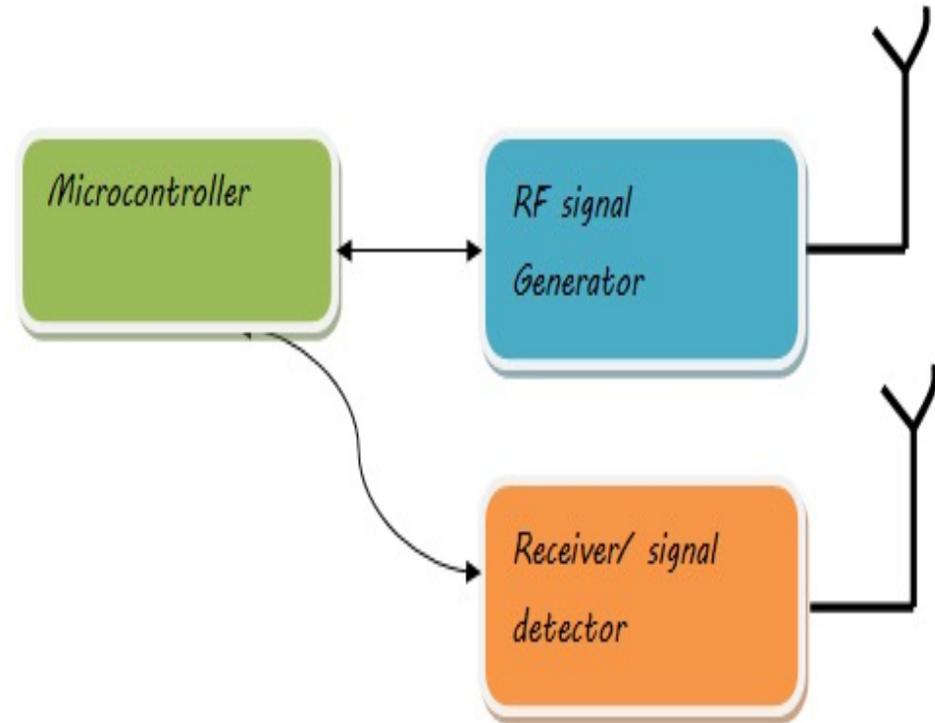
- Transmitter
- Receiver
- Microprocessor
- Antenna
- Tags



# RFID System Components

## Reader

- Reader is an intermediate component that connects the host server or the computer to the tag via the reader antenna.
- The RFID reader consist three main components (as shown)
- Radio signal generated and transmitted through the upper antenna, while the reader receiver detects the signal through the lower antenna.



# RFID System Components

## Tag

- It is an integrated circuit coupled with an antenna that may come with variant shapes and sizes.
- Tag is composed of a protective material that holds the pieces together and shields them from various environmental conditions.
- The tag contains four basic components:
  - Controller
  - Rectifier circuit
  - Transponder
  - Memory
- There are three classes of tags;
  - passive
  - active
  - semi-passive

# RFID System Components

## Antenna

- It aims at linking the reader with the tag by transfer and receive radio frequency signals between both sides.
- The working frequency of RFID system is has very wide

# RFID Operating Frequencies

- RFID systems use electromagnetic waves.
- There are several different frequencies an RFID system can use
  - Low frequencies or LF ( frequencies below 135 kHz)
  - Radio frequencies or HF (frequencies around 13,56 MHz)
  - Ultra-high frequencies or UHF (frequencies around 434 MHz, 869 - 915 MHz and 2,45 GHz)
  - Microwave or SHF (frequencies around 2,45 GHz).

# Energy transmission modes in RFID

- **Active:** which has combined power source (ex: battery), due to that, the operational range of active tags is usually much greater than that of passive tags. It can start communication to a reader or other active tags.
- **Semi-passive:** has an embedded internal battery (long reader rang with high cost) and unable to initiate communications to reader.
- **Passive:** has no embedded Internal power source, has less operational range than semi passive tags. it can initiate a communication.
- Active tags have the longest operating range and the highest cost.

# Data communications modes in RFID

## Inductive Coupling (for LF and HF bands):

- the reader produces a field that is used to match the RFID tag antenna
- Mutual coupling causes a voltage to be generated in the RFID tag's coil; A portion of this voltage is rectified and used to power the controller and memory modules
- In order to send data, when the power is sufficient, by connecting the load of the coil, the current will start to flow through this load, this current will change according to changes in the load impedance.
- Suppose the load is turned on and off. It seems also, The current will alternately turn off and on, causing a voltage to be produced in the RFID reader. The load is turned on and off. The load Turning on and off called load Modulation.

# Data communications modes in RFID

## Electromagnetic Coupling (for UHF band):

- the effect of coupling is electromagnetic When an RFID obstacle collides with an incident wave released by the reader, it is mirrored back to the reader.
- This principle known as backscattered coupling; Figure below explain the working principle

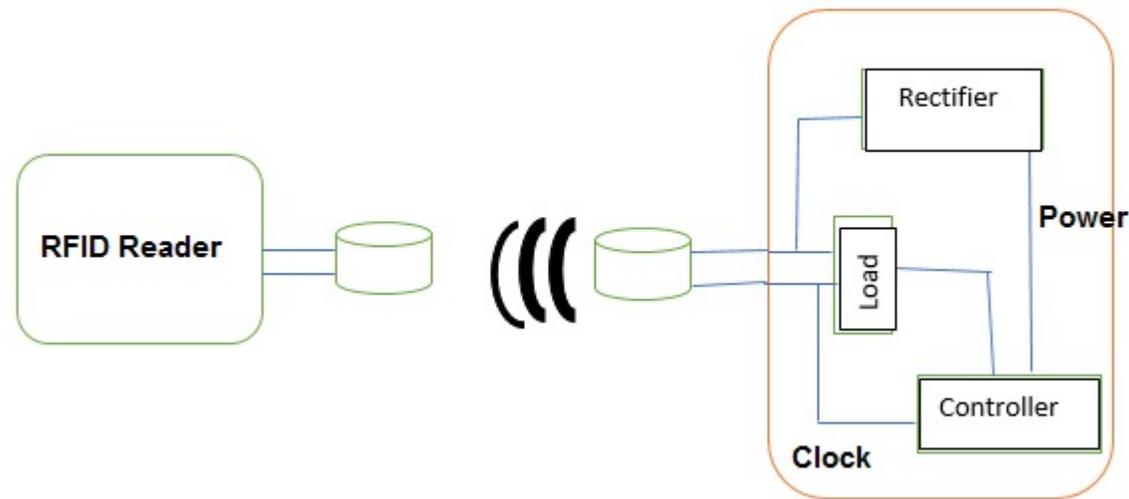
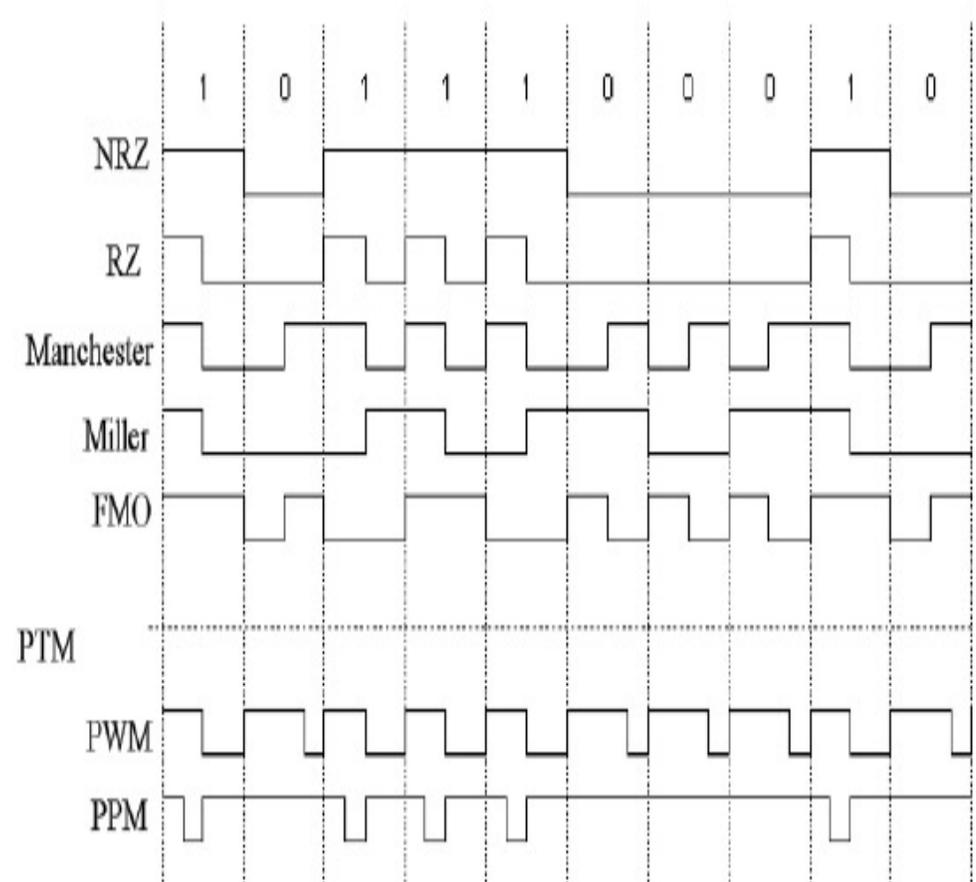


Figure 13-5 electromagnetic coupling (back scattered signal)

source: <https://www.youtube.com/watch?v=Ukfpq71BoMo>

# RFID Data Coding

- **Main three RFID coding types can be classified as follow:**
  - **Binary-voltage level association:** in this type a binary value corresponds to voltage level of the signal.
  - **Coding by signal transition :** a binary value is assigned corresponds to a variation between two voltage levels of the signal.
  - **Manchester coding:** coding by transition a rising edge in the middle of the symbol time corresponds to the binary value (0) and a falling edge corresponds to the binary value (1)
  - Other coding techniques also available like Miller coding, Bi-phase space coding(FM0), PWN, PPM, PIE.



# Applications of RFID

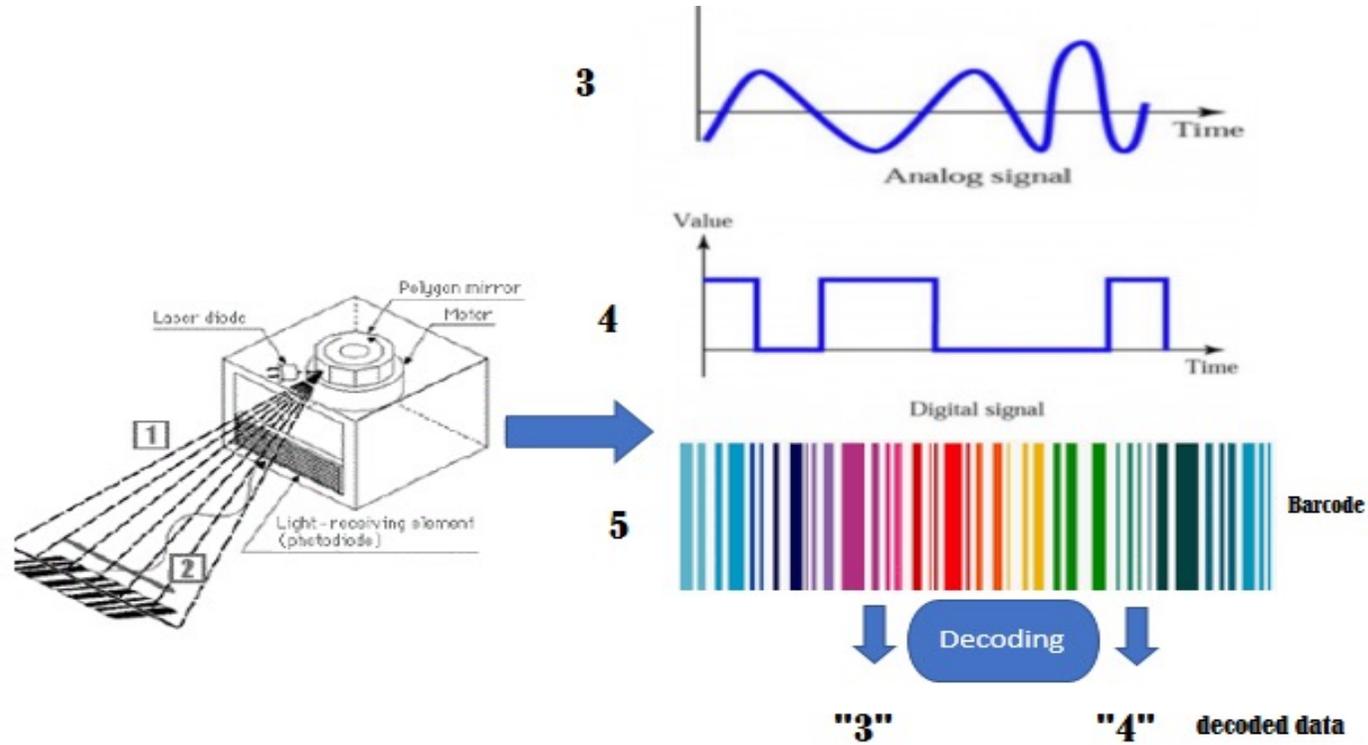
- **In-store traffic patterns**
  - Customer and employee flow
  - Product and packages flow monitoring
- **RFID in Libraries**
  - Lending and returning
  - Inventories are easily and quickly

# Barcode Identification Technique

## Introduction

- Barcode is a graphic visible representation of information.
- The information found in the bar code can pass from the scanner to the device where the code is detected when the bar code reader or scanner is used to scan the bar code.
- It has been widely used in many industries like post transmission and product tracking and management.

# Barcode working Principle



# Barcode Operation

**The main steps of Barcode working principle can be summarized in the following:**

- The laser beams emitted from the laser diode hit the polygon mirror and scan a bar code.
- The diffuse reflection light is obtained by the light-receiving portion (photodiode).
- As seen in the below figure, the diffuse reflection assembles an analog wave
- Analog to digital process done by the bar code reader.
- Digital signals are used to recognize narrow/wide bars and narrow/wide spaces.
- Data decoding done then the output is generated.

# Barcodes types

## One and two dimensions



One-dimensional Barcode



Two-dimensional Barcode

# Barcode Applications and Uses

- **Advertising:**

Advertisers exploit the barcodes capabilities by using them to reach out to customers in a more interactive way. Simply by downloading and installing an app that can read barcodes on a smartphone, you can learn a lot more about the product being advertised.

- **Tracking food intake:**

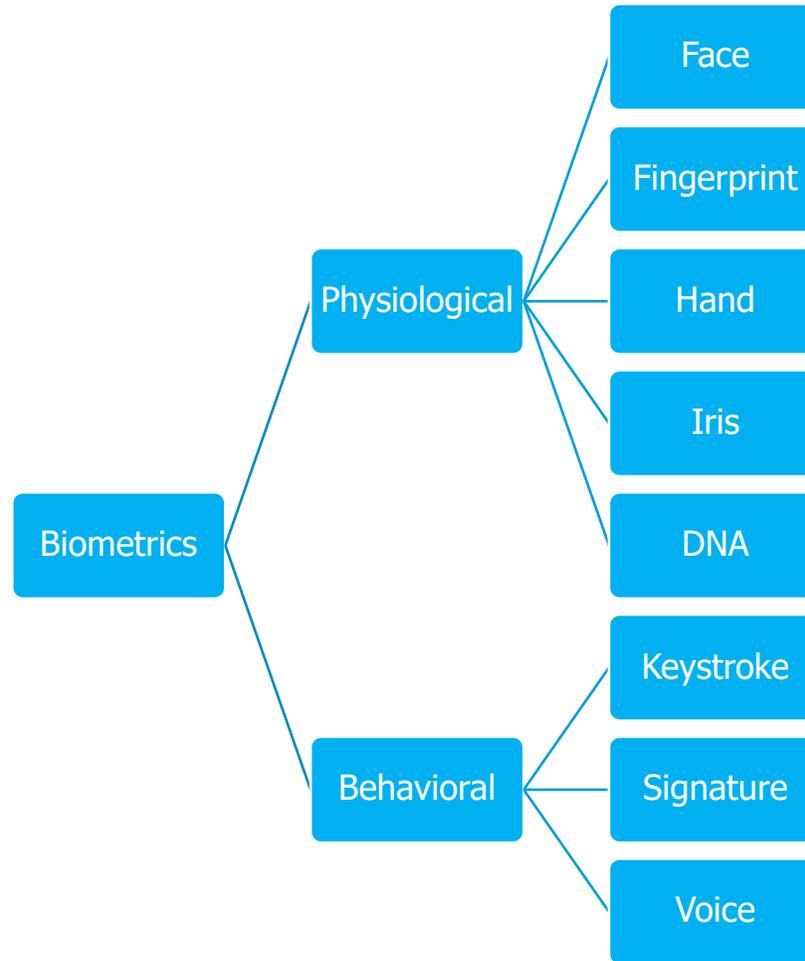
It's pretty good way to know detailed information about the food Ingredients, calories, product or expiry dates and so on. This is done by scanning the bar code on the food wrap using phone application then the user will be able to read all the food information.

# Biometric Identification

- Biometric is a technique that help in identify and authenticate a person based on set recognizable data.
- These data are unique and specific for person.
- **Biometric authentication:** is the method of determining similarities between data for a person's characteristics and data for a person's biometric "sample."
- **Biometric identification:**identity the person or objects

# Categories of Biometrics

## Physiological and Behavioral



# Examples of Biometric recognition techniques

## ○ **Face Recognition:**

- is based on the analysis of the features of the face and the relative position of the eyes, nose and mouth.
- high accuracy and low intrusiveness.

## ○ **Fingerprint Recognition:**

- fingerprints offer more secure and reliable personal identification than passwords, id-cards provide

# Comparison among identification techniques

- RFID is the most effective and reliable technique
- RFID has irreplaceable advantages in many aspects, such as transmission distance, reading and writing performance, anti-interference ability.
- The manufacturing cost of RFID is relative higher than barcode and visual code.
- In barcode there is only reading mode.
- Confidentiality in barcodes is a bad choice which has short lifetime.
- Barcodes are cheap.
- The biometric also works only in reading mode but it has good confidentiality, long lifetime, and high cost.

# Localization

## Introduction

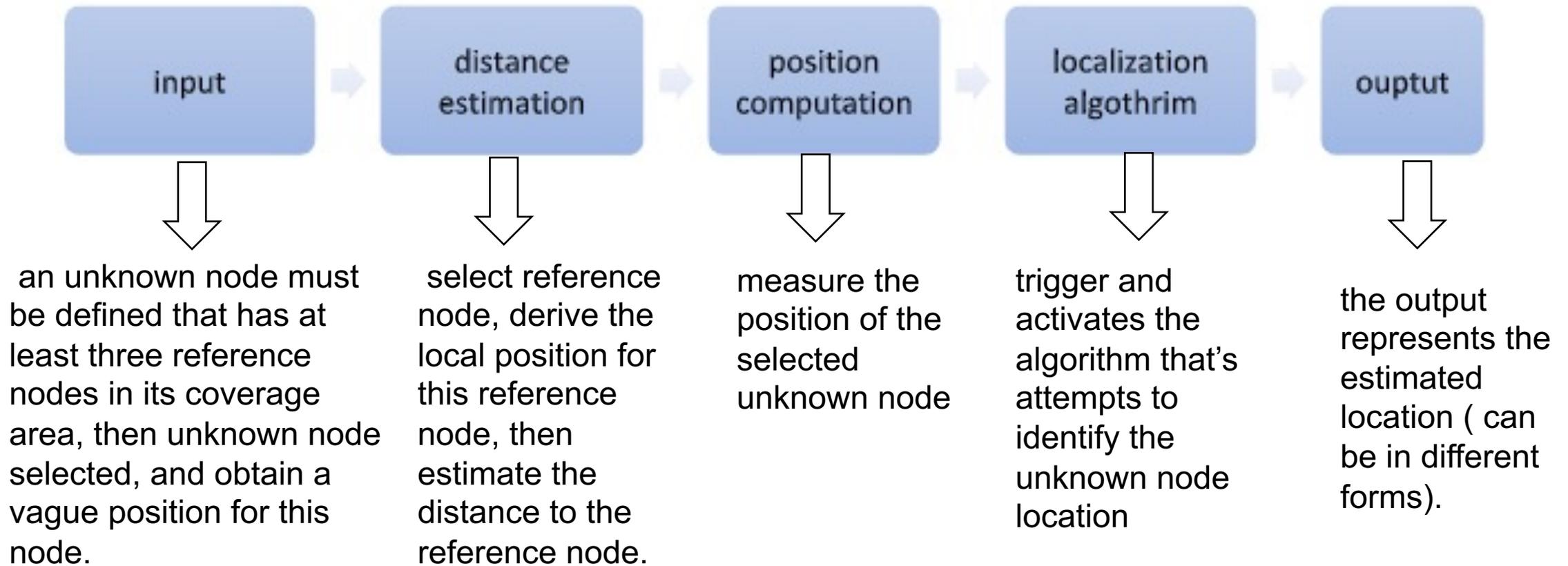
- Localization or positioning term refer to determining the location of an object (ex: Sensors in IoT Network) or human in specific area, or to determine the spatial relationships among different objects.
- Localization is important to provide a real physical context to sensor readings, for example, in environmental monitoring, location identification is necessary to sensor reading. Moreover, the Location information is important for services such as intrusion detection, and surveillance systems.
- The significant issues of the localization process it is how to monitor a moving object, as well as how to manage the location of an interior object.

# Methods of localization

- Many localization techniques depend on methods to specify the position of an object, these methods based on:
  - I. Geometric calculation such as triangulation
  - II. Trilateration
  - III. Scene analysis (called a “footprint”).

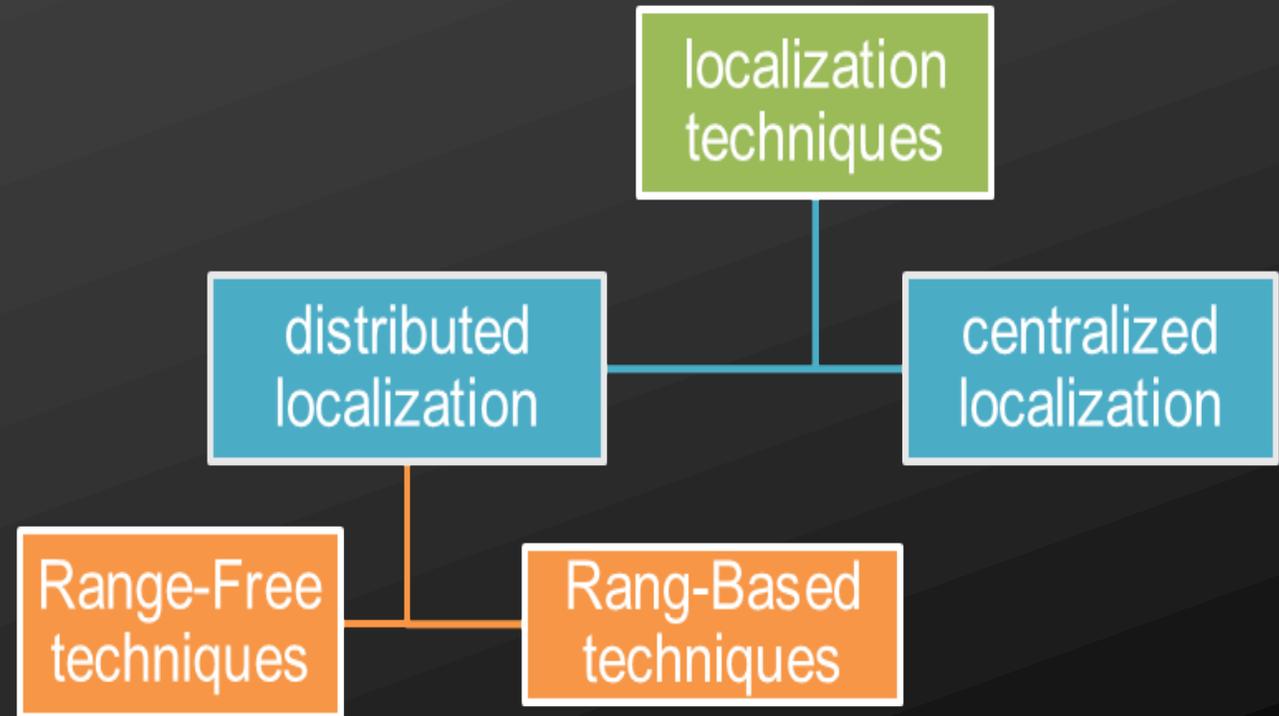
# Localization Process

- The process of localization can be summarized as shown below



# Localization techniques classification

- The classifications of the localization techniques is shown



# Positioning systems

## Indoor Positioning systems

- Indoor and outdoor environments create contradictory conditions for positioning and guidance of machines.
- Indoor environments are limited in size to rooms and buildings
- Ultra Wideband (UWB) ) is one of the most used radio technologies capable of providing sub meter positioning accuracy in indoor environments. UWB communication is strongly immune to multipath and noise. Since the UWB transmit power lies at the noise floor, the technology is also highly resistant to interference; due to very low transmit power.
- Wi - Fi and Bluetooth or RFID are common indoor positioning technologies that may provide accurate results.

# Positioning systems

## Outdoor positioning systems

- Outdoor positioning capabilities require regional or even global coverage.
- Only GNSS (Global Navigation satellites systems) are able to provide sub meter accuracy in outdoor location, cellular based systems are alternatives
- to increase the accuracy of measurements in outdoor localization the Differential GNSS is developed, it is derived from the principle of positioning error differentiation.
- The main disadvantage of DGNSS, which is the fact that the rover must be near the base stations in order for the corrections to take effect.
- Precise Point Positioning (PPP):method only needs a standalone receiver to operate. PPP relies heavily on post processing, with very precise satellite ephemeris, clock information and dual carrier receivers, the solution is able to deliver sub meter accuracy.
- The very long convergence and offset of moving rover are main drawback of PPP systems.

# Positioning systems

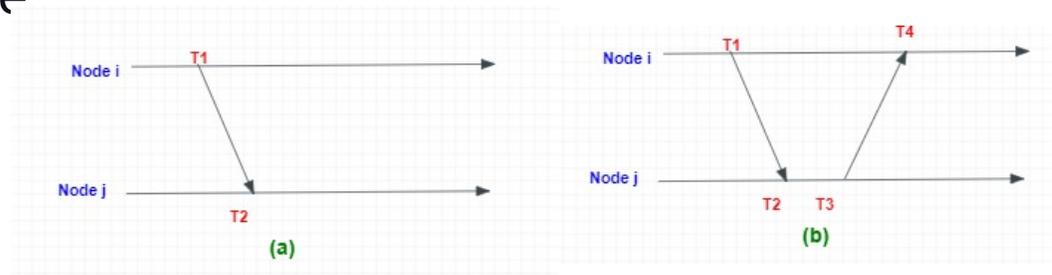
## Factors influence the selection of localization/positioning techniques

1. Accuracy
2. Responsiveness
3. Coverage
4. Scalability
5. Size and the cost of devices

# Ranging techniques

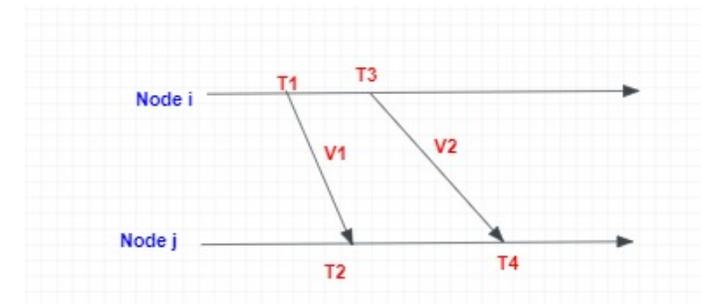
## Time of Arrival (ToA)

- determine the distance between the transmitter and receiver using measured propagation time of signal.
- Two main schemes: one-way time of arrival method measures the one-way propagation time. Two-way time of arrival the round-trip time of a signal is measured at the sender device



## Time Difference of Arrival (TDoA)

- In TDoA approach two different signals that travel with different speeds. The receiver is then able to determine its location.
- doesn't require synchronization clock between the sender and receiver.
- need for extra hardware implantation which increase the cost.



# Ranging techniques

## **Propagation Model**

---

- propagation model and RSS are methods used to calculate the distance between two nodes, when radio signal is detected.
- The received signal strength indicator (RSSI) feature can be used to measure the amplitude of received radio signal.

## **Angel of Arrival (AoA)**

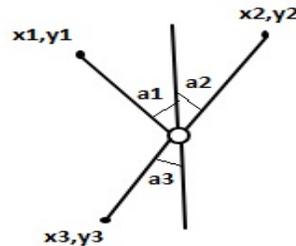
---

- direction of signal propagation
- typically achieved using an array of antennas or microphones
- angle between signal and some reference is orientation
- no need for clock synchronization
- a low accuracy due to the performance of the angular estimator, the accuracy can be high within few degrees.
- adds significant hardware cost

# Range-base Localization

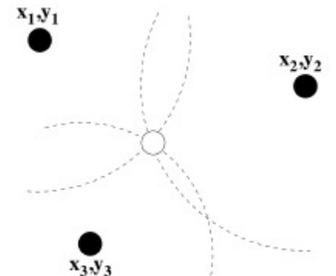
## Triangulation

- Triangulation uses geometric properties of triangle to estimate locations.
- relies on angle measurements.
- Minimum of two bearing lines (and the locations of anchor nodes or the distance between them) are needed for two-dimensional space"



## Trilateration

- Localization based on measured distances between a node and a number of anchor points with known locations"
- Basic concept: given the distance to an anchor, it is known that the node must be along the circumference of a circle centered at anchor and a radius equal to the node-anchor distance"
- In two-dimensional space, at least three non-collinear anchors are needed and in three-dimensional space, at least four non-coplanar anchors are needed



# Multilateration (MLAT)

**Multilateration (MLAT)** is a navigation technique that uses the difference in distance between two stations at known locations that transmit signals at known times to calculate the distance between them.

- There are two main multilateration method

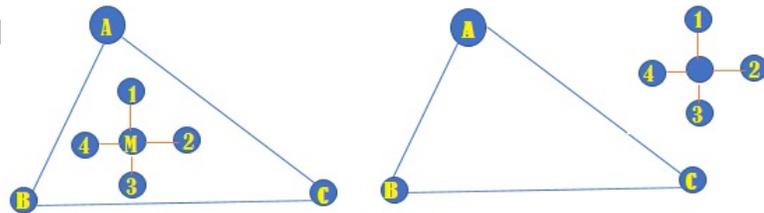
**The iterative multilateration** where the nodes in network exchange their estimated location with each other in repeated manner until they are totally localized.

**The collaborative multilateration**, in order to estimate the location of the node, it use the multi-hop information from neighbour node then attempt to calculate the possible location for both nodes.

# Range-Free Localization

## Approximate Point in Triangulation

- Any combination of three anchors forms a triangle
- Point in Triangulation (PIT) test used to determine the node presence; inside or outside a triangle.
- A node M is outside a triangle formed by anchors A, B, and C if there exists a direction such that a point adjacent to M is either further or closer to all points simultaneously; otherwise M is inside.



## Ad Hoc Positioning System (APS)

- One anchor node broadcasts a message which contains the anchors' positions with hop count.
- receiving node only keeps the minimum value, ignore the higher values.
- DV algorithm used to propagate locations.
- Each node has routing table which exchanged periodically with other nodes.
- Shortest distance computed by using following equation

$$c_i = \frac{\sum \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}}{\sum h_i}$$

# Range-Free Localization

## Fingerprint –based indoor localization techniques

### Visual fingerprint-based localization

- the devices can collect their position through image processing and matching procedure. .
- . the process is start after taking the photo by a camera which embedded in device, which been analysed by specific computing methodologies, next we need to use AI algorithms to gather the features and learn how to react. Then image processing techniques and algorithm activated to find the matching.
- Matching speed is major challenge

### Motion fingerprint-based localization

- detect the motion data of users or objects by using sensors.
- can find some moving patterns or trails of user to match with pre-defined motion signature.
- Drawbacks:
  1. Sensors may generate noises
  2. Accumulated errors

# Range-Free Localization

## Fingerprint –based indoor localization techniques

### Signal fingerprint-based localization:

- Two Phases
  1. offline training phase: form a fingerprint database which stores the correlation between Received signal strength (RSS) patterns from various access points (APs) and fix locations.
  2. fingerprint matching phase: use matching algorithm to search the fingerprint in database

### Event-Driven Localization

- makes use of localization event which are generated and propagate across the area of nodes.

# Comparison of localization techniques

techniques	cost	accuracy	Energy efficient	Hardware size
<b>Centralized based</b>	depend	high	less	depends
<b>Distributed based</b>	depend	low	high	depends
<b>Received Signal strength Indecator</b>	low	Medium	high	small
<b>Time of Arrival</b>	high	Medium	less	large
<b>Time deference of Arrival</b>	low	high	high	Large, but less complex
<b>Angel of Arrival</b>	high	low	Medium	large
<b>Distance vector hop</b>	low	Medium	high	small
<b>approximate point in triangle</b>	Medium	Medium	high	Medium

# Power Management in IoT

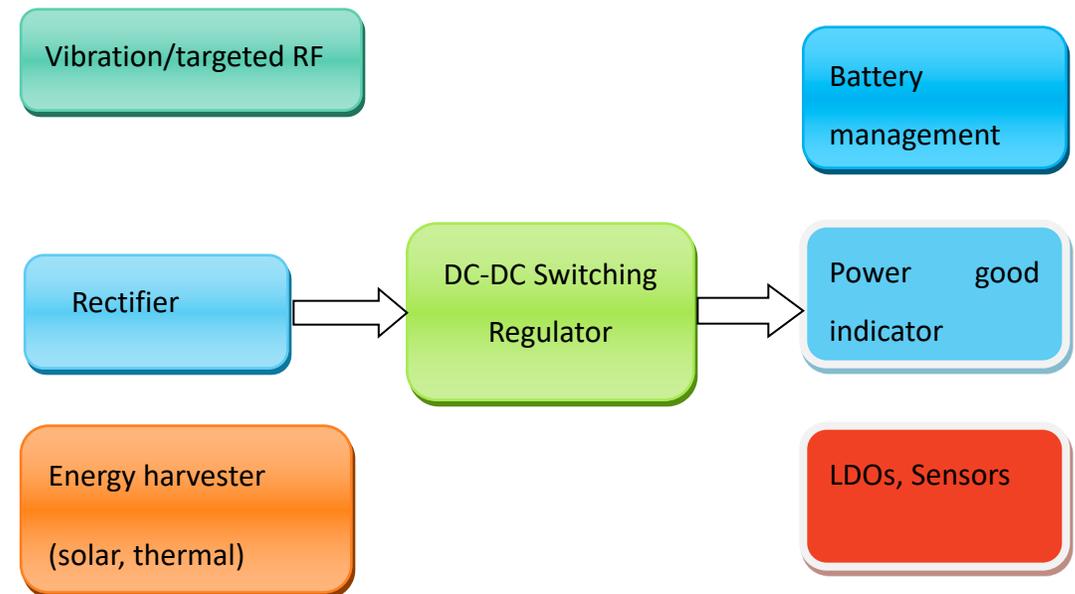
## Introduction

- Power consumption is a very critical issue in IoT.
  - IoT nodes are usually small, and hence, cannot be equipped with heavy/large power resources.
  - IoT nodes can be distributed in harsh environments which make battery charging or replacement is a challenge.
- As such, not every power supply is suitable for an IoT application.
- IoT power supply must:
  - Be highly efficient at both low load and full load
  - Be space-saving
  - Be reliable and affordable.

# IoT Power management

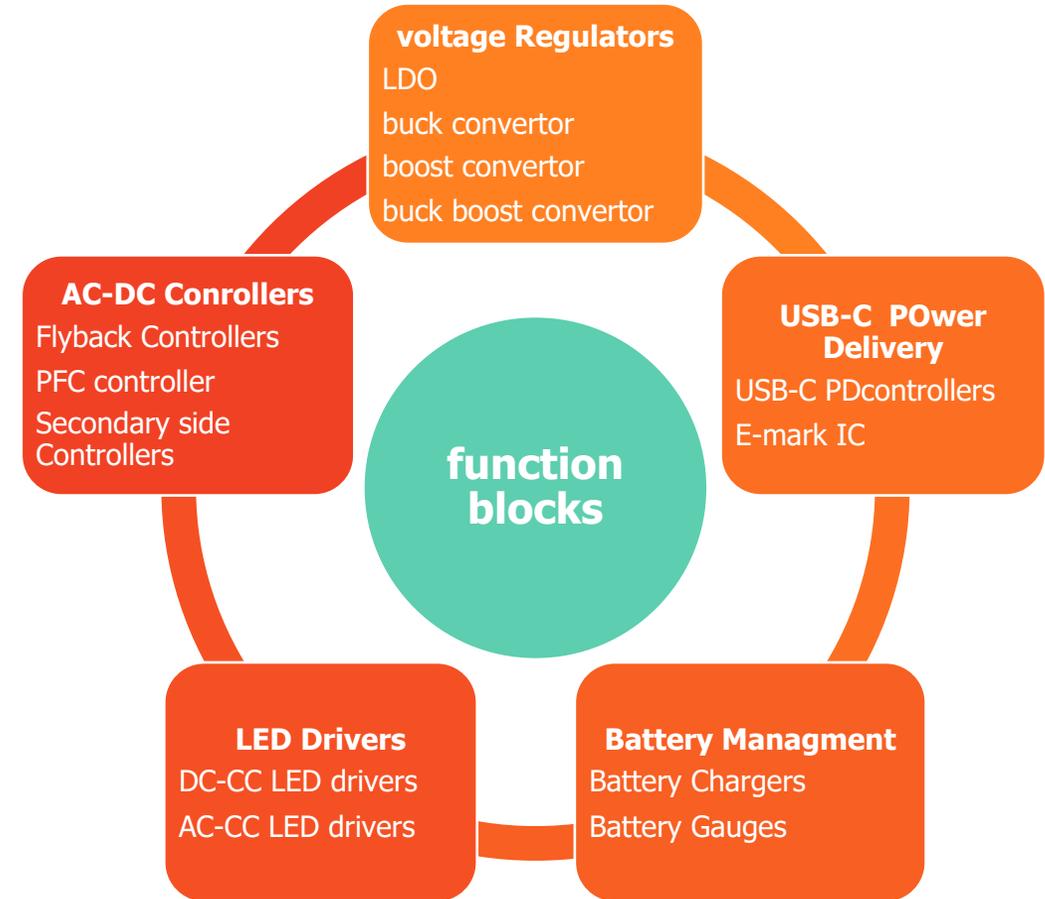
## Typical power management architecture of an IoT

- The rectified input is coupled with the switching regulator, which makes use of the Battery control block as well.
- The sensors output is transmitted to switching converter and further the output of DC-DC converter is sent to battery management.
- Some of the aspects of blocks concerning battery, LDOs, DC-DC converters need to be addressed for power management of whole system.



# Power Management Elements

- Voltage regulators: which take input voltage and regulate it into another application. For this purpose, we can use (linear regulator, buck convertor, boost convertor, or buck boost convertor). The regulator needs to step up or down the voltage between the battery and different sub-circuits in the device.
- AC-DC Controllers which take the AC output from main AC socket and regulate it into a stable voltage.
- LED Drivers which make from a DC voltage constant current.
- Battery Management block to control the power in batteries, by charging batteries or checking the battery state of charge.
- USB- C power delivery function block: it is a new technology used in power management



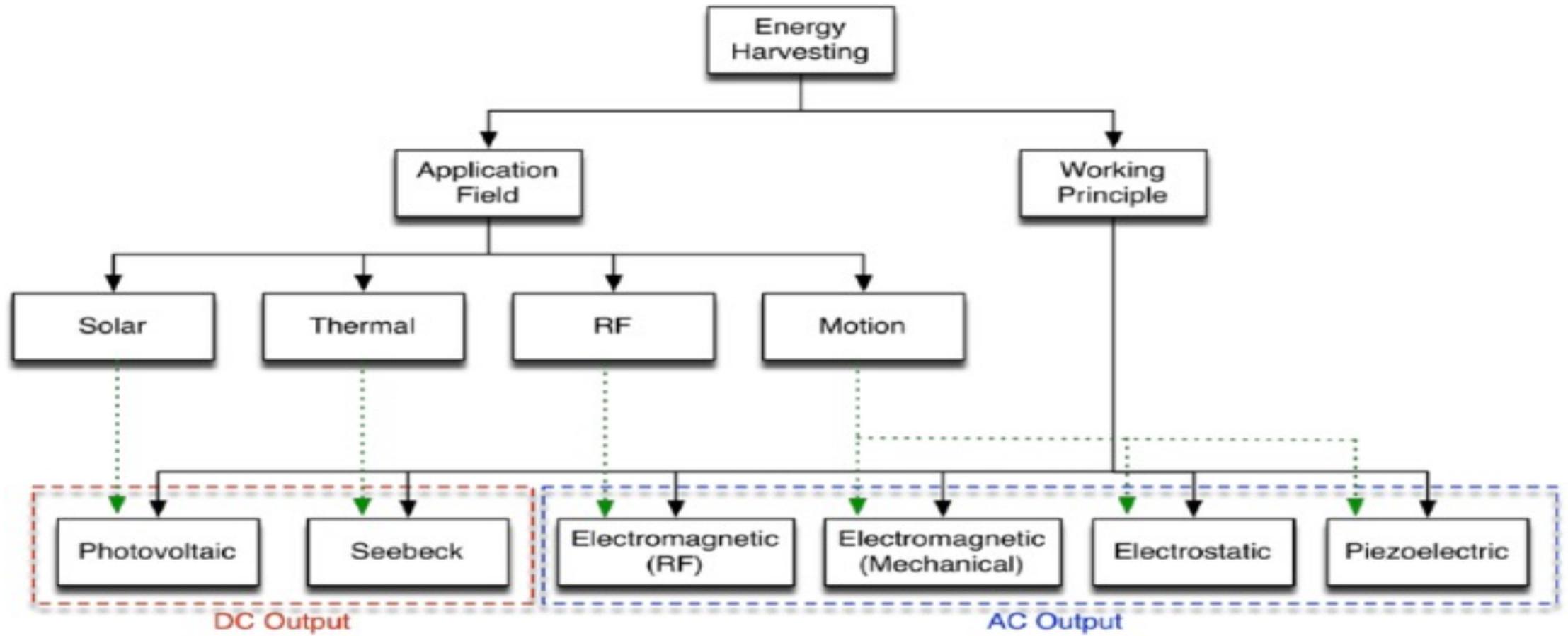
# Energy Harvesting

## Introduction

- Low-power electronics are powered by batteries.
- They have a limited existence and need to be replaced every few years.
- A technology that has been developed to counter this issue is “energy harvesting”.
- Energy harvesting is the method of extracting energy from natural sources and using it to power machines.
- energy harvesting can boost the system lifetime while lowering repairs; this can be done by exploiting some sustainable resources, such as solar, vibration, wind and thermal gradients.

# Energy Harvesting

## Hierarchy of energy harvesting technologies



# Energy Harvesting

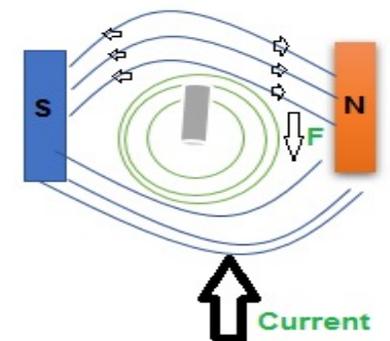
## Piezoelectric technology

- Energy is generated when mechanical stress or friction is applied on a piezoelectric material.
- When mechanical force or pressure is applied to such special materials, referred to as piezoelectric materials, the atomic structure of the crystal changes due to the net motions of positive and negative ions with respect to each other, resulting in electrical dipoles or polarization
- Thus, the crystal changes from a dielectric to a charged material.
- The amount of voltage generated is directly proportional to the amount of stress or tension applied to the crystal.

# Energy Harvesting

## Electromagnetic Energy Harvesting

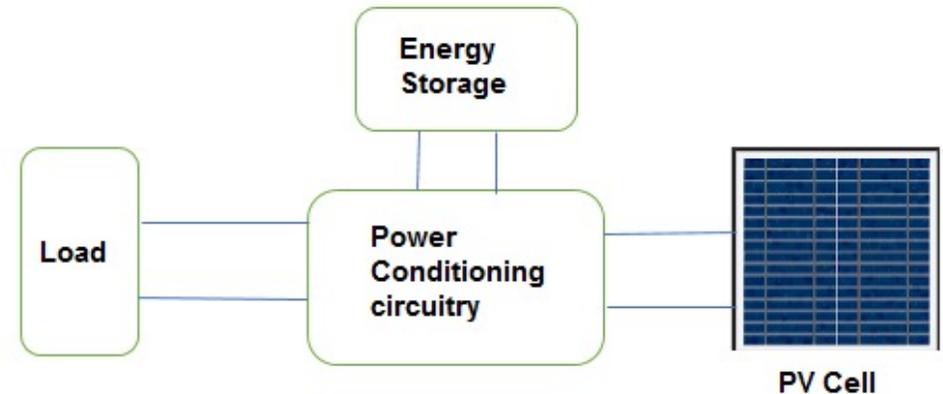
- The electromagnetic energy harvester was developed to transform a frequency range of 100-200 Hz input source (such as vibration) into usable electrical energy.
- This technology's basic idea is based on the magnetic field power that can be used to charge a battery in a fixed period of time.
- Faraday's law on electromagnetic induction (Faraday's law address that a current will be induced in a conductor when the magnetic field changed) is the core concept of electromagnetic energy harvester design



# Energy Harvesting

## Photovoltaic (Solar) Energy Harvesting

- thermoelectric generator converts heat directly into electricity according to a phenomenon called as the Seebeck effect.
- Thermoelectric harvesters benefits
- they are no maintenance needed
- due to the use of highly durable and lightweight solid-state equipment; they are silent and quiet
- heat is obtained from waste heating systems and transferred into electricity.so, they are highly effective in terms of the atmosphere



# Energy Harvesting

## Electrostatic Energy Harvesting

- The harvesting of electrical energy is achieved by fixing one of the capacitor plates and moving the other by an external mechanical motion to change one of the variable capacitor parameters (either plate area or plate separation).
- This allows mechanical movement to be converted into electrical energy according to the principle of electrostatics.



# Battery Technologies in IoT

- The physical size of the IoT node may limit the physical size of the battery
- The operating environment will direct whether the battery choices are restricted to those having modern, car, or business temperature appraisals.
- Components and functionality will decide the vitality and force necessities, just as whether an essential or battery-powered battery is more qualified for the application.
- Duty cycle of radio transmit/ receive pulses, communication protocols, sensor activity, and activation of visual indicators are some of the most power consumptive factors influencing the battery capacity needed in order to operate for an acceptable period between replacement or recharge.
- Batteries can be categorized as either non-rechargeable or rechargeable cell, referred as primary and secondary cells, respectively. the batteries can be made from wide variant chemical compounds.
- The essential chemistry of primary batteries is called (Alkaline), examples of primary batteries are (AA,AAA,C, D-Cell).
- Secondary batteries like Li-ion cylindrical and Li-polymer pouch cells are made from (lithium)

# Battery Technologies in IoT

## Factors influence batteries selection for IoT Application

- The nominal and cut-off voltage of an IoT application:
  - we should pick the one that will ensure the device be above the cut-off voltage throughout its lifetime.
- The environment's temperature:
  - Think about where the IoT device would be implemented to ensure an optimum and consistent supply of power to the object
- The consumption's profile and the maximum pulse current and frequency
  - determine if the IoT platform requires a low or high pulse frequency.



# Thank You

-  Dr Name
-  Telephone
-  email
-  website

**PROUDER**  
Introducing Recent Electrical Engineering  
Developments Into Undergraduate Curriculum