MODULE 1

Introduction to IoT

By
Dr. M. Jagadeesh Chandra Prasad
Professor & Head , ECE Dept.

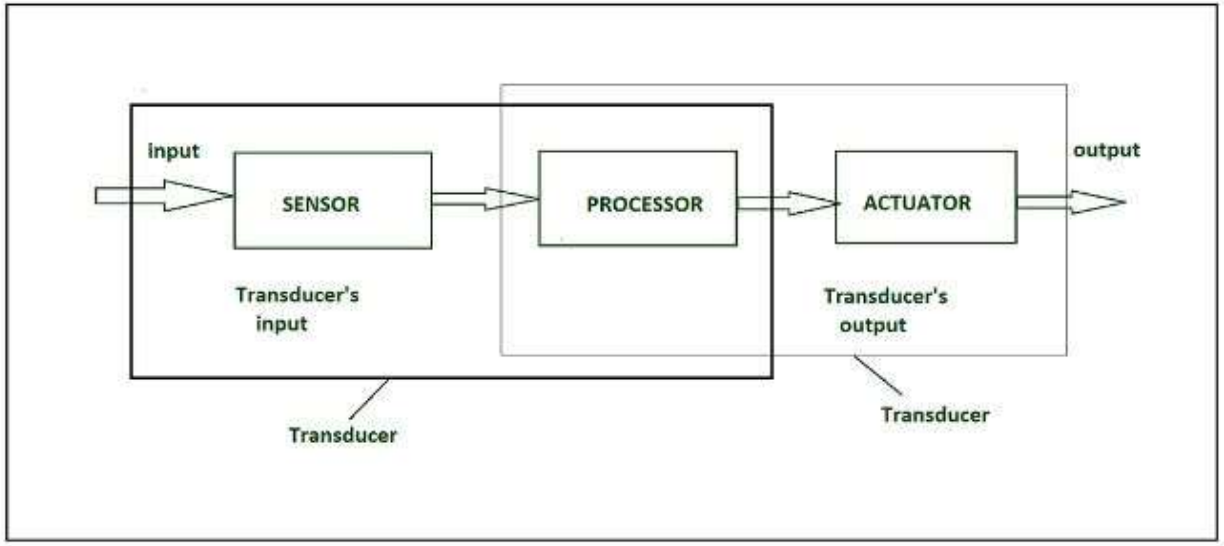
Outline

- Sensors and actuators
- Wireless sensor networks
- Machine-to-Machine communications
- IoT definition
- Characteristics of IoT
- Physical Design of IoT
- Logical Design of IoT
- IoT Protocols
- IoT Enabling Technologies
- IoT Levels & Deployment Templates

Sensors and actuators

Sensor: A device that can sense its environment, and translate physical quantities such as light, sound, temperature, and motion into electrical signals.

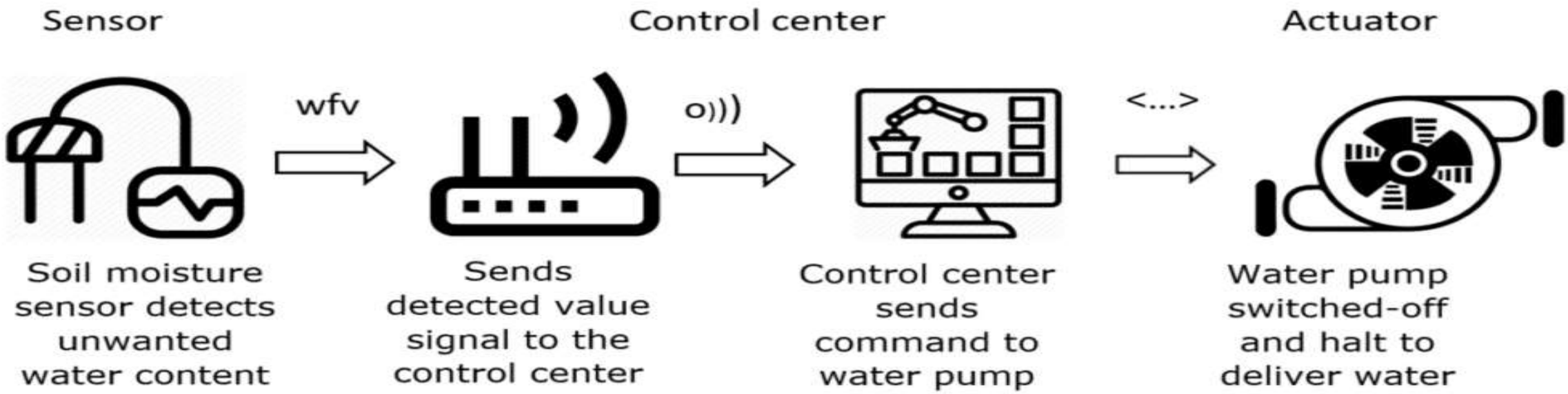
Actuator: A device that converts electrical energy into a physical quantity



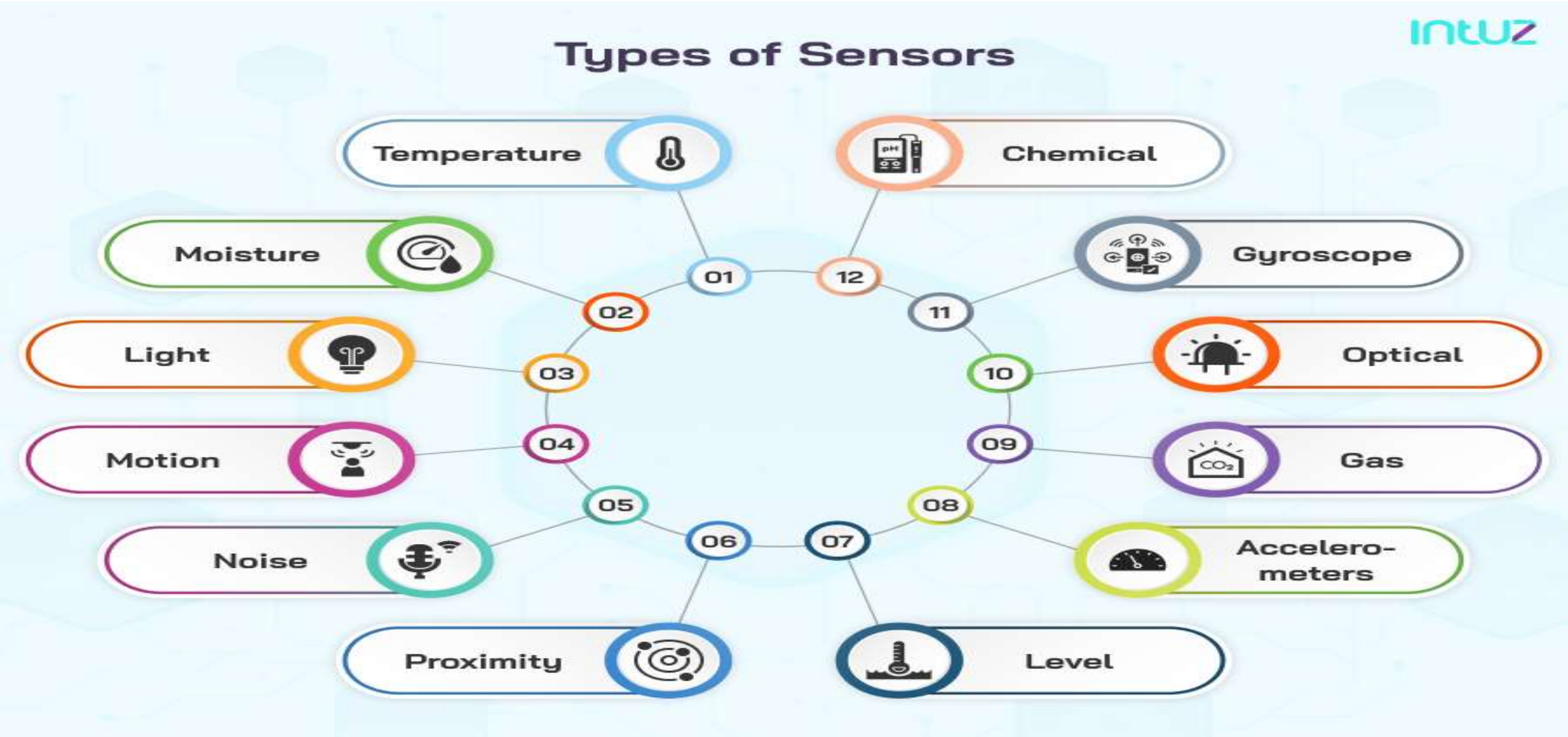
Difference b/w Sensor & Actuator

Sensor	Actuator
Sensor converts physical quantities and characteristics into electrical signals	Actuator converts electrical signals into physical action such as force and motion
It acts as an input device in any control system and placed in input port	It acts as an output device in a control system and placed in output port
Sensor takes input from environment and senses surroundings condition	Actuator takes input from output signal conditioning unit of system
It gives information to the system about environment condition to monitor and control	It accepts command from system to deliver physical action
Sensors are often used to measure process pressure, temperature, fluid levels, flow, vibration, speed etc.	Actuators are often used to operate control valves, dampers, guide vanes, and to move objects from one place to another, to move conveyor belts in robotic arms movement etc..

Example:


Sensor
Control center
Actuator
wfv
o)))
<...>
Soil moisture sensor detects unwanted water content
Sends detected value signal to the control center
Control center sends command to water pump
Water pump switched-off and halt to deliver water

Sensors in IoT



Different Types of Sensors

Thermistor
(Temperature Sensor)

IR Sensor
(Transmissive Type)

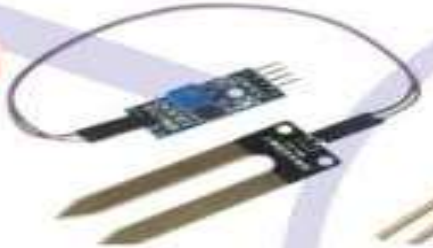
IR Sensor
(Reflective Type)

Ultrasonic Sensor

Gyroscope
Sensor

Accelerometer
Sensor

Rain Sensor



Soil Moisture Sensor

Phototransistor
(Light Sensor)

Water Flow Sensor

Heartbeat Sensor

Alcohol Sensor

Color Sensor

PIR Sensor

Gas Sensor

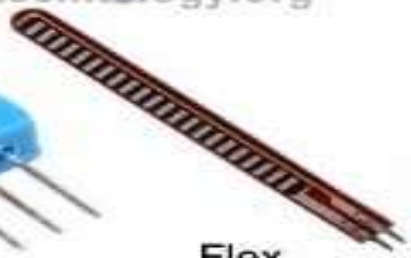
Smoke Sensor

LM35 (Temperature
Sensor)

IR
Receiver

LDR (Light
Sensor)

Humidity Sensor



Flex
Sensor

Touch Sensor

Solar Cell
Light Sensor

Metal
Dedector

Real Time
Clock Sensor

Vibration
Sensor

Sensors characteristics :Static, Dynamic

Static characteristics : how the output of a sensor changes in response to an input change after steady state condition.

i. **Accuracy:** to give a result close to the true value of the measured quantity.

ii. **Range:** Gives the highest and the lowest value of the physical quantity within which the sensor can actually sense

iii. **Resolution:** Provide the smallest changes in the input that a sensor is able to sense.

iv. **Precision: G**ive the same reading when repetitively measuring the same quantity under the same prescribed conditions.

Sensors characteristics :Static, Dynamic

Dynamic Characteristics :

- **Zero-order system:** The output shows a response to the input signal with no delay.
- **First-order system:** When the output approaches its final value gradually.
- **Second-order system:** Complex output response. The output response of the sensor oscillates before steady state.

Sensor Classification :

1. Passive & Active
2. Analog & digital
3. Scalar & vector

Passive Sensor :Can not independently sense the input. Ex- Accelerometer, soil moisture, water level and temperature sensors.

Active Sensor: Independently sense the input. Example- Radar, sounder and laser altimeter sensors.

Analog Sensor :The response or output of the sensor is some continuous function of its input parameter. Ex- Temperature sensor, LDR, analog pressure sensor and analog hall effect.

Sensor Classification :

Digital sensor :Response in binary nature. Example – Passive infrared (PIR) sensor and digital temperature sensor(DS1620).

Scalar sensor :Detects the input parameter only based on its magnitude. Example – temperature, gas, strain, color and smoke sensor.

Vector sensor :The response of the sensor depends on the magnitude of the direction and orientation of input parameter. Example – Accelerometer, gyroscope, magnetic field and motion detector sensors.

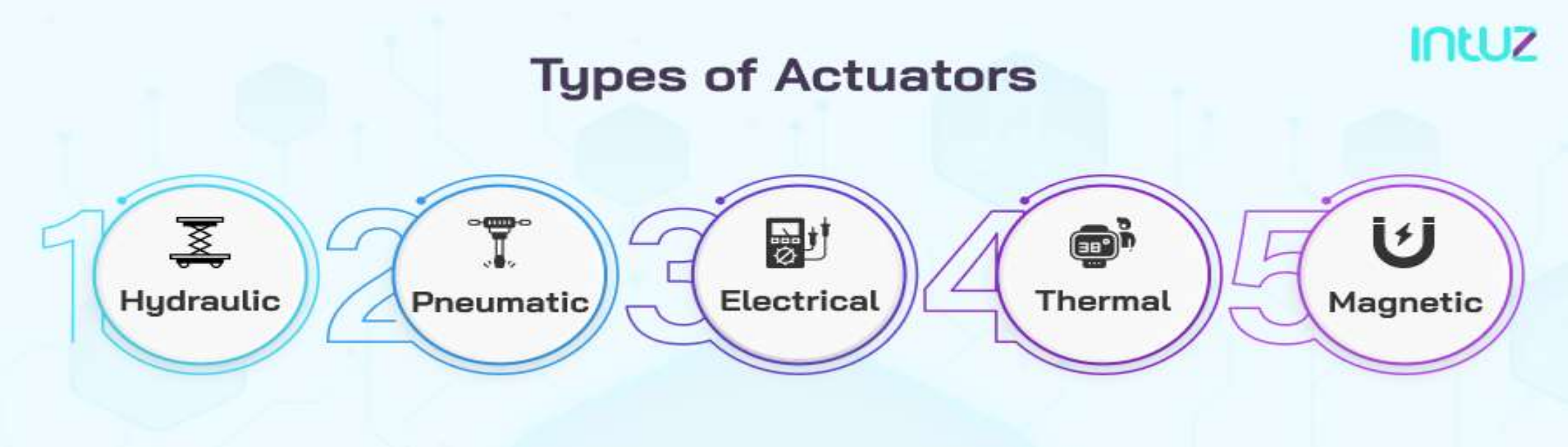
Different types of Sensors in IoT

1. Temperature: detect the temperature of the air or a physical object .
2. Pressure sensors: measure the pressure or force per unit area applied to the sensor.
3. Moisture: Where temperature sensors record the heat, moisture sensors record the amount of humidity
4. Noise: As the name suggests, record the noise levels in the given environment.
5. Level: These sensors detect the quantity or level of different substances. Ex: Manufacturing industries
6. Proximity sensors: can detect the presence or absence of objects

Different types of Sensors in IoT

7. Smoke sensors : An electronic fire-protection device that automatically senses the presence of smoke, as a key indication of fire, and sounds a warning to building occupants.
8. Infrared (IR) sensors : An electronic device, that emits the light in order to sense some object of the surroundings.
9. Gyroscope: These sensors are used to measure the velocity of a moving object.
10. Accelerometers: Accelerometers are an impressive type of IoT sensor used to record and measure an object's acceleration.

Types of Actuators



Types of Actuators

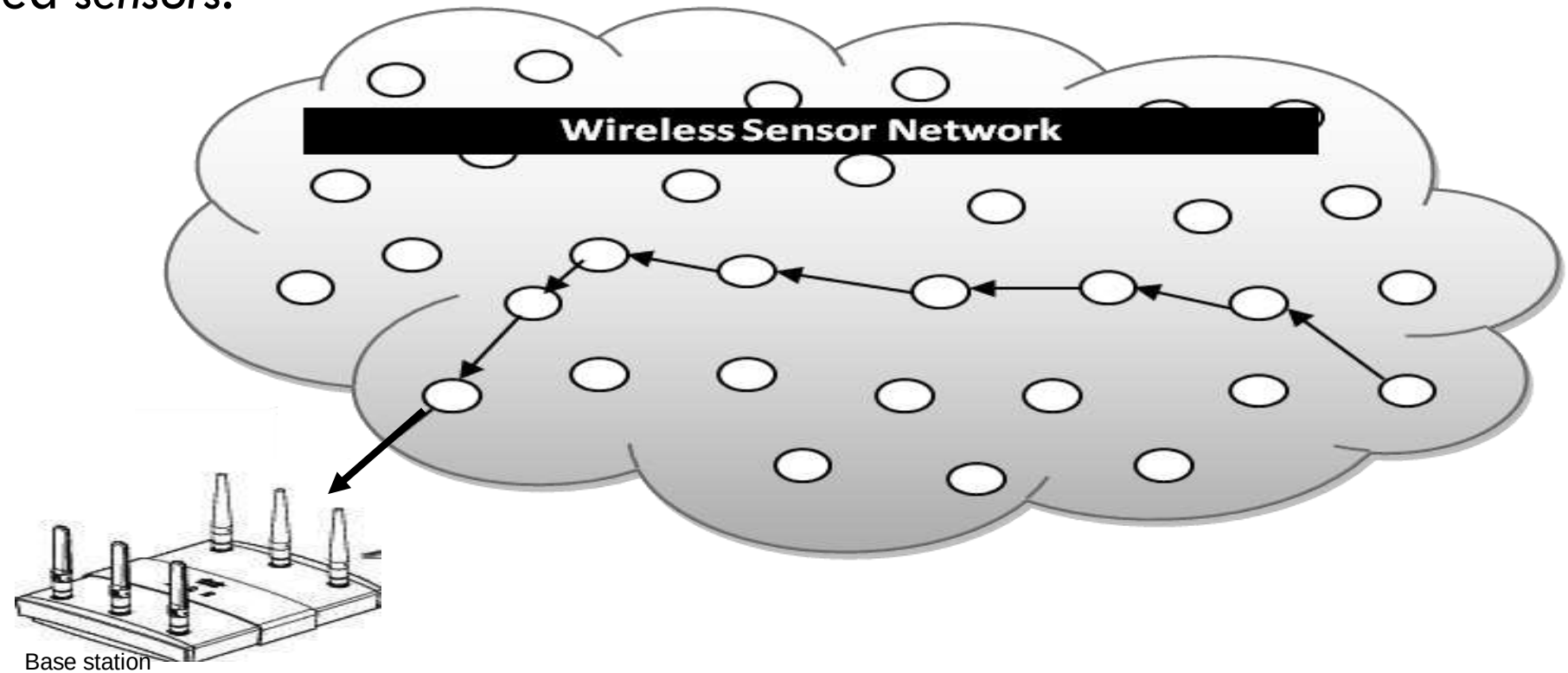
1. Hydraulic: A hydraulic actuator uses hydraulic power to perform a mechanical operation. They are actuated by a cylinder or fluid motor. The mechanical motion is converted to rotary, linear, or oscillatory motion, according to the need of the IoT device.
2. Pneumatic: A pneumatic actuator uses energy formed by vacuum or compressed air at high pressure to convert into either linear or rotary motion.
3. Electrical: An electric actuator uses electrical energy, is usually actuated by a motor that converts electrical energy into mechanical torque. An example of an electric actuator is a solenoid based electric bell.
4. Thermal: convert a temperature change into a mechanical force to push/pull, open/close, or move a load.
5. Magnetic: A device that allows for electric currents in machines to be used to move the various components within the machine

Wireless Sensor Networks (WSNs)

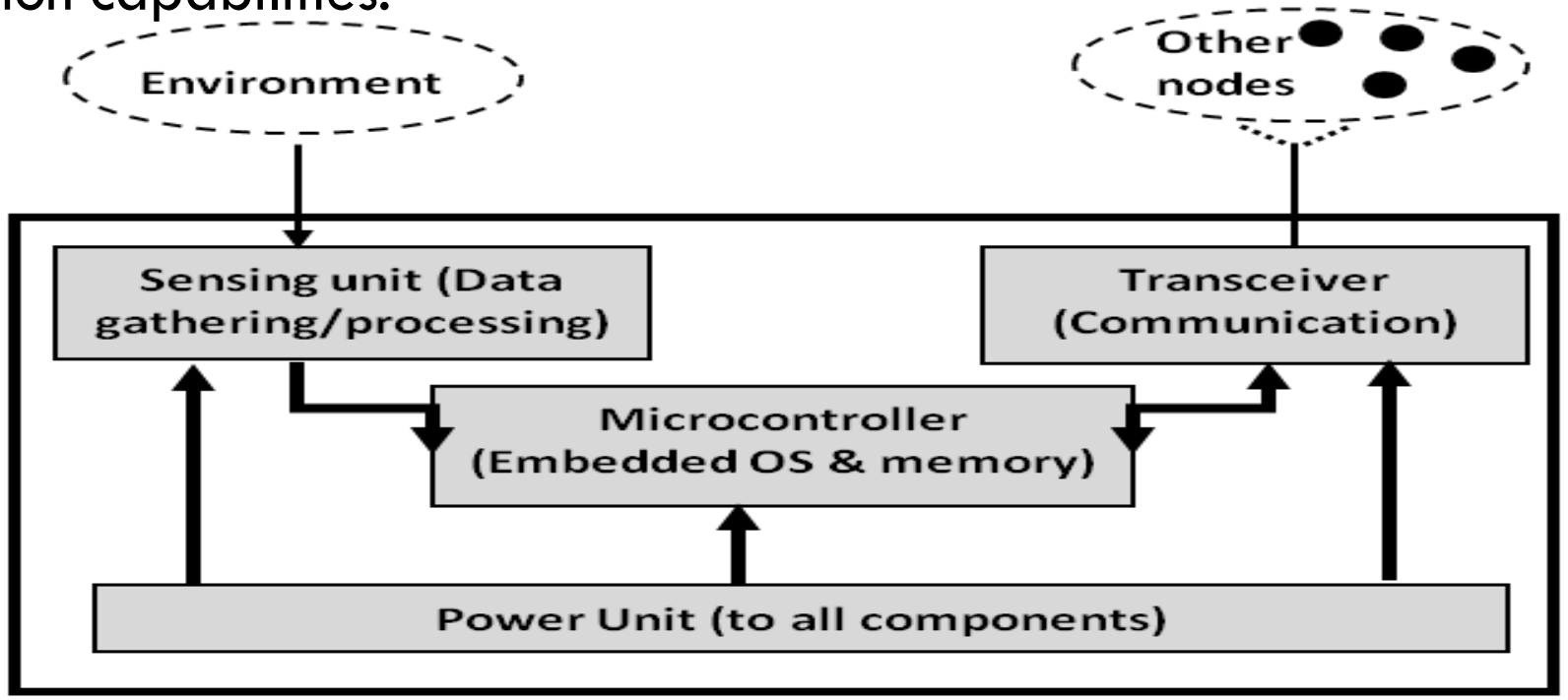
Introduction

- Wireless Sensor Networks are networks that consists of sensors which are distributed in an ad hoc manner.
- These sensors work with each other to sense some physical phenomenon and then the information gathered is processed to get relevant results.
- Wireless sensor networks consists of protocols and algorithms with self-organizing capabilities.

- A sensor network is a wireless network that consists of thousands of very small nodes called *sensors*.



- WSN **Sensors** are equipped with sensing, limited computation, and wireless communication capabilities.



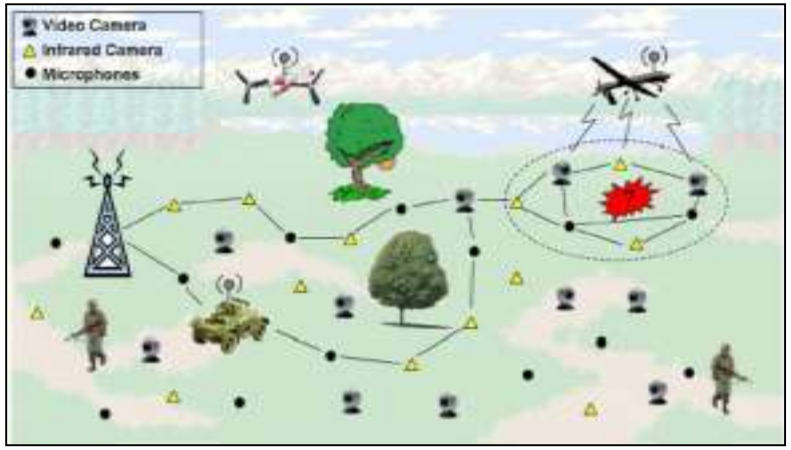
Comparison with ad hoc networks

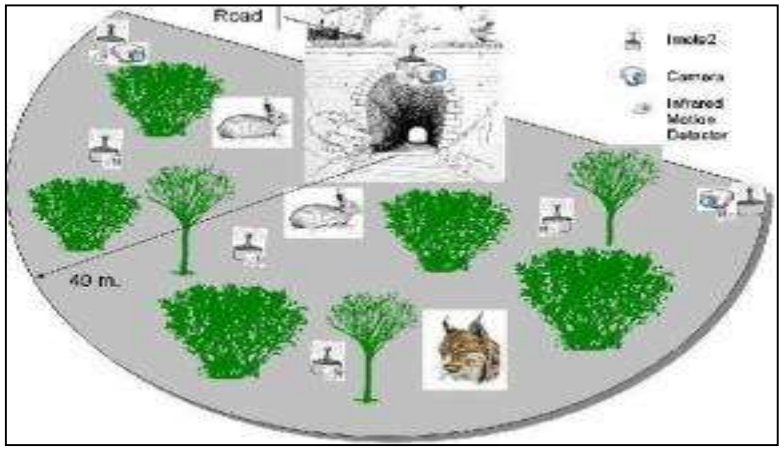
- Wireless sensor networks mainly use **broadcast** communication while ad hoc networks use **point-to-point** communication.
- Unlike ad hoc networks wireless sensor networks are **limited by sensors** limited power, energy and computational capability.
- Sensor nodes may **not have global ID** because of the large amount of overhead and large number of sensors.

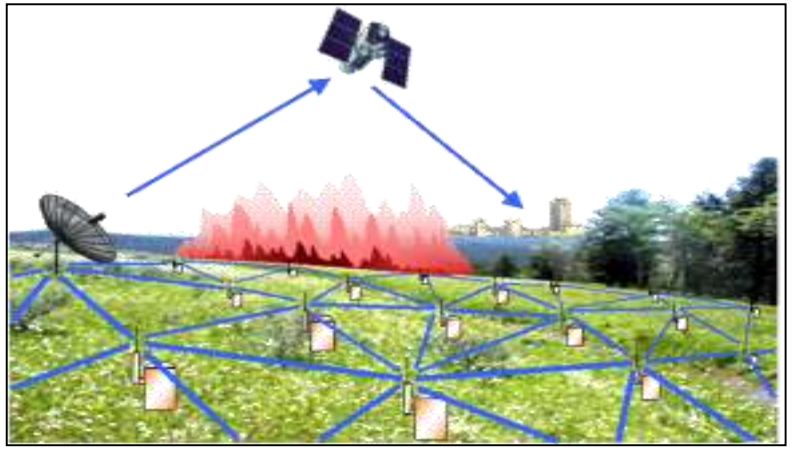
WSNs Applications

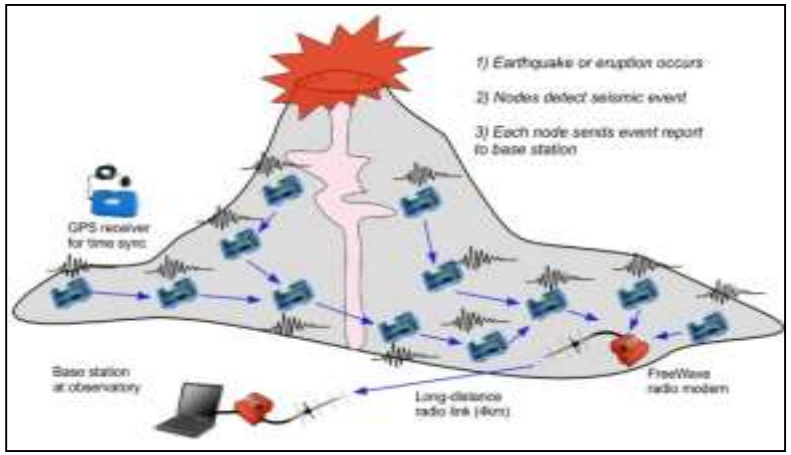
- WSNs have many advantages over traditional networking techniques.
- They have an ever-increasing number of applications, such as infrastructure protection and security, surveillance, health-care, environment monitoring, food safety, intelligent transportation, and smart energy.

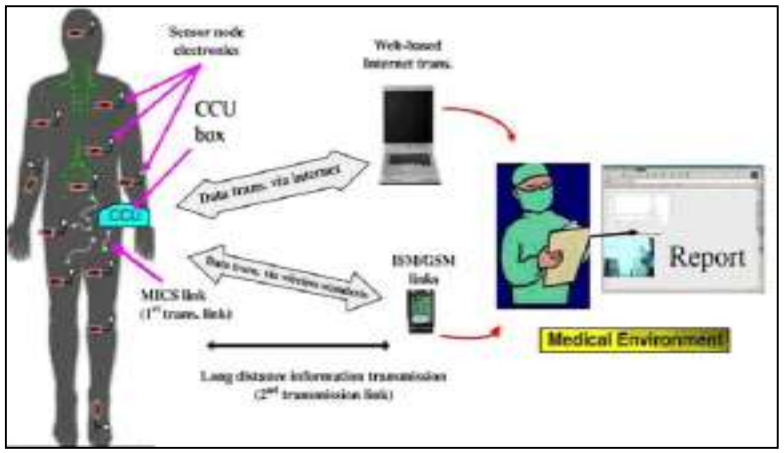
WSNs Applications

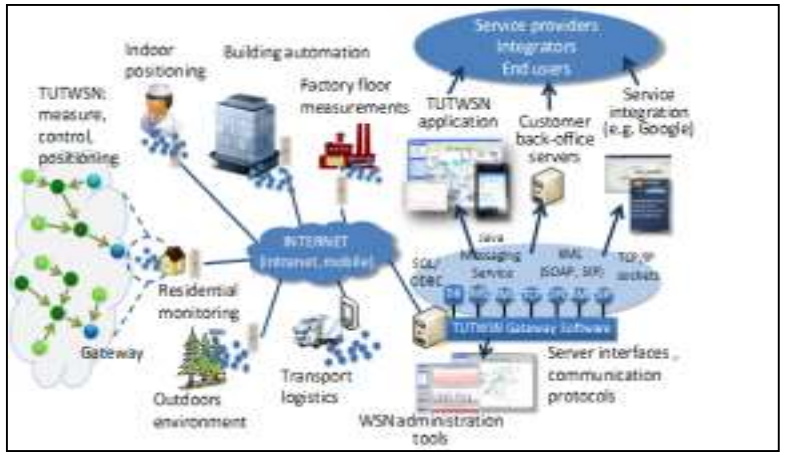












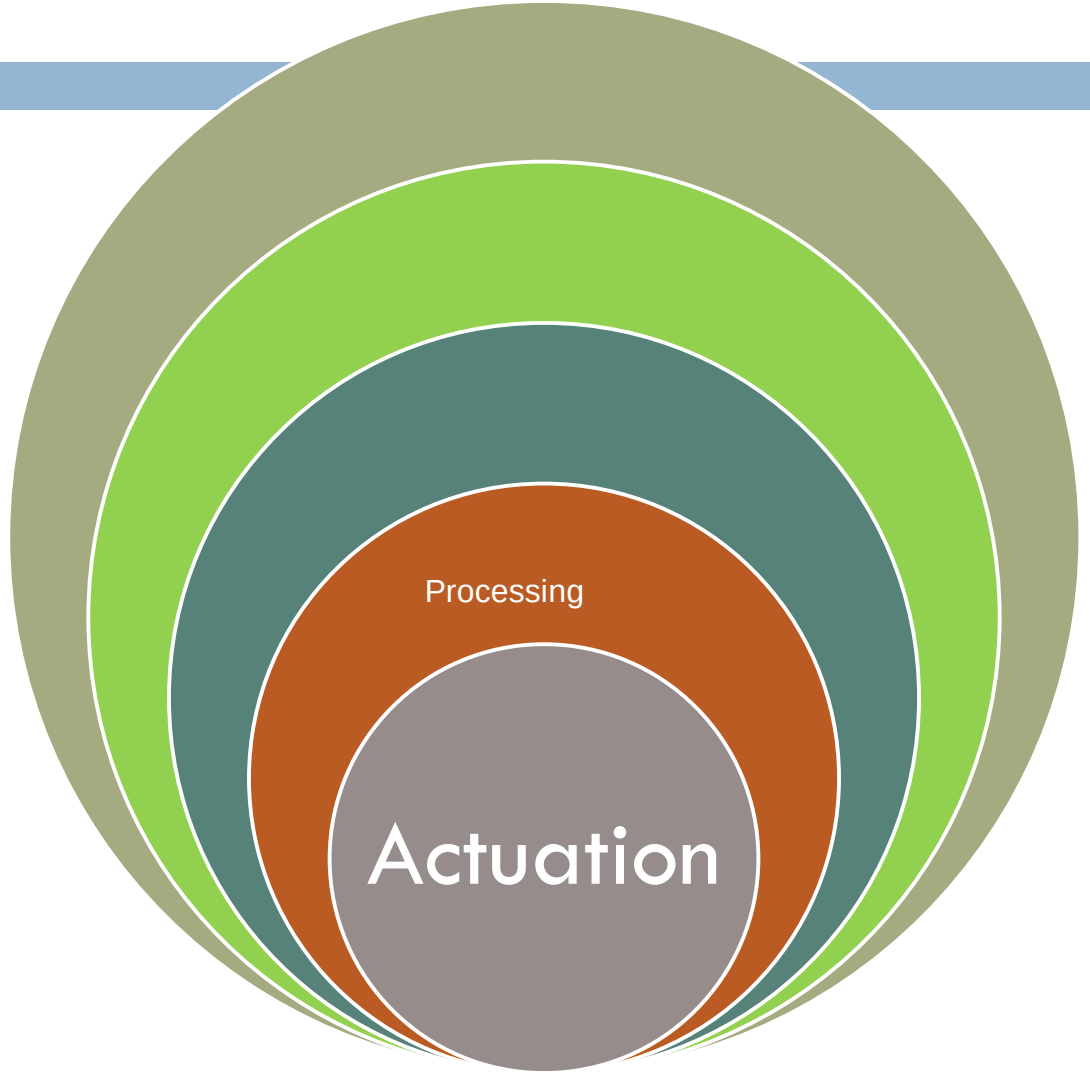
What is M2M?

- Machine to manchine learning (M2M) Communication refers to communication between the computers and embedded processors smart sensors and mobile devices with out any human intervention.
- In simple we can say “Exchange of information and perform actions without the manual assistance of humans.
- For this the modern Technologies like Artificial Intelligence (AI) and machine learning(ML) facilitate the communication between the systems and allowing the machines to make their own autonomous choices.

How M2M works?

- As machines communicate using language called ”Telemetry”.
- M2M system often use public networks and access methods to perform tasks. These devices send information they collect back to other devices on the network.
- The main components of an M2M system include like sensors, WIFI , cellular communications link and automatic computing software to help network device interpret the data and make decisions.
- M2M technology relies on software-controlled communication between machines and devices. Special applications translate information into relevant to the end user.

M2M Overview



Definition of IoT

The Internet of Things (IoT) describes physical objects embedded with sensors and actuators that communicate with computing systems via wired or wireless networks, allowing the physical world to be digitally monitored or even controlled.

Characteristics of the Internet of Things:

1. Connectivity
2. Intelligence and Identity
3. Scalability
4. Dynamic and Self-Adapting (Complexity)
5. Architecture
6. Safety
7. Self Configuring
8. Interoperability

- Connectivity: Connectivity is an important requirement of the IoT infrastructure. Things of IoT should be connected to the IoT infrastructure. Anyone, anywhere, anytime can connect, this should be guaranteed at all times. For example, the connection between people through Internet devices like mobile phones, and other gadgets, also a connection between Internet devices such as routers, gateways, sensors, etc.
- Intelligence and Identity: The extraction of knowledge from the generated data is very important. For example, a sensor generates data, but that data will only be useful if it is interpreted properly. Each IoT device has a unique identity. This identification is helpful in tracking the equipment and at times for querying its status.

- Scalability : The number of elements connected to the IoT zone is increasing day by day. Hence, an IoT setup should be capable of handling the massive expansion. The data generated as an outcome is enormous, and it should be handled appropriately.
- **Dynamic and Self-Adapting (Complexity):** IoT devices should dynamically adapt themselves to changing contexts and scenarios. Assume a camera meant for surveillance. It should be adaptable to work in different conditions and different light situations (morning, afternoon, and night).
- **Architecture:** It cannot be homogeneous in nature. It should be hybrid, supporting different manufacturers products to function in the IoT network. IoT is not owned by anyone engineering branch. IoT is a reality when multiple domains come together.

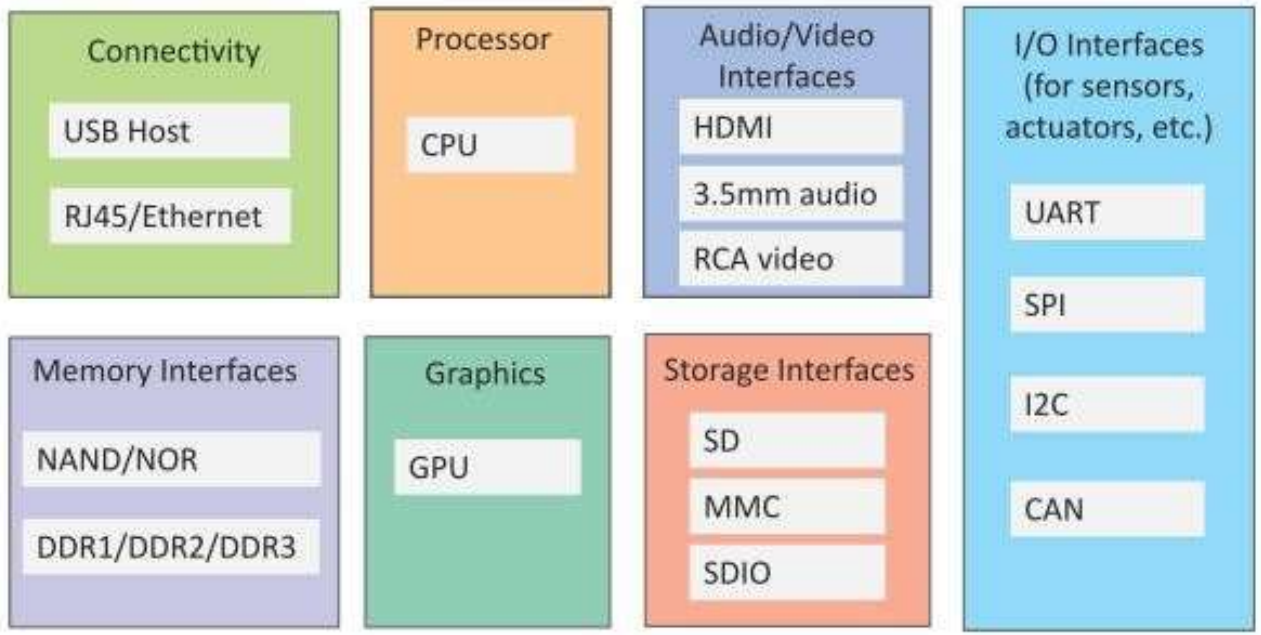
- **Safety :**There is a danger of the sensitive personal details of the users getting compromised when all his/her devices are connected to the internet. This can cause a loss to the user. Hence, data security is the major challenge. Besides, the equipment involved is huge. IoT networks may also be at risk. Therefore, equipment safety is also critical.
- **Self Configuring :**This is one of the most important characteristics of IoT. IoT devices are able to upgrade their software in accordance with requirements with a minimum of user participation. Additionally, they can set up the network, allowing for the addition of new devices to an already-existing network.
- **Interoperability :** It refers to the ability of different IoT devices and systems to communicate and exchange data with each other, regardless of the underlying technology or manufacturer.

How IoT works?

- 1) Sensors/Devices
- 2) Connectivity
- 3) Data Processing
- 4)User Interface

Generic block diagram of an IoT Device

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



PHYSICAL DESIGN OF IOT

•The **physical design** of an IoT system is referred to as the **Things/Devices** and protocols that are used to build an IoT system.

•All these things/Devices are called Node Devices and every device has a unique identity that performs remote sensing, actuating and monitoring work.

•The protocols that are used to establish communication between the Node devices and servers over the internet.

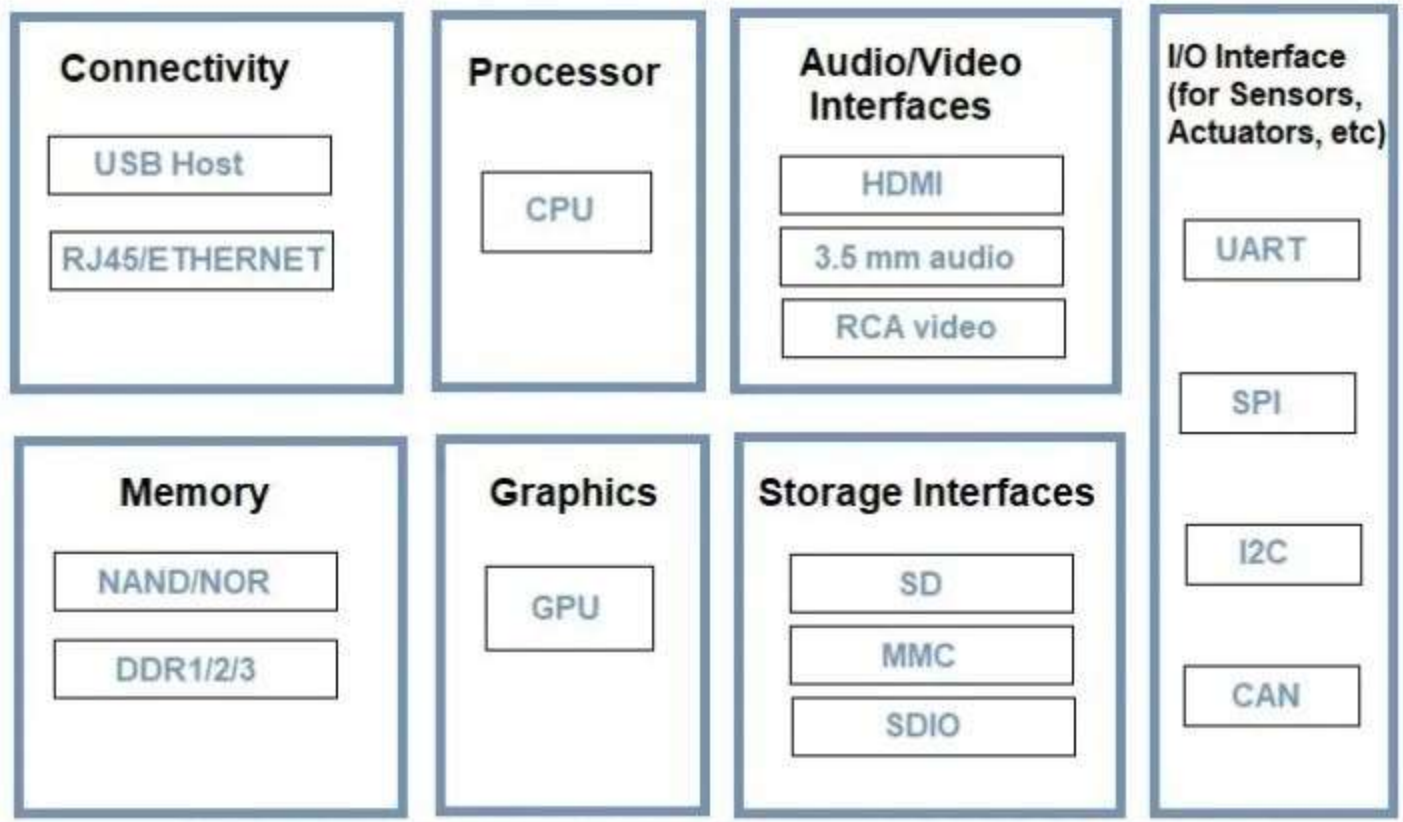
PHYSICAL DESIGN OF IOT

Things/Devices

Things/Devices are used to build a connection, process data, provide interfaces, provide storage, and provide graphic interfaces in an IoT system.

All these generate data in a form that can be analyzed by an analytical system and program to perform operations and used to improve the system.

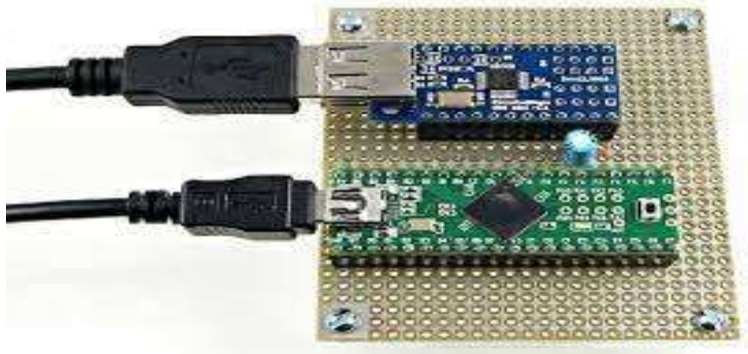
for example temperature sensor that is used to analyze the temperature generates the data from a location and is then determined by algorithms.

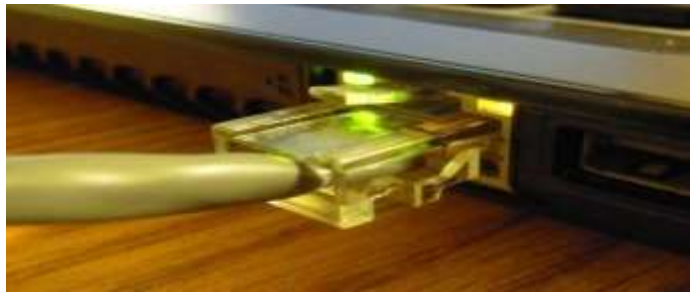


Generic Block Diagram of IoT Devices

Connectivity

Devices like USB hosts and ETHERNET are used for connectivity between the devices and the server.



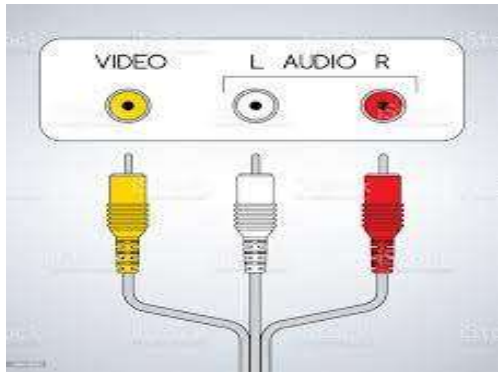


Processor

A processor like a CPU and other units are used to process the data. These data are further used to improve the decision quality of an IoT system.

Audio/Video Interfaces

An interface like HDMI and RCA devices is used to record audio and videos in a system.



Input/Output interface

To give input and output signals to sensors, and actuators we use things like UART(**universal asynchronous receiver-transmitter)**, SPI(**Serial Peripheral Interface)**, CAN(**Controller Area Network)**, etc.

Storage Interfaces

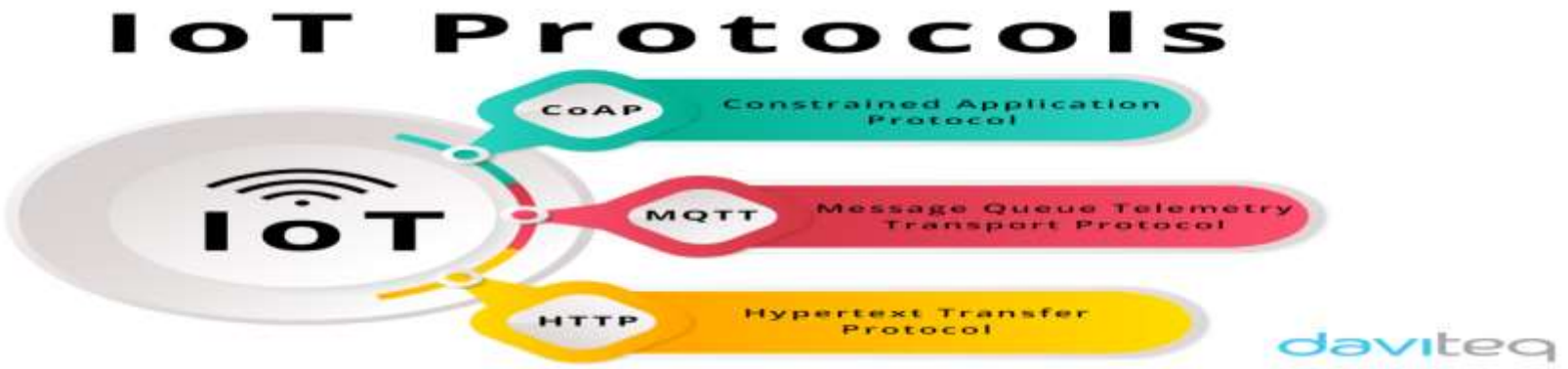
Things like SD(Secure Digital), MMC(MultiMediaCard), and SDIO are used to store the data generated from an IoT device.

(Secure Digital I/O Card) **A version of the SD Memory Card that adds wireless transmission to a handheld device**.

Other things like DDR **Double data rate** (DDR) RAM and GPU (**graphics processing unit)** are used to control the activity of an IoT system.

IOT PROTOCOLS

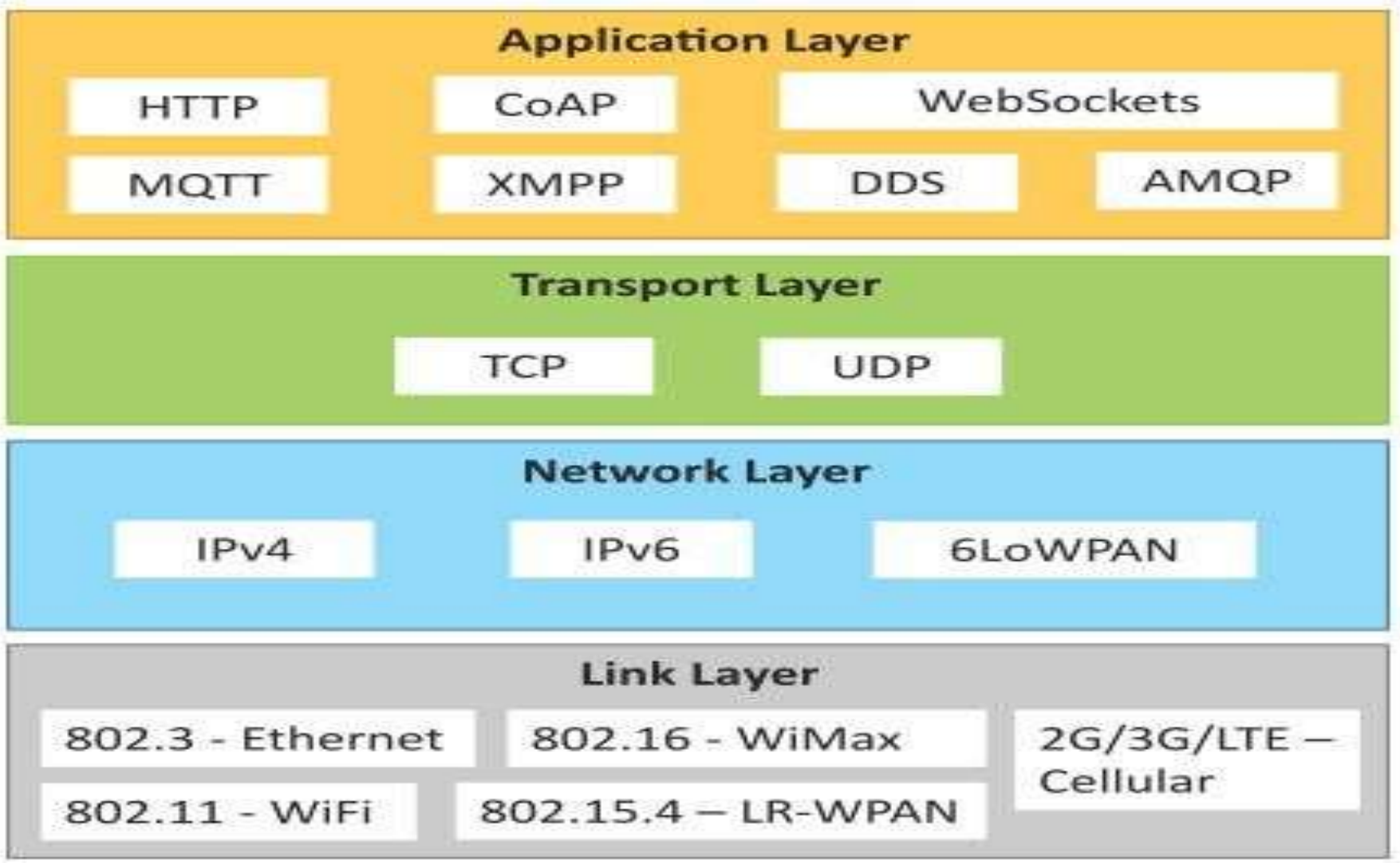
These protocols are used to establish communication between a node device and a server over the internet.
It helps to send commands to an IoT device and receive data from an IoT device over the internet.



we use different types of protocols that are present on both the server and client-side and these protocols are managed by network layers like application, transport, network, and link layer.

IoT Protocols

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



Layer	Protocol
Application layer	•Advanced Message Queuing Protocol (AMQP) •Message Queue Telemetry Transport (MQTT) •Constrained Application Protocol (CoAP)
Transport layer	•User Datagram Protocol (UDP) •Transmission Control Protocol (TCP)
Network layer	•6LoWPAN •IP
Datalink layer	•LPWAN •IEEE 802.15.4 MAC
Physical layer	•IEEE 802.15.4 MAC •Near field communication (NFC) •Radio frequency identification (RFID) •Bluetooth Low Energy (BLE) •Ethernet

Logical design of IoT

Logical design of IoT system refers to an abstract representation of the entities & processes without going into the low-level specifies of the implementation. For understanding Logical Design of IoT, we describes given below terms.

IoT Functional Blocks
IoT Communication Models
IoT Communication APIs

IoT Functional Blocks

➢These functional blocks consist of devices that provide monitoring control functions, handle communication between host and server, manage the transfer of data, secure the system using authentication and other functions, and interface to control and monitor various terms.

Application

•It is an interface that provides a control system that use by users to view the status and analyze of system.

Management

•This functional block provides various functions that are used to manage an IoT system.

Services

•This functional block provides some services like monitoring and controlling a device and publishing and deleting the data and restoring the system.

Communication

•This block handles the communication between the client and the cloud-based server and sends/receives the data using protocols.

Security

•This block is used to secure an IoT system using some functions like authorization, data security, authentication, 2-step verification, etc.

Device

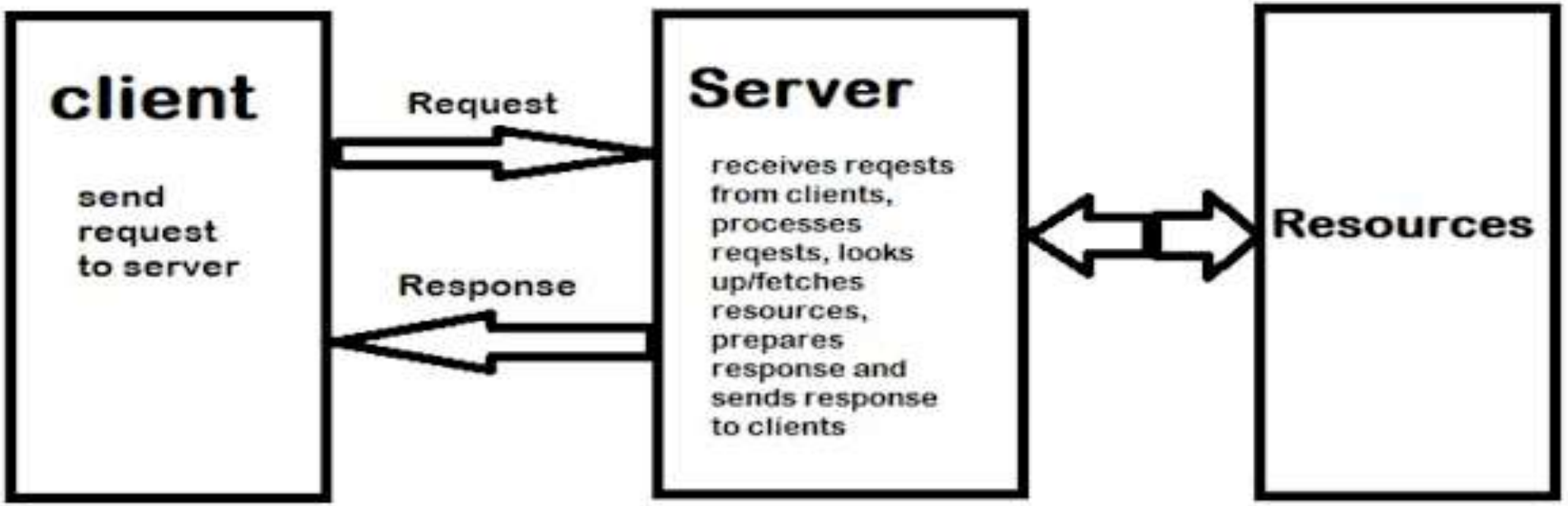
•These devices are used to provide sensing and monitoring control functions that collect data from the outer environment.

IoT Communication Models

There are several different types of models available in an IoT system that is used to communicate between the system and server like

1. Request-Response Communication Model
2. Publish-Subscribe Communication Model
3. Push-Pull
4. Exclusive Pair

1. Request-Response Communication Model



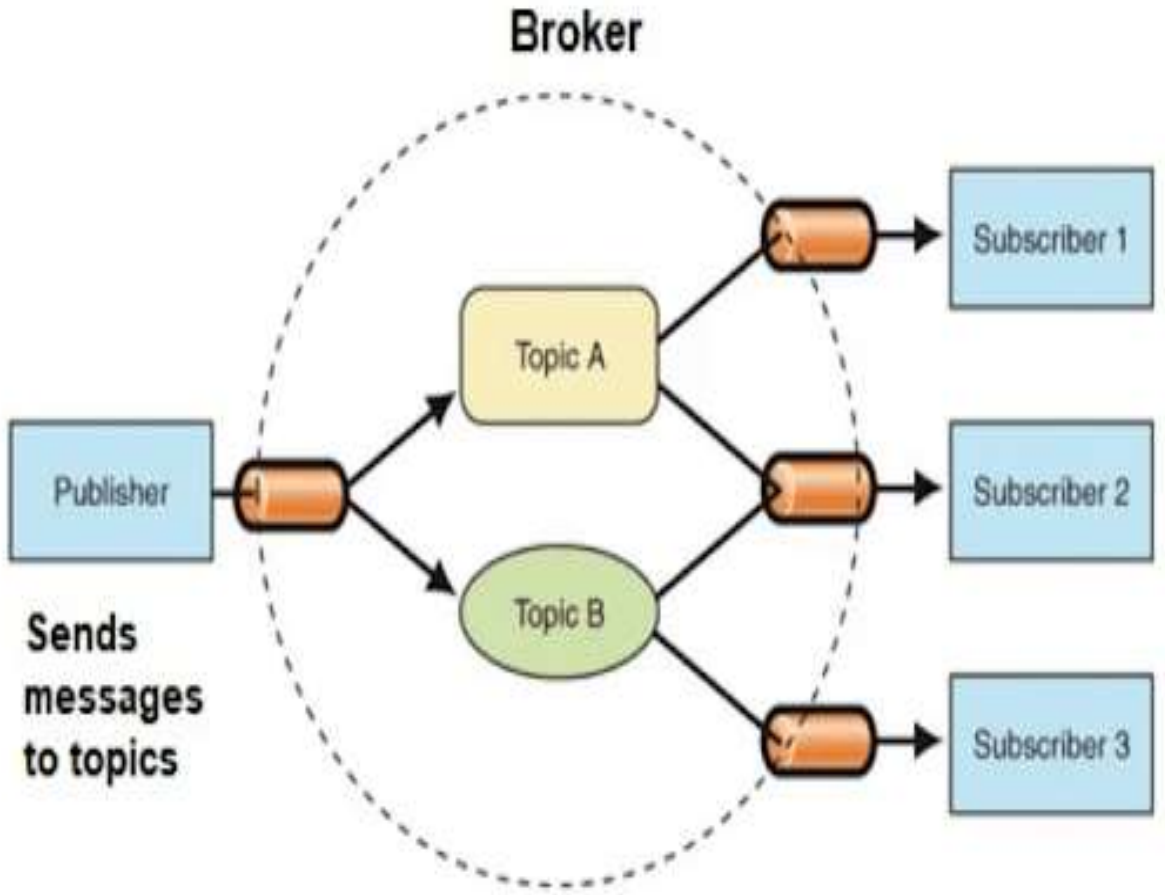
Request-Response Communication Model

2. Publish-Subscribe Communication Model

Example

•On the website many times we subscribed to their newsletters using our email address.

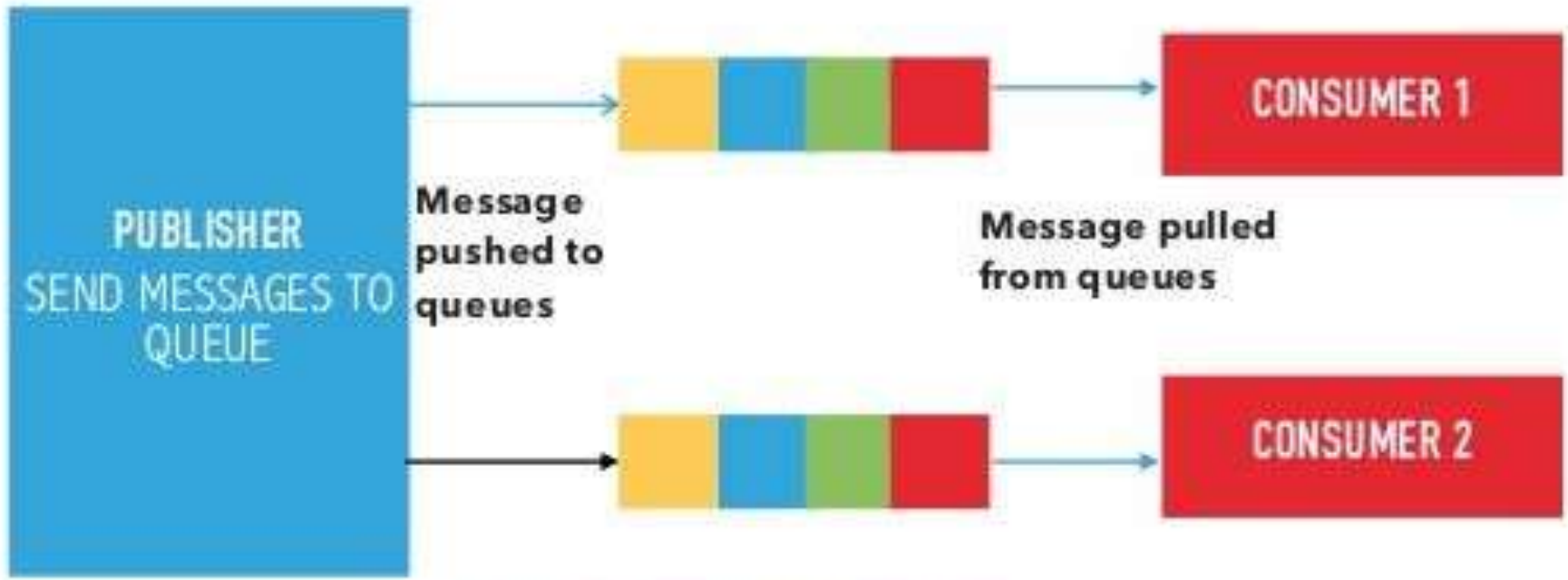
•These email addresses are managed by some third-party services and when a new article is published on the website it is directly sent to the broker and then the broker sends these new data or posts to all the subscribers.



3. Push-Pull Communication Model

EXAMPLE:

Company Newsletter: Imagine a company that sends out a weekly newsletter to all its employees. The human resources department compiles the latest updates, announcements, and important news regarding the company's policies, events, and achievements. The newsletter is sent to all employees' email addresses without them having to request it actively. This is an example of the push communication model, where information is pushed out to recipients regardless of their immediate need or request..



PUSH PULL MODEL

4. Exclusive Pair Communication Model

Example: Alice and Bob's Private Chatroom

- Alice and Bob are two close friends who prefer private and exclusive communication.
- They decide to create an exclusive pair communication model by setting up a private chatroom. In this chatroom, only they have access, and all communication between them is encrypted to ensure privacy and security.

EXCLUSIVE PAIR COMMUNICATION MODEL

IoT Communication API's

REST-based Communication APIs :

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

IoT Enabling Technologies

- **Wireless Sensor Network**

- **Cloud Computing**

- **Big Data Analytics**

- **Communication Protocols**

- **Embedded Systems**

1. Wireless Sensor Network

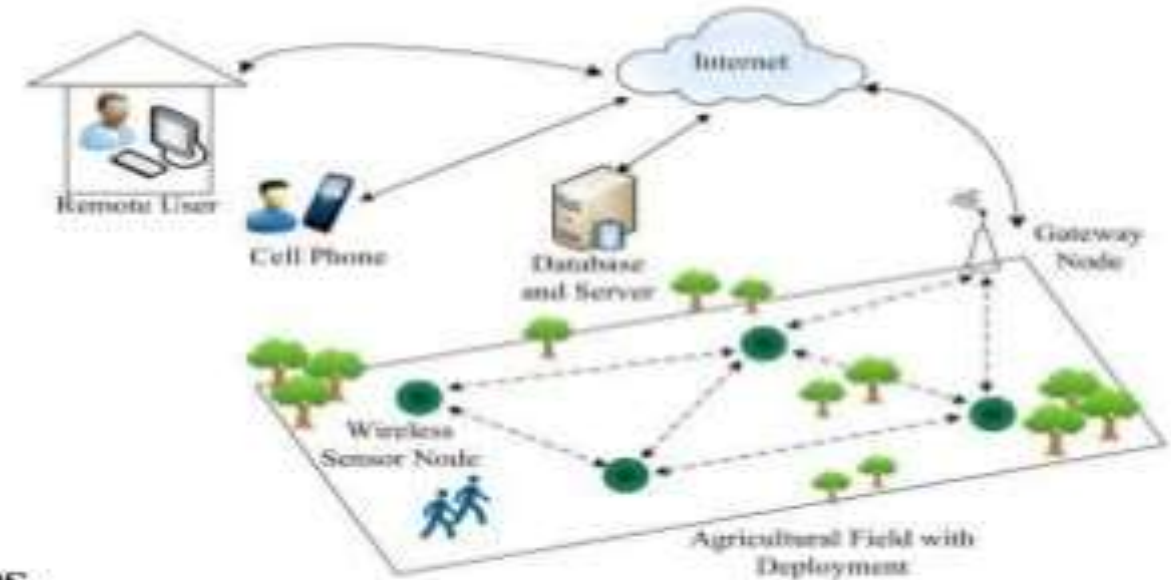
- **Distributed Devices with sensors** used to monitor the environmental and physical conditions

Or

- It is a network formed by **large no. of sensor nodes** to detect light, heat , pressure ect.

i.e. used to monitor environmental and physical conditions.

- Each node can have several sensors attached to it.
- Each node can also acts as a routers
- **Coordinator** collects data from all nodes
- Coordinator acts as **gateway** that connects WSN to the internet.



Examples of WSNs

- Indoor Air Quality Monitoring system
- Weather Monitoring System
- Soil Moisture Monitoring System
- Survelliance Systems
- Health Monitoring Systems

Protocols used

WSNs are enabled by wireless communication protocols such as **IEEE802.15.4**

Zigbee is one of the most popular wireless technology used by WSNs. Zigbee specifications are based on **IEEE802.15.4** which is used for low powered devices.

Data rate: up to **250KBps**. Range: upto **100 Meters**

CLOUD COMPUTING

- IoT devices provide a bridge between the digital world and the physical world by capturing information about the environment, so remote consumers can make decisions and initiate actions.

- Those actions may be as simple as turning a light on or off or part of a complex operational process (such as a manufacturing system)

- Once data is in the cloud, it can be merged with other data, analyzed for meaning and relevance and, in some cases, used to drive automation. There are many business problems IoT can help companies solve.

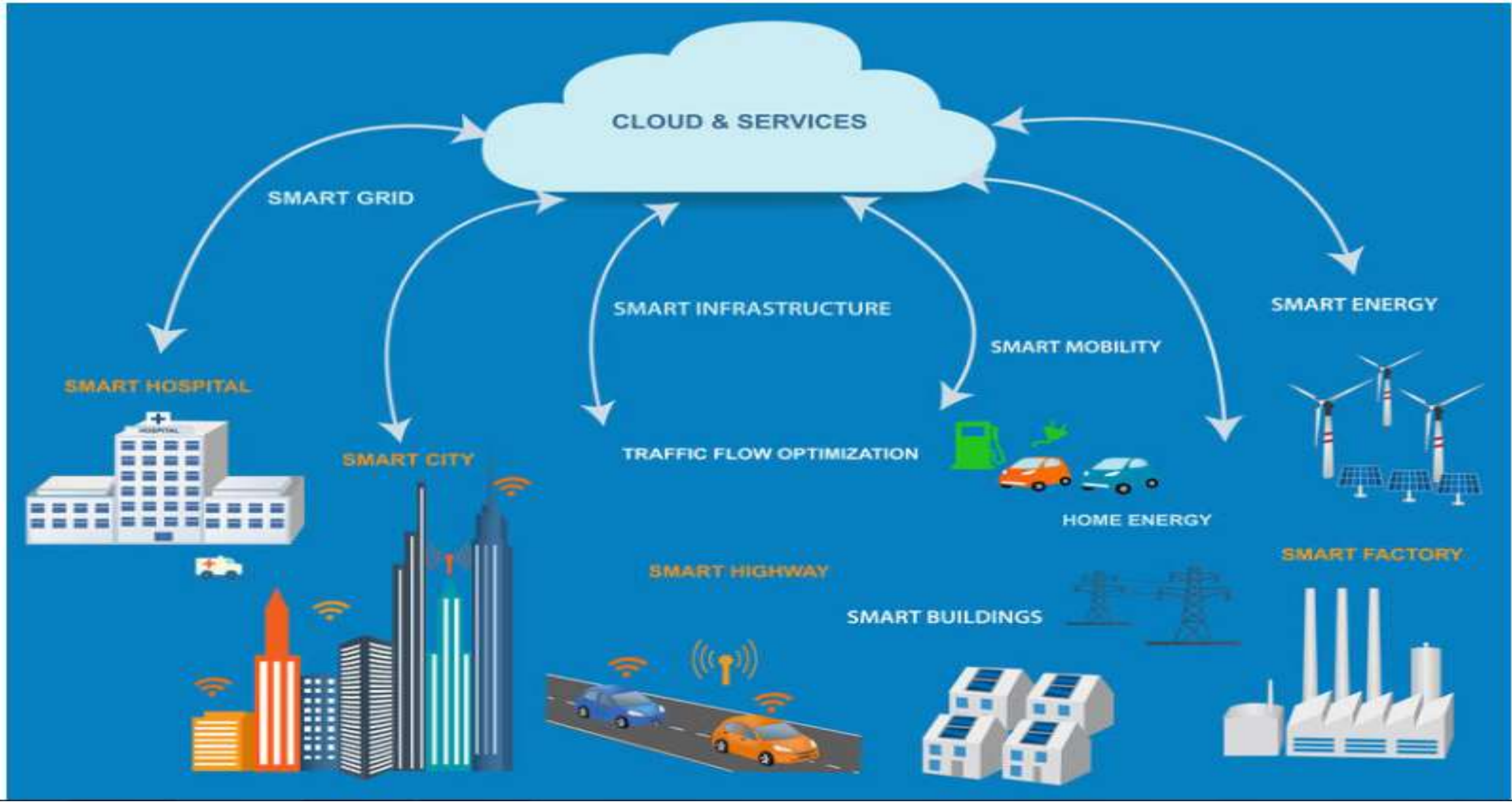
Cloud Computing: Services are offered to users in different forms.

Cloud computing services are offered to users in different forms:

Infrastructure as a Service (IaaS): Hardware is provided by an external provider and managed for you

Platform as a Service (PaaS): In addition to hardware, your operating system layer is managed for you

Software as a Service (SaaS): further to the above, an application layer is provided and managed for you – you won't see or have to worry about the first two layers.


CLOUD & SERVICES
SMART GRID
SMART INFRASTRUCTURE
SMART ENERGY
SMART MOBILITY
SMART HOSPITAL
SMART CITY
TRAFFIC FLOW OPTIMIZATION
HOME ENERGY
SMART HIGHWAY
SMART FACTORY
SMART BUILDINGS

Big Data Analytics

Some examples of big data generated by IoT are

- Sensor data generated by IoT systems.
- Machine sensor data collected from sensors established in industrial and energy systems.
- Health and fitness data generated IoT devices.
- Data generated by IoT systems for location and tracking vehicles.
- Data generated by retail inventory monitoring systems.

Communication Protocols

These form the back-bone of IoT systems and enable network connectivity and coupling to applications.

Allow devices to exchange data over network.

Define the exchange formats, data encoding addressing schemes for device and routing of packets from source to destination.

It includes sequence control, flow control and retransmission of lost packets.

Embedded Systems

Embedded Systems is a computer system that has computer hardware and software embedded to perform specific tasks. Embedded System range from low cost miniaturized devices such as digital watches to devices such as digital cameras, POS terminals, vending machines, appliances etc.,

IoT Levels & Deployment Templates

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

IoT Levels & Deployment Templates

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

IoT Level-1

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

IoT Level- 2

- A level-2 IoT system has a single node that performs sensing and/or actuation and local analysis
- suitable for solutions where the data involved is big and primary analysis requirement is not computationally intensive

IoT Level- 3

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

IoT Level- 4

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

IoT Level- 5

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

MODULE II

IoT Devices and Endpoints

Outline

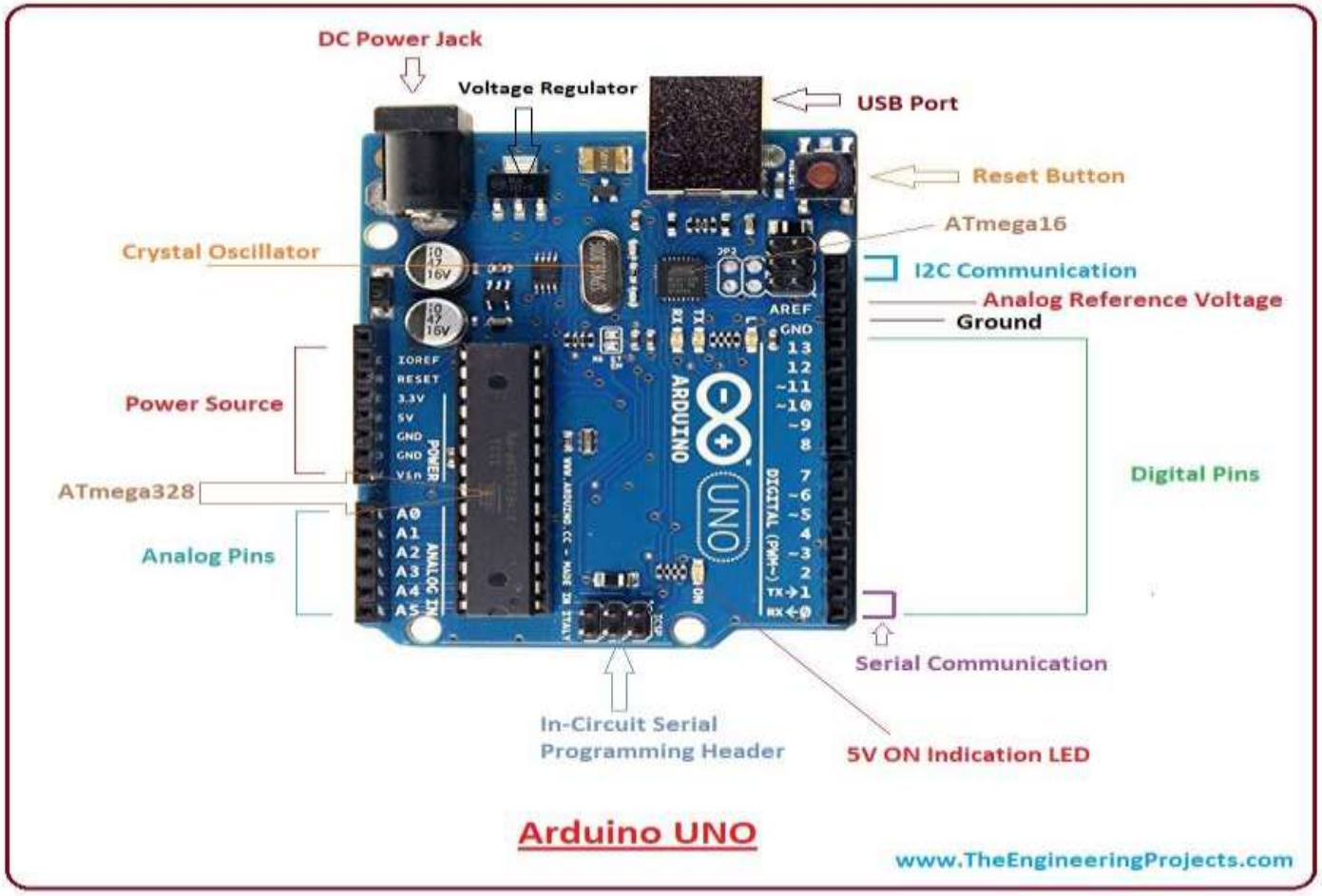
- Introduction to Arduino UNO
- Pin layout, Installing the Software
- Fundamentals of Arduino Programming.
- Introduction to Raspberry Pi
- Pin Layout, Operating Systems on Raspberry Pi
- Installing Raspberry Pi, Connecting Raspberry Pi via SSH
- Raspberry Pi Interfaces, Interfacing Hardware with the Raspberry Pi
- Raspberry Pi Remote Access and “headless mode”.

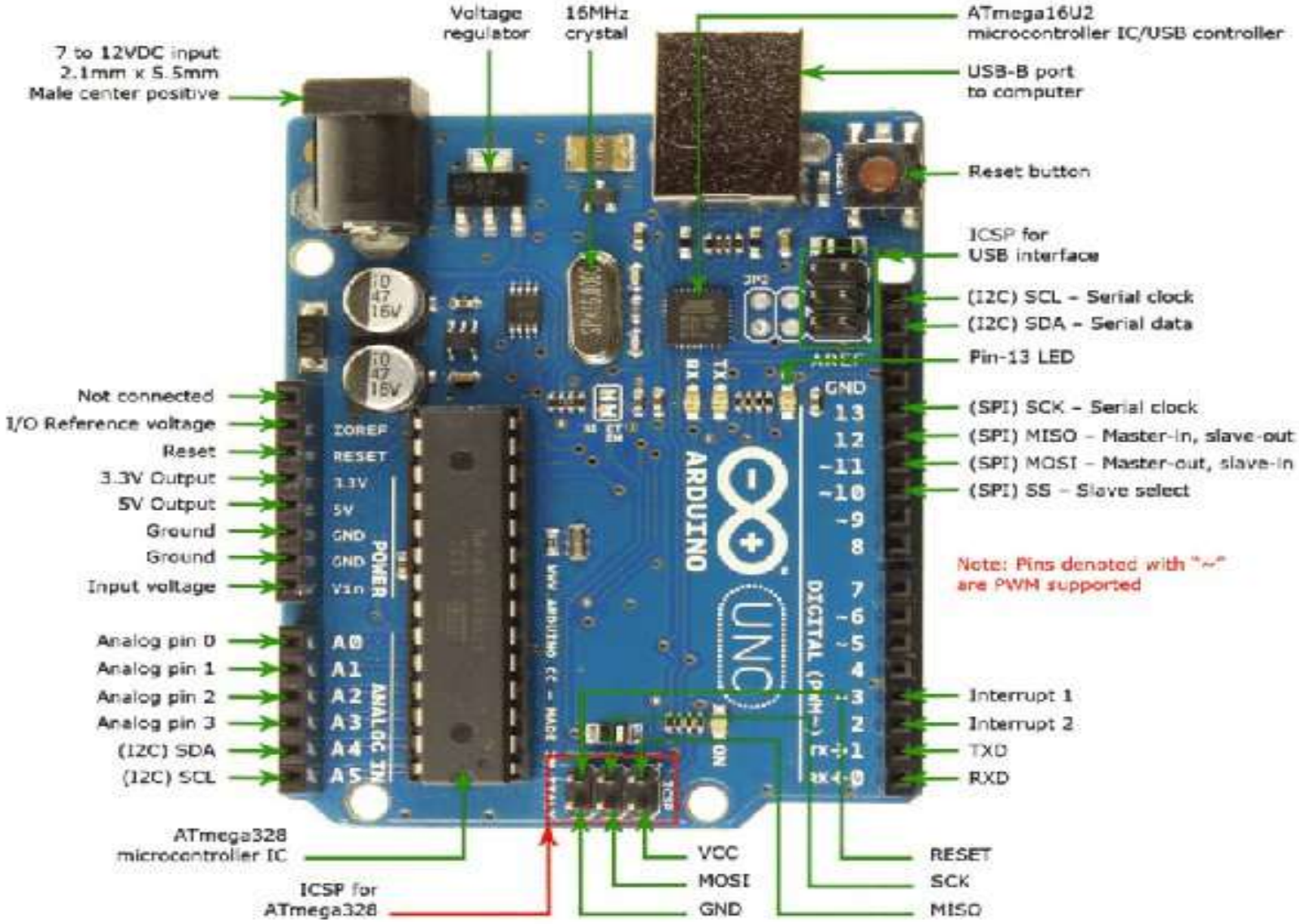
Introduction to Arduino UNO

- Arduino UNO is a low-cost, flexible, and easy-to-use programmable open-source microcontroller board that can be integrated into a variety of electronic projects.
- **Arduino Uno** is a microcontroller board, developed by Arduino.cc, based on the Atmega328 microcontroller and is marked as the first Arduino board developed(UNO means "one" in Italian).
- The software used for writing, compiling & uploading code to Arduino boards is called **Arduino IDE** (Integrated Development Environment), which is free to download from Arduino Official Site.
- It has an **operating voltage of 5V** while the input voltage may vary from 7V to 12V. Arduino UNO has a **maximum current rating of 40mA**, so the load shouldn't exceed this current rating or you may harm the board.

- Arduino UNO comes with 3 types of memories associated with it, named:
  - Flash Memory: 32KB
  - SRAM: 2KB
  - EEPROM: 1KB
- Arduino UNO supports 3 types of communication protocols, used for interfacing with third-party peripherals, named:
  - Serial Protocol
  - I2C Protocol
  - SPI Protocol

Pin layout


DC Power Jack
Voltage Regulator
USB Port
Reset Button
ATmega16
Crystal Oscillator
I2C Communication
Analog Reference Voltage
Ground
Power Source
ATmega328
Digital Pins
Analog Pins
Serial Communication
In-Circuit Serial
Programming Header
5V ON Indication LED
Arduino UNO
www.TheEngineeringProjects.com


Voltage regulator
16MHz crystal
ATmega16U2 microcontroller IC/USB controller
7 to 12VDC input 2.1mm x 5.5mm Male center positive
USB-B port to computer
Reset button
ICSP for USB interface
(I2C) SCL – Serial clock
(I2C) SDA – Serial data
Pin-13 LED
Not connected
I/O Reference voltage
Reset
3.3V Output
5V Output
Ground
Ground
Input voltage
(SPI) SCK – Serial clock
(SPI) MISO – Master-in, slave-out
(SPI) MOSI – Master-out, slave-in
(SPI) SS – Slave select
Note: Pins denoted with "~" are PWM supported
Analog pin 0
Analog pin 1
Analog pin 2
Analog pin 3
(I2C) SDA
(I2C) SCL
Interrupt 1
Interrupt 2
TXD
RXD
ATmega328 microcontroller IC
ICSP for ATmega328
VCC
MOSI
GND
RESET
SCK
MISO

- There are several I/O digital and analog pins placed on the board which operates at 5V. These pins come with standard operating ratings ranging between 20mA to 40mA. Internal pull-up resistors are used in the board that limits the current exceeding the given operating conditions. However, too much increase in current makes these resisters useless and damages the device.
- **Arduino Uno Pinout** consists of 14 digital pins starting from **D0 to D13**.
- It also has **6 analog pins** starting from **A0 to A5**.
- It also has **1 Reset Pin**, which is used to reset the board programmatically. In order to reset the board, we need to make this pin LOW.
- It also has **6 Power Pins**, which provide different voltage levels
- Out of 14 digital pins, 6 pins are used for generating PWM pulses of 8-Bit resolution. PWM pins in Arduino UNO are **D3, D5, D6, D9, D10 and D11**.

- **LED.** Arduino Uno comes with a built-in LED which is connected through pin 13. Providing HIGH value to the pin will turn it ON and LOW will turn it OFF.
- **Vin.** It is the input voltage provided to the Arduino Board. It is different than 5 V supplied through a USB port. This pin is used to supply voltage. If a voltage is provided through a power jack, it can be accessed through this pin.
- **5V.** This board comes with the ability to provide voltage regulation. 5V pin is used to provide output regulated voltage. The board is powered up using three ways i.e. USB, Vin pin of the board or DC power jack.
- USB supports voltage around 5V while Vin and Power Jack support a voltage ranges between 7V to 20V. It is recommended to operate the board on 5V. It is important to note that, if a voltage is supplied through 5V or 3.3V pins, they result in bypassing the voltage regulator that can damage the board if the voltage surpasses its limit.

- **GND.** These are ground pins. More than one ground pins are provided on the board which can be used as per requirement.
- **Reset.** This pin is incorporated on the board which resets the program running on the board. Instead of physical reset on the board, IDE comes with a feature of resetting the board through programming.
- **IOREF.** This pin is very useful for providing voltage reference to the board. A shield is used to read the voltage across this pin which then selects the proper power source.
- **PWM.** PWM is provided by 3,5,6,9,10, 11pins. These pins are configured to provided 8-bit output PWM.
- **SPI.** It is known as Serial Peripheral Interface. Four pins 10(SS), 11(MOSI), 12(MISO), 13(SCK) provide SPI communication with the help of the SPI library.

- **AREF.** It is called Analog Reference. This pin is used for providing a reference voltage to the analog inputs.
- **TWI.** It is called Two-wire Interface. TWI communication is accessed through Wire Library. A4 and A5 pins are used for this purpose.
- **Serial Communication.** Serial communication is carried out through two pins called Pin 0 (Rx) and Pin 1 (Tx).
- Rx pin is used to receive data while Tx pin is used to transmit data.
- **External Interrupts.** Pin 2 and 3 are used for providing external interrupts. An interrupt is called by providing LOW or changing value.
- Apart from USB, a battery or AC to DC adopter can also be used to power the board.

Arduino Uno applications

- Embedded System
- Security and Defense System
- Digital Electronics and Robotics
- Parking Lot Counter
- Weighing Machines
- Traffic Light Count Down Timer
- Medical Instrument
- Emergency Light for Railways
- Home Automation
- Industrial Automation

Installing the Software
Integrated Development Environment(IDE)

- Step 1 – First you must have your Arduino board (you can choose your favorite board) and a USB cable. ...
- Step 2 – Download Arduino IDE Software.
- Step 3 – Power up your board.
- Step 4 – Launch Arduino IDE.
- Step 5 – Open your first project.
- Step 6 – Select your Arduino board.

Fundamentals of Arduino Programm

- Each block has a set of statements enclosed in curly braces:
- void setup( )
- {
- statements-1;
- .
- .
- .
- statement-n;
- }
- void loop ( )
- {
- statement-1;
- .
- .
- statement-n;
- }

- Arduino programs have a minimum of 2 blocks,
- Preparation & Execution
- Here, setup ( ) is the preparation block and loop ( ) is an execution block.

- The setup block is the first to execute when the program is executed, and this function is called only once.
- The setup function is used to initialize the pin modes and start serial communication.
- This function has to be included even if there are no statements to execute.

- void setup ( )
- {
- pinMode (pin-number, OUTPUT); // set the ‘pin-number’ as output
- pinMode (pin-number, INPUT); // set the ‘pin-number’ as output
- }

- The execution block hosts statements like reading inputs, triggering outputs, checking conditions etc
- Void loop ( )
- {
- digitalWrite (pin-number,HIGH); **// turns ON the component connected to ‘pin-number’**
- delay (1000); // wait for 1 sec
- digitalWrite (pin-number, LOW); **// turns OFF the component connected to ‘pin-number’**
- delay (1000); //wait for 1sec
- }

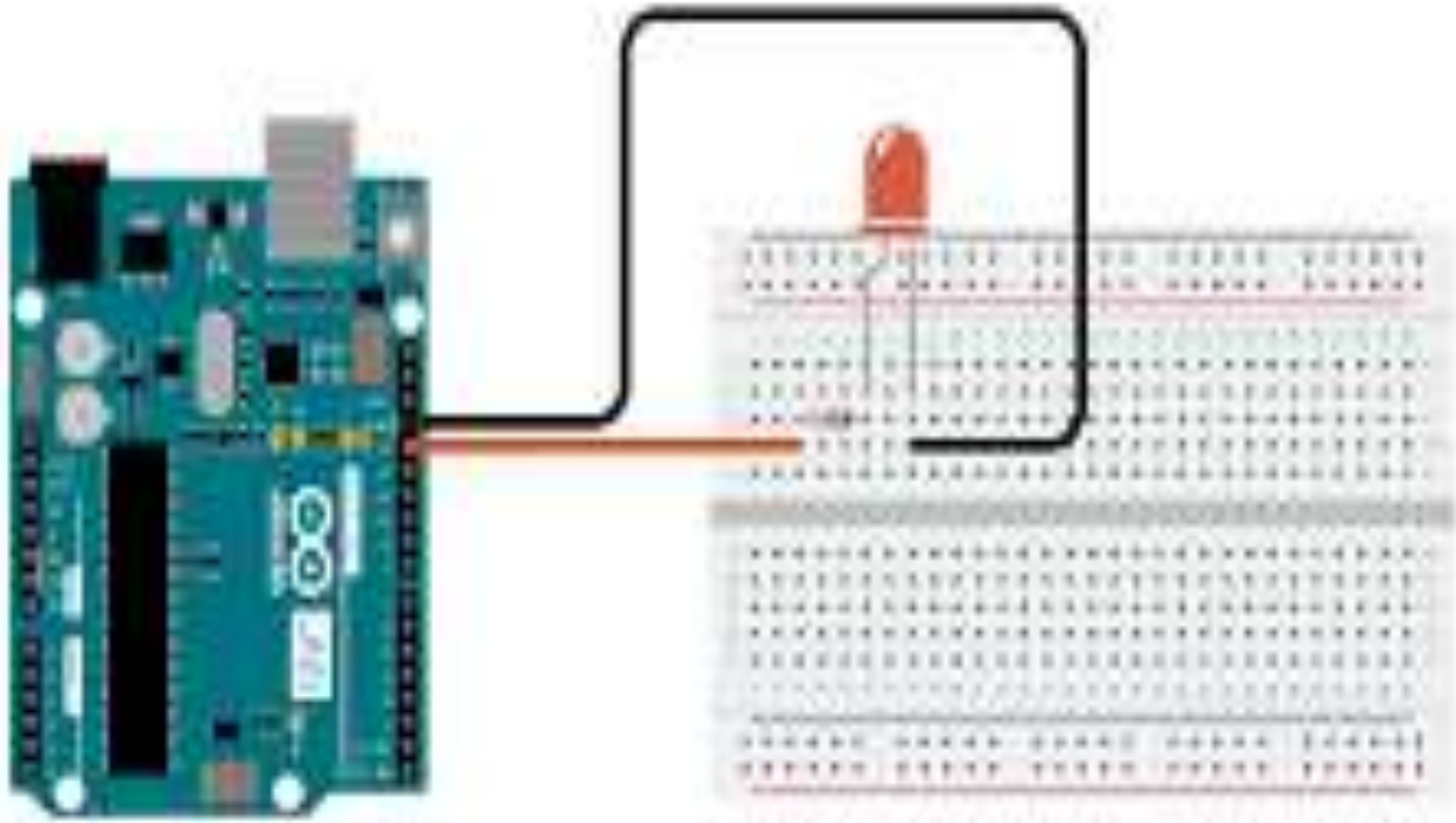
Experiments with Arduino

Components Required for LED experiment

- Arduino UNO R3 -1
- Breadboard -1
- Breadboard connectors -3
- LED -1
- 1K resistor -1

- # Blinking the LED
- Steps in building a breadboard connection:
- **Step-1**: Connect the Arduino to the Windows / Mac / Linux system via a USB cable
- **Step-2**: Connect the 13th digital pin of Arduino to the positive power rail of the breadboard and GND to the negative
- **Step-3**: Connect the positive power rail to the terminal strip via a 1K ohm resistor
- **Step-4**: Fix the LED to the ports below the resistor connection in the terminal strip
- **Step-5**: Close the circuit by connecting the cathode (the short chord) of the LED to the negative power strip of the breadboard

- **Arduino program for LED blink (Version-1)**
- int LED =13; // The digital pin to which the LED is connected
- void setup ( )
- {
- pinMode (LED, OUTPUT); //Declaring pin 13 as output pin
- }
- void loop( ) // The loop function runs again and again
- {
- digitalWrite (LED, HIGH); //Turn ON the LED
- delay(1000); //Wait for 1sec
- digitalRead (LED, LOW); // Turn off the LED
- delay(1000); // Wait for 1sec
-



Fade-in and fade-out the LED

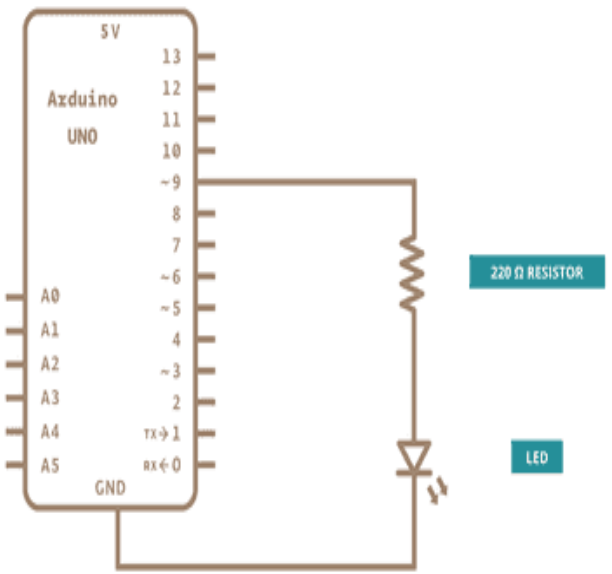
- **Components Required**:
- Arduino board
- LED
- 220 ohm resistor
- hook-up wires
- breadboard

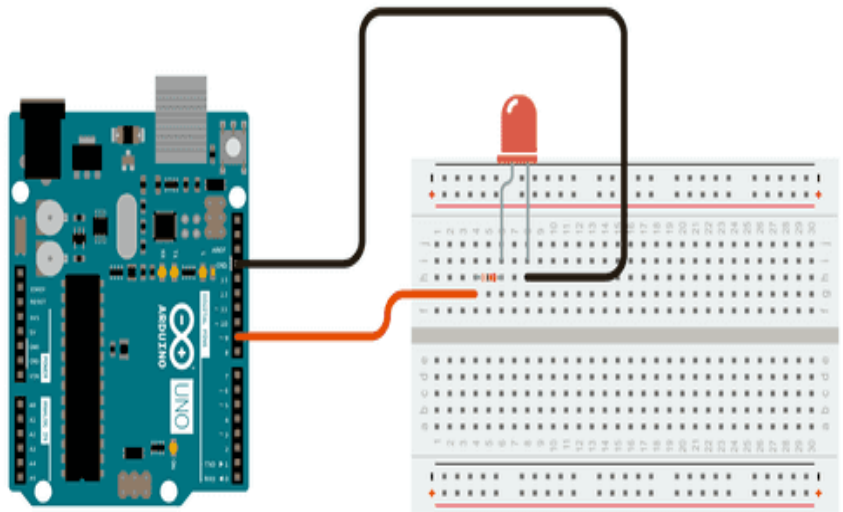
- **Circuit**
- Connect the **anode** (the longer, positive leg) of your LED to digital output pin 9 on your board through a 220 ohm resistor. Connect the **cathode** (the shorter, negative leg) directly to ground.
-

Steps to be followed:

1. After declaring pin 9 to be your ledPin, no need of setup() function.
2. analogWrite() function that you will be using in the main loop of your code requires two arguments
3. One telling the function which pin to write to, and
4. one indicating what PWM value to write.
5. In order to fade your LED off and on, gradually increase your PWM value from 0 (all the way off) to 255 (all the way on), and then back to 0 once again to complete the cycle.

```
int led = 9;        // the PWM pin the LED is attached to
int brightness = 0;  // how bright the LED is
int fadeAmount = 5;  // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```


5V
Azduino
UNO
13
12
11
10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
A0
A1
A2
A3
A4
A5
GND
220 Ω RESISTOR
LED


ARDUINO
UNO

Introduction to Raspberry pi

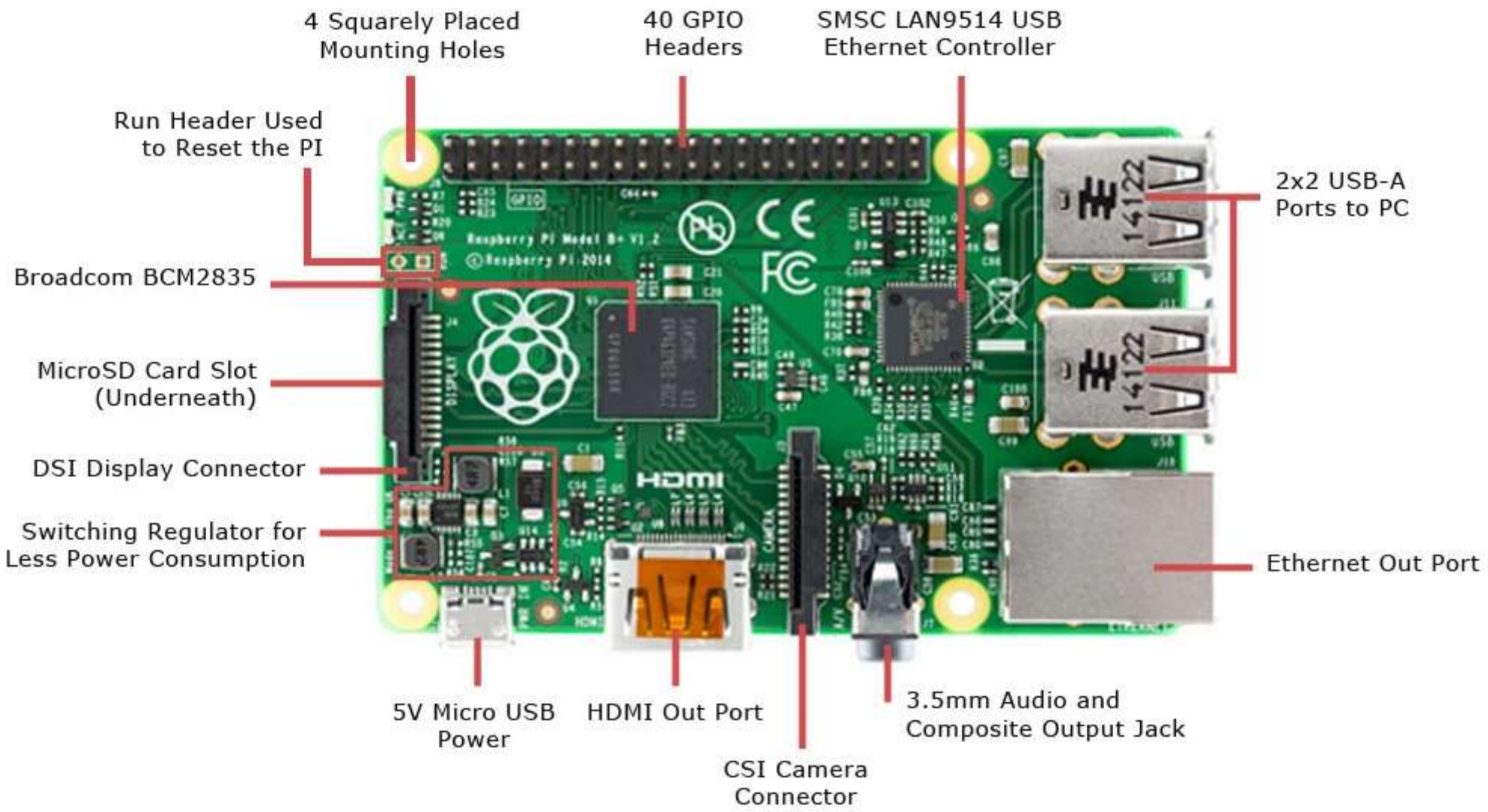
- Raspberry Pi is a small, affordable, and credit card-sized computer that was created with the goal of promoting basic computer science knowledge and programming skills among people of all ages.
- It was developed by the Raspberry Pi Foundation, a UK-based charity organization, and was first released in 2012. Since then, it has gained immense popularity worldwide due to its versatility and wide range of applications.
- The Raspberry Pi board is powered by an ARM processor and is equipped with various input and output ports, including USB, HDMI, audio, and General-Purpose Input/Output (GPIO) pins.
- It also has built-in networking capabilities, allowing it to connect to the internet via Ethernet or Wi-Fi.
- The Raspberry Pi can run on various operating systems, including Linux-based distributions such as Raspbian (now known as Raspberry Pi OS), as well as other third-party operating systems.

- **Since Raspberry Pi runs Linux operating system, it supports Python "out of the box".**
- One of the key features of the Raspberry Pi is its affordability.
- The board itself is reasonably priced, making it accessible to students, hobbyists, and professionals alike.
- This affordability, coupled with its small form factor and low power consumption, makes it ideal for a wide range of projects and applications.
- Its GPIO pins allow users to connect and control various external devices, such as sensors, motors, LEDs, and relays, enabling the creation of interactive and hardware-based projects.
- The Raspberry Pi Foundation provides extensive resources and support for users, including official documentation, tutorials, and a vast online community.
- Whether you're a beginner or an experienced developer, the Raspberry Pi offers endless possibilities for experimentation, learning, and innovation.

Raspberry Pi Pinout Diagram

GPIO: 30
5V : 02
3.3V: 02
GND: 06

- Main Pins:-
- GPIO (General Purpose IO): Raspberry Pi GPIO (General Purpose IO) describes the pins on a Raspberry Pi that can be used for digital input and output. The GPIO pins can be used to control devices such as LEDs, motors, and sensors.
- SPI (Serial Peripheral Interface): SPI is a full-duplex serial protocol, meaning data can be sent and received simultaneously. SPI is a full-duplex serial protocol that can be used for communication with flash memory, sensors, real-time clocks (RTCs), analogue-to-digital converters, and more.
- I2C (Inter-integrated Circuit): The I2C pin is a pin that can be used to communicate with I2C devices. The I2C bus can be used to connect multiple devices, such as sensors, EEPROMs, and other I2C devices.
- UART (Universal Asynchronous Receiver/Transmitter) :In computing, a universal asynchronous receiver/transmitter (UART /ˈjuːɑːrt/) is a computer hardware device for asynchronous serial communication in which the data format and transmission speeds are configurable.
- PCM (Pulse Code Modulation):he PCM interface is a digital interface for connecting audio devices. It is used to convert analog audio signals into digital audio signals.
- Ground
- 5v (Power): Raspberry Pi 5v power supply is a great way to get started with powering your own projects. The power supply provides 5 volts of power and is compatible with all versions of the Raspberry Pi.
- 3.3v (Power):All Raspberry Pi models since the B+ can provide up to 500mA on the 3v3 pins.


4 Squarely Placed Mounting Holes
40 GPIO Headers
SMSC LAN9514 USB Ethernet Controller
Run Header Used to Reset the PI
2x2 USB-A Ports to PC
Broadcom BCM2835
MicroSD Card Slot (Underneath)
DSI Display Connector
Switching Regulator for Less Power Consumption
Ethernet Out Port
5V Micro USB Power
HDMI Out Port
CSI Camera Connector
3.5mm Audio and Composite Output Jack

Raspberry Pi – Operating System

- Before you get started with your Raspberry Pi board, you need to provide with an OS (operating system). Linux is the most frequently used OS on the Raspberry Pi.
- For using an OS, we need to create a Secure Digital (SD) or MicroSD card with an OS on it. The prerequisite for setting up the SD or MicroSD is a computer having an internet connection and the ability to write to SD or MicroSD cards.

NOOBS Software

- NOOBS means new-out-of-box software and it is the easiest way to get started with the Raspberry Pi. It is easy to copy NOOBS to your SD or MicroSD card. Once copied, it provides us with a simple menu for installing various operating systems.
- There is an option to buy a card with NOOBS already installed on it, but it is always useful to know how to create your own NOOBS cards.

Download NOOBS

- Follow the below given steps to download NOOBS −
- Step 1 − Go to the website www.raspberrypi.org/downloads/noobs
- Step 2 − Select from the two versions of NOOBS available. Version 1 is the main version and includes Raspbian. This is the officially supported OS, which you can use even without any network connection.
- Another option is to choose the OS from the menu. You can download and install the OS from the menu, if you have a network connection. It is always recommended to download NOOBS for your first OS.

Raspberry Pi – Operating System

- Downloading and Installing Raspberry Pi OS
- Insert a microSD card / reader into your computer.
- Download and install the official Raspberry Pi Imager. ...
- Click Choose OS.
- Select Raspberry Pi OS (32-bit) from the OS menu (there are other choices, but for most uses, 32-bit is the best).

Connecting Raspberry Pi via SSH

- What is SSH
- Secure Shell (SSH) enables you to access the command line of a Raspberry Pi from another computer or device on the same network. This is very handy for quickly installing software or editing configuration files. SSH is pre-installed on Linux, Mac and some Windows operating systems and can also be installed on mobile devices. SSH does not provide any visual access to the Raspberry Pi Desktop.

To enable SSH (Secure Shell) on a Raspberry Pi, follow these steps:

1. Graphical desktop environment
2. Command line interface

- Boot up your Raspberry Pi and make sure it is connected to a network.
- If you have a graphical desktop environment running on your Raspberry Pi, you can enable SSH using the graphical interface.
  - Go to the Raspberry Pi Configuration tool by clicking on the Raspberry Pi icon in the top-left corner, selecting "Preferences,"
  - and then "Raspberry Pi Configuration."
  - In the "Interfaces" tab, click the checkbox next to "SSH" and click "OK."
  - You may need to enter your password to confirm the changes.

Connecting Raspberry Pi via SSH.

- If you are using the command line interface (CLI) on your Raspberry Pi, you can enable SSH by following these steps:
  - Open a terminal window.
  - Enter the following command to open the Raspberry Pi Configuration tool:
    sudo raspi-config
  - In the configuration tool, navigate to "Interfacing Options" and press Enter.
  - Select "SSH" and press Enter.
  - Choose "Enable" and press Enter.
  - Exit the configuration tool.

Raspberry Pi Interfaces(UART,SPI & I2C)

- UART (Serial):
  - Serial communication or the **UART (Universal Asynchronous Receiver / Transmitter)** pins provide a way to communicate between two microcontrollers or the computers.
  - **TX pin is used to transmit the serial data** and **RX pin is used to receive serial data coming from a different serial device.**
    - TX (GPIO14)
    - RX (GPIO15)
  - **SPI (Serial Peripheral Interface)** is another protocol used for **master-slave communication.**
  - It is used by the Raspberry pi board to quickly **communicate between one or more peripheral devices.**
  - Data is synchronized using a **clock (SCLK at GPIO11)** from the master (RPi) and the data is sent from the Pi to our **SPI device using the MOSI (Master Out Slave In) pin.**
  - If the SPI device needs to communicate back to **Raspberry Pi, then it will send data back using the MISO (Master In Slave Out) pin**
    - **SPI0:** GPIO9 (MISO), GPIO10 (MOSI), GPIO11 (SCLK), GPIO8 (CE0), GPIO7 (CE1)
    - **SPI1:** GPIO19 (MISO), GPIO20 (MOSI), GPIO21 (SCLK), GPIO18 (CE0), GPIO17 (CE1), GPIO16 (CE2)

Raspberry Pi Interfaces

I2C (Inter-Integrated Circuit)

I2C is used by the Raspberry Pi board to communicate with devices that are compatible with Inter-Integrated Circuit (a low-speed two-wire serial communication protocol).

- This communication standard requires master-slave roles between both the devices.
- I2C has two connections: SDA (Serial Data) and SCL (Serial Clock).
- They work by sending data to and using the SDA connection, and the speed of data transfer is controlled via the SCL pin.

- **Data**: (GPIO2), Clock (GPIO3)
- **EEPROM Data**: (GPIO0), EEPROM Clock (GPIO1)

Interfacing Hardware with the Raspberry Pi

- Interfacing hardware with a Raspberry Pi is a common task in various electronics and IoT (Internet of Things) projects. The Raspberry Pi is a versatile single-board computer with GPIO (General Purpose Input/Output) pins that can be used to connect and control a wide range of hardware components, such as sensors, LEDs, motors, and more. Here's a general guide on how to interface hardware with a Raspberry Pi:
- 1.Understand the GPIO Pins:Raspberry Pi models have different GPIO pin configurations. Consult the pinout diagram for your specific Raspberry Pi model. You can find this information in the official documentation or online resources.
- 2.Select Your Hardware Components:Decide what hardware components you want to interface with your Raspberry Pi. Common components include LEDs, buttons, sensors (e.g., temperature, humidity, motion), motors, displays, and more.
- 3.Power Considerations:Ensure that your hardware components are powered correctly. Some components may require external power sources, while others can be powered directly from the Raspberry Pi's GPIO pins. Be mindful of voltage and current requirements.
- 4.Connection Method:Connect your hardware components to the GPIO pins. You may need to use jumper wires, breadboards, or HATs (Hardware Attached on Top) depending on your project's complexity.

- 5.Install Necessary Libraries/Modules:Depending on the hardware component you're using, you may need to install Python libraries or modules to interface with them. Common libraries include RPi.GPIO for GPIO control and libraries specific to your sensors or actuators.
- 6.Write Code:Write Python scripts to control and interact with your hardware. You can use the GPIO library to control the GPIO pins, read sensor data, or drive motors. Import the necessary libraries and create functions to handle your hardware components.
- 7.Testing:Test your code and hardware connections incrementally. Make sure each component works as expected before moving on to more complex interactions.
- 8.Safety Considerations:Be cautious when interfacing with hardware to avoid damaging your Raspberry Pi or components. Avoid short circuits, reverse polarity, and excessive voltage/current.
- 9.Prototyping:Consider using a breadboard or a prototyping board to build and test your circuit before making permanent connections.
- 10.Documentation:Keep track of your hardware connections, pin assignments, and code. Good documentation is essential for troubleshooting and future reference.

- 11.Integration and Application:Once you've successfully interfaced your hardware, integrate it into your desired application or project. This may involve integrating sensors with software for data logging or creating a user interface for controlling hardware components.
- 12.Power Management:Depending on your project, you might need to implement power management strategies, such as using sleep modes or shutting down unused components, to optimize power consumption and prevent overheating.

Raspberry Pi Remote Access

- VNC (Virtual Network Computing):
  - Install and set up a VNC server on your Raspberry Pi. One popular option is RealVNC.
  - Ensure that your Raspberry Pi is connected to the network.
  - On your local computer, install a VNC viewer application. RealVNC provides VNC Viewer for various platforms.
  - Open the VNC viewer and enter the IP address or hostname of your Raspberry Pi.
  - Enter the username and password for the Raspberry Pi user account.
  - You should now have a graphical remote desktop connection to your Raspberry Pi.
- Both SSH and VNC provide different levels of remote access. SSH allows you to access the command line interface, while VNC provides a graphical interface for remote desktop access.
- Remember to secure your remote access by using strong passwords and, if necessary, configuring additional security measures such as key-based authentication or firewall settings.