IoT Module 4

CO4	**Illustrate the development of IoT applications with embedded computing boards.**		
M4.01	Illustrate the working of sensors and actuators	2	Understanding
M4.02	Outline the features of embedded computing boards - Arduino/Node MCU & Raspberry PI	2	Understanding
M4.03	*Illustrate the interfacing of basic sensors with embedded computing board - Arduino/Node MCU/ESP32	3	Understanding
M4.04	Recall the programming constructs in Python	3	Understanding
M4.05	*Illustrate the interfacing of basic sensors with embedded computing board - Raspberry PI	5	Understanding
M4.06	Summarize the applications building with IoT - Smart Perishable tracking/Smart transportation, Smart Healthcare, Smart Lavatory maintenance, Smart water through IoT, Smart warehouse monitoring, Smart Retail, Smart Driver assistance system	5	Understanding

Sensors and Actuators - Role of Sensors and Actuators in IoT - Working of Sensors and Actuators - Examples.
Embedded Computing Boards - features and characteristics of Arduino/Node MCU & Raspberry PI, Interfacing basic sensors with Computing boards
Python - data types, control structures, modules, packages, input/output.
Programming Raspberry Pi with python, Interfacing sensors and actuators with Raspberry PI.
Case Study - Applications building with IoT- Smart Perishable tracking/Smart transportation, Smart Healthcare, Smart Lavatory maintenance, Smart water through IoT, Smart warehouse monitoring, Smart Retail, Smart Driver assistance system.
*(sensors for applications like agricultural (temperature, humidity), pollution (gas/pollution), industrial (fire alarm),)

(Ref: Several Websites)

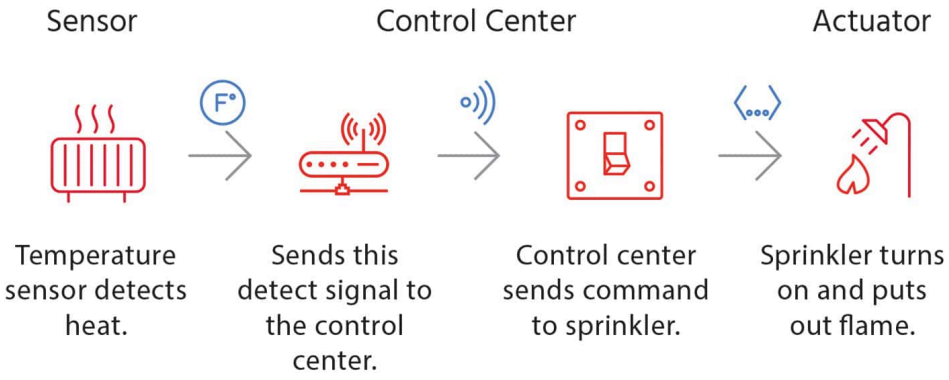
Sensors and Actuators

Role of Sensors and Actuators

An IoT device is made up of a Physical object (“thing”) + Controller (“brain”) + Sensors + Actuators + Networks (Internet). Sensors in the device sense the environment, then control signals are

generated for the actuators according to the actions needed to perform. An actuator is a machine component or system that moves or controls the mechanism of the system.

The following is the diagram of an automatic fire extinguishing system which shows the role of sensors and actuators in an IoT system.



The control system acts upon an environment through the actuator. It requires a source of energy and a control signal. When it receives a control signal, it converts the source of energy to a mechanical operation.

Sensors:

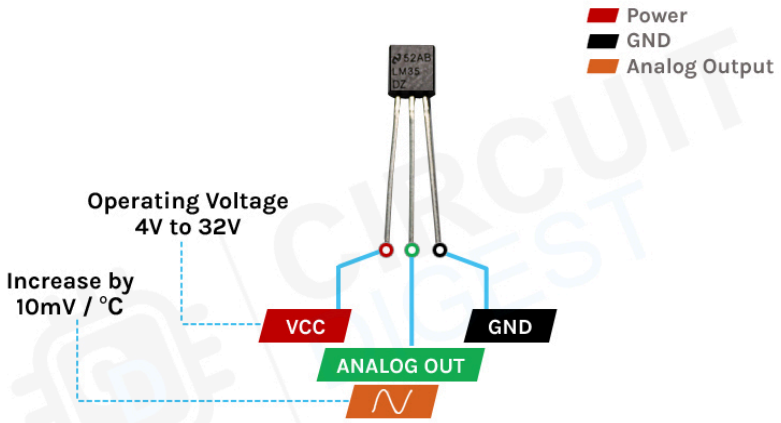
- IoT sensors are pieces of hardware that detect changes in an environment and collect data.
- They may detect things like temperature, pressure, movement, etc.
- If they are connected to a network, they share data with the network.
- There are two main types of sensors used in IoT applications:
    - Active and Passive Sensors
        - Passive sensors: detect changes in their environment without any dedicated power supply (e.g., some temperature sensors and IR sensors)
        - Active sensors: require some form of power source to function (eg: ultrasonic sensor)
    - Contact and Non-contact Sensors
        - Contact sensors require physical contact with the object/environment to measure the parameter. (eg: touch sensor)
        - Non-contact sensors do not require direct contact with the object/environment. (eg: optical sensor)

Types of sensors and their working:

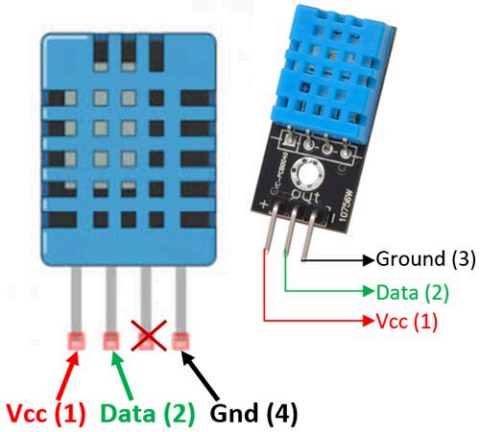
Temperature Sensors

- Temperature sensors measure the amount of heat generated from an area or an object.
- They detect a temperature change and convert it to data.
- The module uses a thermistor to detect the ambient temperature where the resistance of a thermistor will increase when the ambient temperature decreases.
- Temperature sensors are used in various industries, including manufacturing, healthcare, and agriculture.

- Some examples are thermistors, thermocouples, and resistor temperature detectors (RTD).
- An example is the LM35 temperature sensor which is inexpensive, accurate and easy-to-use. It has a range of −55°C to 150°C. It outputs data in analog format, making it very prone to external noise and interference.

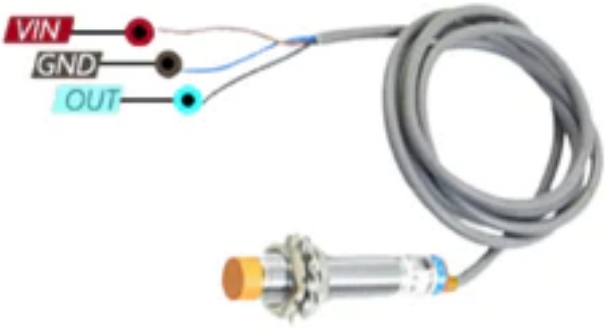


- The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and outputs a digital signal corresponding to both Temperature and Humidity through serial Data (no analog input pins needed). It needs a power supply of 3.5V to 5.5V. Its temperature is 0°C to 50°C and Humidity Range is 20% to 90%.



Proximity Sensors

- Proximity sensors detect the presence or absence of objects near the sensor without physical contact.
- They often emit a beam of radiation like infrared or an electromagnetic field to detect the presence.
- The sensor will output a HIGH if it detects the presence of an object, otherwise the output will be LOW.
- They can be used for process monitoring and control, object counting, assembly lines, and determining available space.
- Proximity sensors are common in retail settings, industrial complexes, and parking lots.

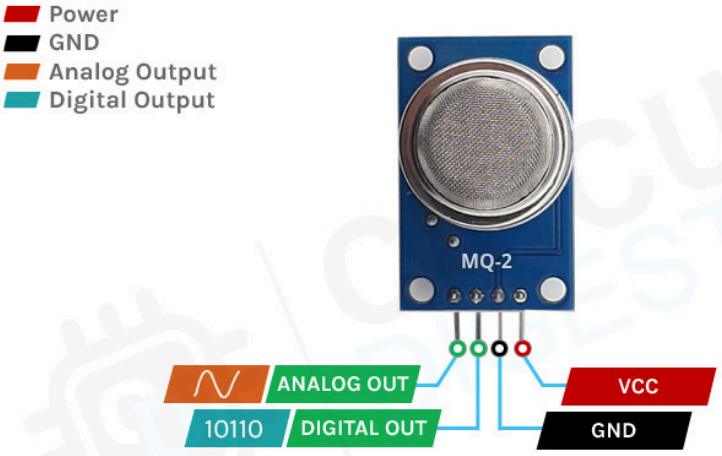


Pressure Sensors

- These sensors detect changes in a gas or liquid.
- When the pressure range is beyond a set threshold, pressure sensors alert the problem.
- They are used for leak testing, water systems, vehicles, and aircraft.
- For example, the BMP180 is a digital pressure sensor found in cell phones and GPS navigation devices.
- Some vehicles use a tire pressure monitoring system (TPMS) to alert when tire pressure is low and potentially unsafe.
- DPS368 is a high precision, digital barometric pressure sensor. This sensor can measure both temperature and pressure. This sensor is robust to water, dust, and humidity. This sensor is of very small size and is suitable for application in smartphones.

Gas Sensors

- These sensors monitor air quality for the presence of toxic or hazardous gas.
- They often use semiconductor, electrochemical, or photo-ionization technologies for detection.
- They are typically used in industrial and manufacturing firms.
- Some types are Carbon dioxide sensor, Carbon monoxide detector, Hydrogen sensor, Air pollution sensor, breathalyzer, etc.
- The MQ2 sensor can detect LPG, smoke, alcohol, propane, hydrogen, methane, and carbon monoxide concentrations in the air ranging from 200 to 10000 ppm (Parts-per-million).

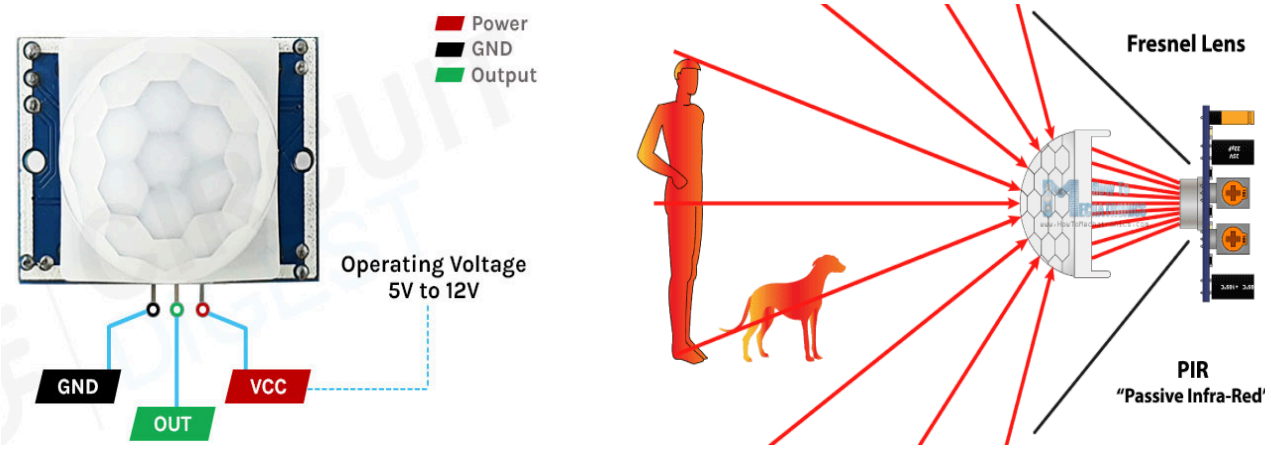


- The MQ2 gas sensor operates on 5V DC and consumes approximately 800mW.

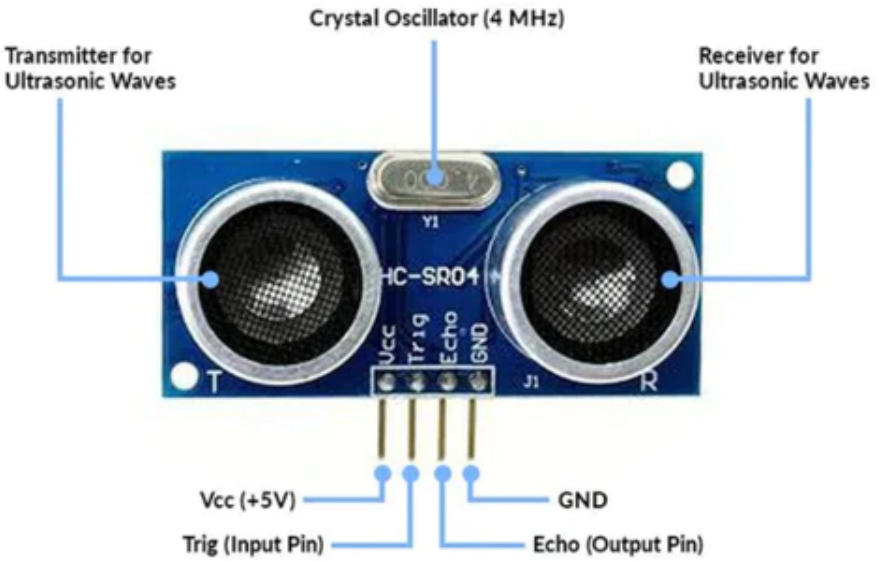
- Analog output pin of the board that will give us an analog signal which will vary between vcc and ground based on the gas level detected.
- The digital output goes HIGH if the sensor detects the presence of the gas in the atmosphere.
- The MQ-2 Gas sensor can be used for both *Detecting the gas* and also for *Measuring Butane and Hydrogen Gas level in PPM*.

Motion Sensors

- Motion sensors detect physical movement in an area.
- These sensors play a significant role in the security industry.
- Applications include automatic door controls, energy management systems, and automated parking systems.
- Standard motion sensors include passive infrared (PIR), ultrasonic and microwave.
- Passive Infrared (PIR) sensor: It Detects body heat (infrared energy) and gives a HIGH output. It is the most Commonly used sensor in IOT for home security.



- Ultrasonic sensor: Sends out pulses of ultrasonic waves and measures the reflection of a moving object by tracking the speed of sound waves.



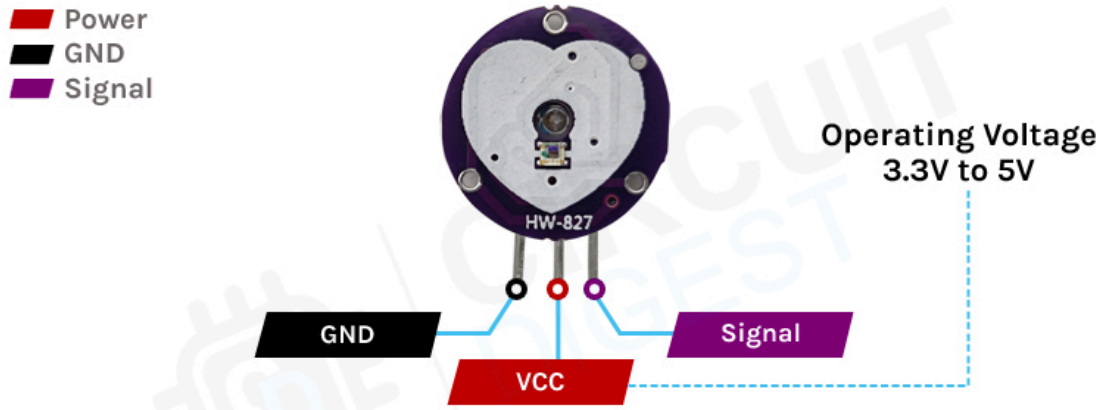
- Microwave: Sends out radio wave pulses and measures the reflection of a moving object.

Humidity Sensors

- These sensors measure the amount of water vapor in the air.
- Typical uses include heating and air conditioning systems and weather monitoring and prediction.
- When humidity must be tightly controlled, such as in museums, hospitals, and greenhouses, humidity sensors play a vital role.

Heartbeat Sensor

- The heartbeat sensor operates on the principle of detecting the subtle changes in blood flow through the skin caused by the expansion and contraction of blood vessels with each heartbeat.
- The sensor typically uses an optical or electrical method to capture these changes.



- In an optical sensor a light source, usually an LED, emits light that penetrates the skin and blood vessels. A photodetector then measures the amount of light that is absorbed or reflected back. As the blood vessels expand and contract with each heartbeat, the amount of light absorbed or reflected changes, allowing the sensor to calculate the heart rate.

Some other sensors are;

Infrared Sensor: It is a sensor that is used to sense specific characteristics of its surroundings by either emitting or detecting infrared radiation. It is also capable of measuring the heat being emitted by objects.

Accelerometer: An accelerometer is a transducer that measures the physical or measurable acceleration experienced by an object due to inertial forces and converts the mechanical motion into an electrical output. Eg: Smart phone.

Light Sensors: It measures the physical quantity of light rays and converts it into an electrical signal easily readable by the user or an electronic instrument/device.

Moisture sensors: These sensors are used in agriculture to monitor soil moisture levels, helping farmers make more informed irrigation and fertilizer application decisions.

Water quality sensor: They detect water quality and Ion monitoring, primarily in water distribution systems. Some types are pH Sensor, Conductivity Sensor, Chlorine Residual Sensor, etc.

Smoke sensor: It is a device that senses smoke (airborne particulates & gasses) and its level.

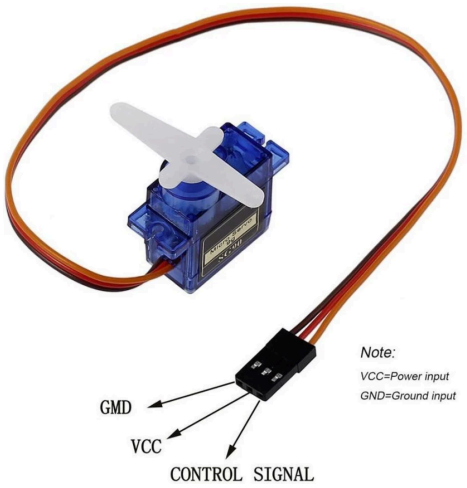
Image sensors: They convert optical images into electronic signals for displaying or storing files electronically. The primary use of image sensors is in digital cameras & modules, medical imaging and night vision equipment, thermal imaging devices, radar, sonar, media houses, and Biometric & IRIS devices.

Actuator

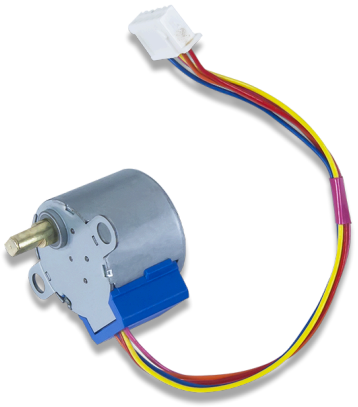
- An actuator is a part of a device or machine that helps it to achieve physical movements by converting energy, often electrical, into mechanical force.
- It is the component in any machine that enables movement.
- Actuators are essential to the operation of IoT devices because they enable the automation and control of physical systems and machinery.
- Actuators in IoT include motors, solenoids, hydraulic and pneumatic systems, and other devices that may control or manipulate physical things.

Examples:

- Electric Motors:
    - These motors turn electrical energy into mechanical motion, which powers IoT equipment.
    - Eg: Robots, drones, and home automation systems.
- Servo Motors:
    - A servo motor has a motor, a feedback sensor, and a control system. The feedback sensor monitors the position of the motor. It then gives feedback to the control system in real-time. This feedback loop guarantees that the motor moves to the exact correct position.
    - Eg: In Robotic systems, they allow for the precise and synchronized motions of robot limbs, grippers, and joints.



- Stepper Motors:
    - Stepper motors are intended for accurate position and rotation control. They can move in small increments, moving from one step to the next, unlike other types of motors.



- Solenoid Valves:
    - Converts electrical energy into linear motion. They are a wire coil coiled around a cylindrical core. When an electric current flows through the coil, a magnetic field is created that interacts with the core. This interaction causes the core to move linearly, either attracting or repelling it.
    - Eg: Door locks, fluid valves

Difference between sensors and actuators:

SENSOR	ACTUATOR
It converts physical characteristics into electrical signals.	It converts electrical signals into physical characteristics.
It takes input from the environment.	It takes input from the processing unit of the system.
It gives output to the input processing unit of the system.	It gives output to the environment.
Sensors generate electrical signals.	Actuator generates heat or motion.
It is placed at the input port of the system.	It is placed at the output port of the system.
It is used to measure the physical quantity.	It is used to measure the continuous and discrete process parameters.
It gives information to the system about the environment.	It accepts commands to perform a function.
Example: Photo-voltaic cell which converts light energy into electrical energy.	Example: Stepper motor where electrical energy drives the motor.

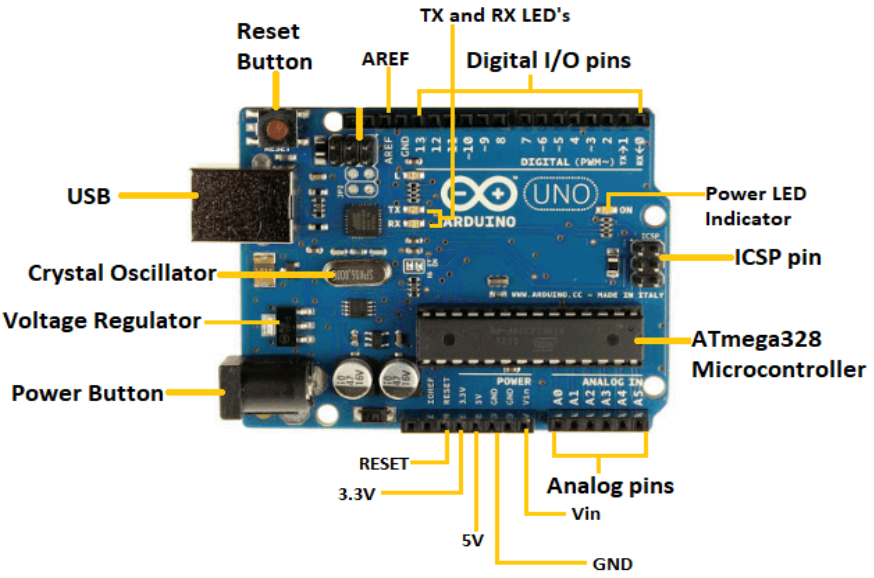
Embedded Computing Boards

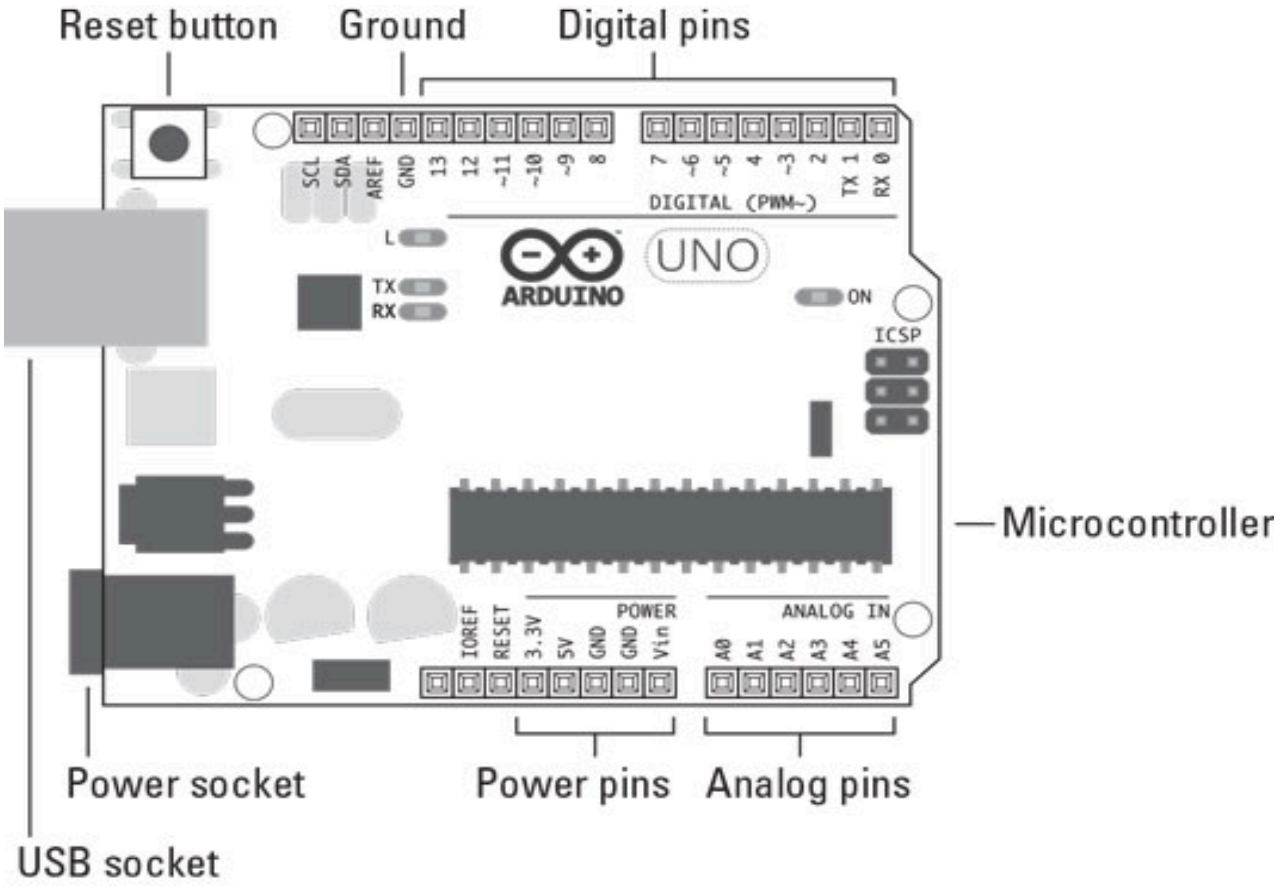
Arduino

- Arduino is an open hardware development board that can be used by tinkerers, hobbyists, and makers to design and build devices that interact with the real world.
- Arduino boards are able to read inputs (light on a sensor, a finger on a button, or a Twitter message, etc.) and turn it into an output (activating a motor, turning on an LED, publishing something online etc.)
- There are many models of Arduino boards such as Arduino Uno, Nano, Mega, Micro, etc.
- The programming is done using Arduino IDE.
- Features:
    - Inexpensive
    - Cross-platform
    - The simple, clear programming environment
    - Open source and extensible software and hardware.

Arduino UNO:

- Here UNO means 'one' in Italian. It is easy to use compared to other boards, such as the Arduino Mega board, etc. and hence it is most popular.
- The various components present on the Arduino boards are Microcontroller, Digital Input/output pins, USB Interface and Connector, Analog Pins, Reset Button, Power button, LED's, Crystal Oscillator, and Voltage Regulator.
- It has a 16 MHz ATmega328P microcontroller with 2KB SRAM, 32KB flash, 1KB of EEPROM, 0.5 KB of the Flash Memory is used by the bootloader code.
- The Arduino UNO includes 6 analog pin inputs, 14 I/O digital pins (out of which six capable of PWM output), a USB connector, a power jack, and an ICSP (In-Circuit Serial Programming) header.
- Its Operating Voltage is 5V but the supply voltage can be 6V to 20V.
- It is the most used standard form from the list of all available Arduino Boards.





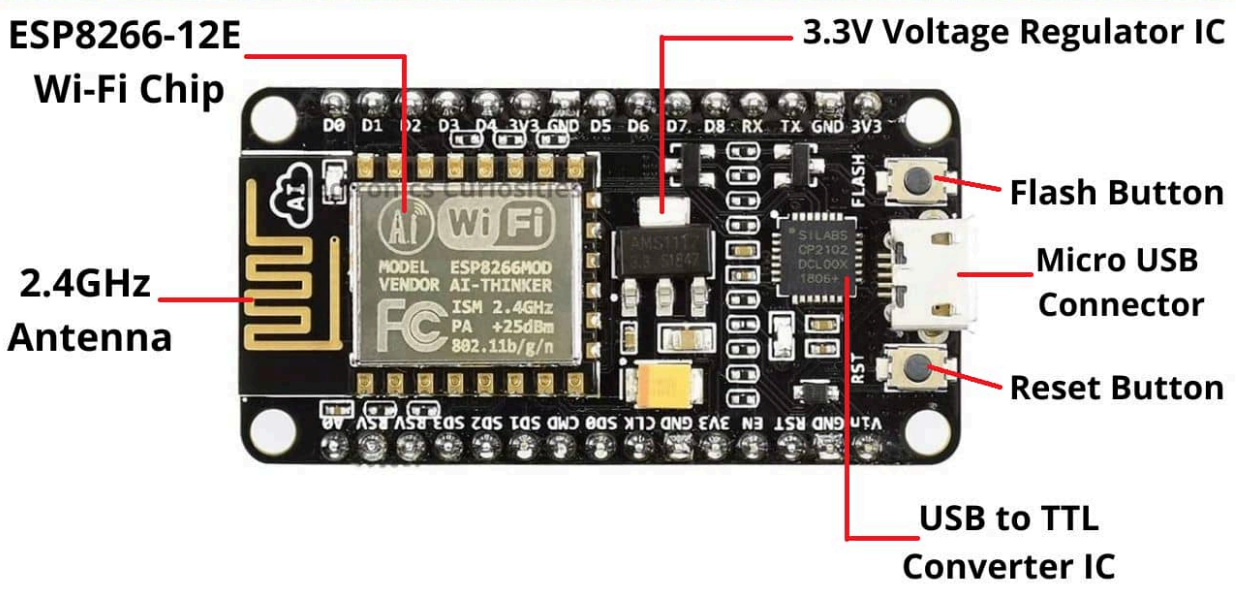
- **ATmega328P Microcontroller**- It is a single chip Microcontroller of the ATmel family. The processor core inside it is 8-bit with 2KB SRAM, 32KB flash, 1KB of EEPROM, 0.5 KB of the Flash Memory is used by the bootloader code. It combines Analog to Digital Converter, SPI serial ports, I/O lines, registers, timer, external and internal interrupts, and oscillator.
- **Digital I/O pins**- The digital pins can have the values HIGH or LOW. The pins numbered from D0 to D13 (or *0 to 13*) are digital pins. The pin13 is also connected to an on-board LED for testing purposes. *Arduino does not have Analog Output pins*. The analog outputs are performed through some special digital pins called the PWM Pins (3, 5, 6, 9, 10 and 11). These six pins provide an 8-bit PWM output by using analogWrite() function. The PWM pins can be identified by a ~ symbol just before the pin number on the board.
- **Analog Input Pins**- The six pins numbered from A0 to A5 are analog pins. The function of Analog pins is to *read* the analog sensor used in the connection. These can also act as GPIO (General Purpose Input Output) pins similar to the digital I/O pins. The analog pins can read the voltage from 0V to 5V (and all values in between).
- **GND**- Ground pins. The ground pin acts as a pin with zero voltage.
- **Vin**- It is the pin for input voltage for the board, if not already given through USB or Power socket.
- **USB Socket**- It allows the board to connect to the computer. It is essential for the programming of the Arduino UNO board. It can also give power to the board.

- **Power Socket** - To power up the board directly from the DC source.
- **3.3V and 5V pins**- Can be given to the components.
- **Crystal Oscillator**- The Crystal oscillator has a frequency of 16MHz, which makes the Arduino UNO a powerful board.
- **ICSP pin** - The In-Circuit Serial Programming pin allows the user to program using the firmware of the Arduino board.
- **Power LED Indicator**- The ON status of LED shows the power is activated. When the power is OFF, the LED will not light up.
- **TX and RX LEDs**- The successful flow of data is represented by the lighting of these LED's.
- **AREF**- The Analog Reference (AREF) pin is used to feed a reference voltage to the Arduino UNO board from the external power supply.
- **Reset button**- It resets the program on the Arduino or stops it completely when held down for a time. Connecting a wire between GND and the reset pin, which is located next to the 3.3V, achieves the same results.
- **Voltage Regulator**- The voltage regulator converts the input voltage to 5V.

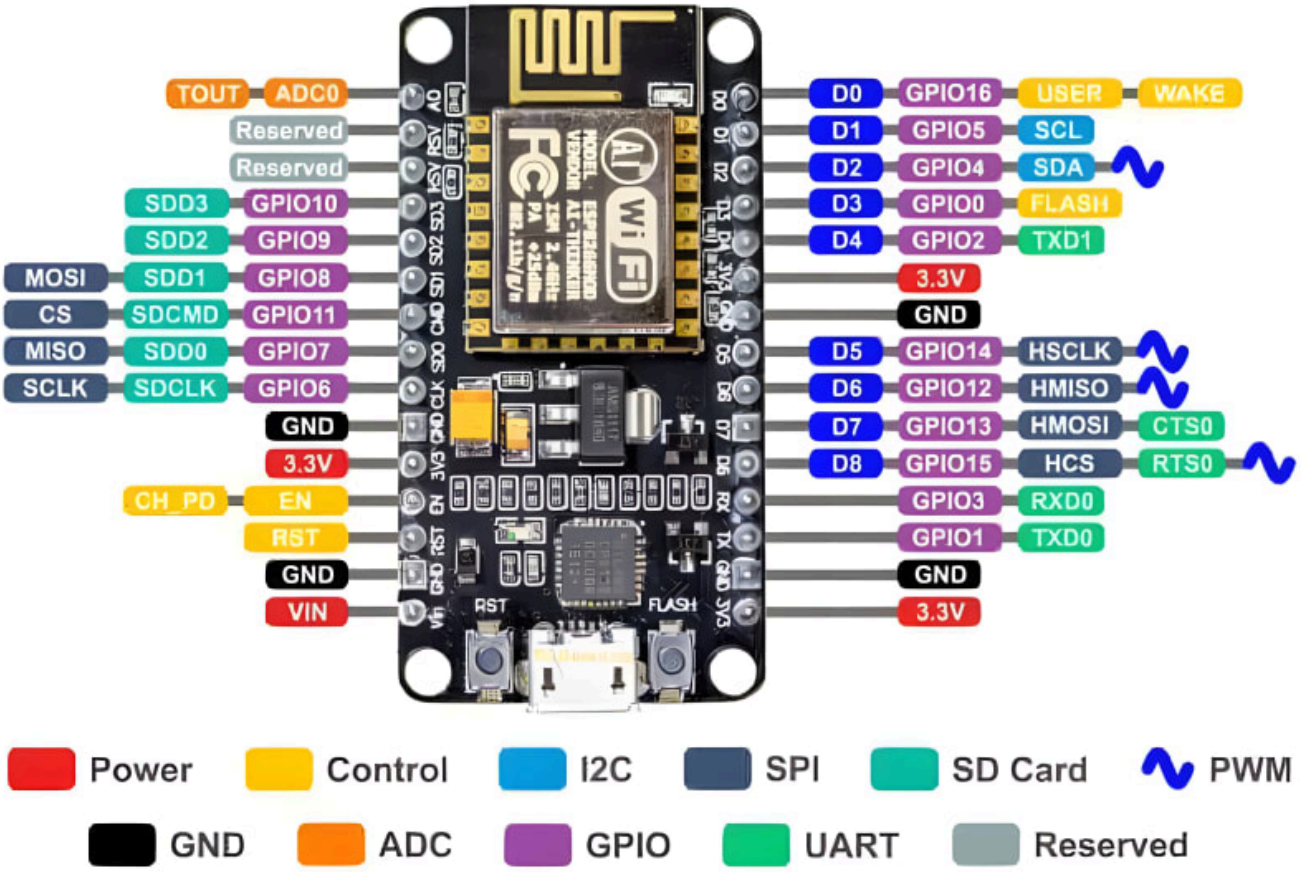
Node MCU

- NodeMCU is an open source firmware IoT platform for which open source prototyping board designs are available.
- The name "NodeMCU" combines "node" and "MCU" (micro-controller unit).
- The firmware uses the Lua scripting language.
- NodeMCU consists of a Microcontroller called the ESP8266. The ESP8266 is a low-cost Wi-Fi chip (System-on-a-Chip, *SoC*) developed by Espressif Systems with TCP/IP protocol.
- **The NodeMCU board has an ESP12 module which contains the ESP8266 chip.**

Features and Specifications



- ESP8266 Microcontroller Specifications
  - Microprocessor: Tensilica 32-bit RISC CPU Xtensa LX106
  - Operating Voltage: 3.3V (Can be fed through microUSB)
  - Input Voltage: 7-12V
  - Digital I/O Pins (DIO): 16
  - Analog Input Pins (ADC): 1
  - UARTs: 1
  - SPIs: 1
  - I2Cs: 1
  - Flash Memory: 4 MB
  - SRAM: 64 KB
  - Clock Speed: 80 MHz



- **Power Pins:**
  - There are four power pins. VIN pin and three 3.3V pins.
  - VIN can be used to directly supply the NodeMCU/ESP8266 and its peripherals. Power delivered on VIN is regulated through the onboard regulator on the NodeMCU module. You can also supply 5V regulated to the VIN pin.
  - 3.3V pins are the output of the onboard voltage regulator and can be used to supply power to external components.
- **GND:**
  - These are the ground pins of NodeMCU/ESP8266.

- **GPIO Pins:**
  - General-purpose input/output (GPIO) is a pin that can be either an input pin or output pin, whose behavior can be controlled at the run time. ESP8266 NodeMCU has 17 GPIO pins. Since several lines are used internally within the ESP8266 SoC, we have about 11 GPIO pins remaining for GPIO purposes. 2 pins out of 11 are generally reserved for RX and TX in order to communicate with a host PC from which compiled object code is downloaded. Hence finally, this leaves just 9 general-purpose I/O pins i.e. D0 to D8. The following table shows the mapping of pins.

D0	GPIO16
D1	GPIO5
D2	GPIO4
D3	GPIO0
D4	GPIO2
D5	GPIO14
D6	GPIO12
D7	GPIO13
D8	GPIO15

- **Analog (ADC) Pin:**
  - The NodeMCU is embedded with a 10-bit precision ADC and is labeled as A0. Thus the input analog values can range from 0 to 1023.
- **PWM Pins**
  - The Pulse Width Modulation (PWM) output can be implemented programmatically and used for driving digital motors and LEDs. To produce a PWM signal on a given pin you use the function analogWrite(pin, value); where pin can be 0 to 16 and value should be in range from 0 to 255. Commonly used PWM pins are D1 to D8.
- **I2C pins**
  - Inter-Integrated Circuit (I2C) is a serial bus interface connection protocol. It is also called TWI (two-wire interface) since it uses only two wires for communication. Those two wires are SDA (serial data) and SCL (serial clock). These two pins are used to connect all sorts of I2C sensors and peripherals in your project.
- **UART (Universal Asynchronous Receiver Transmitter) Pins**
  - NodeMCU based ESP8266 has two UART interfaces, UART0 and UART1. Since UART0 (RXD0 & TXD0) is used to upload firmware/codes to the board, we can't use them in applications while uploading firmware/codes.
- **SPI Pins (Serial Peripheral Interface)**
  - SPI is a full-duplex master-slave communication protocol. SPI enabled devices to work in two basic modes of SPI operation i.e. SPI Master Mode and SPI Slave Mode.

Raspberry PI

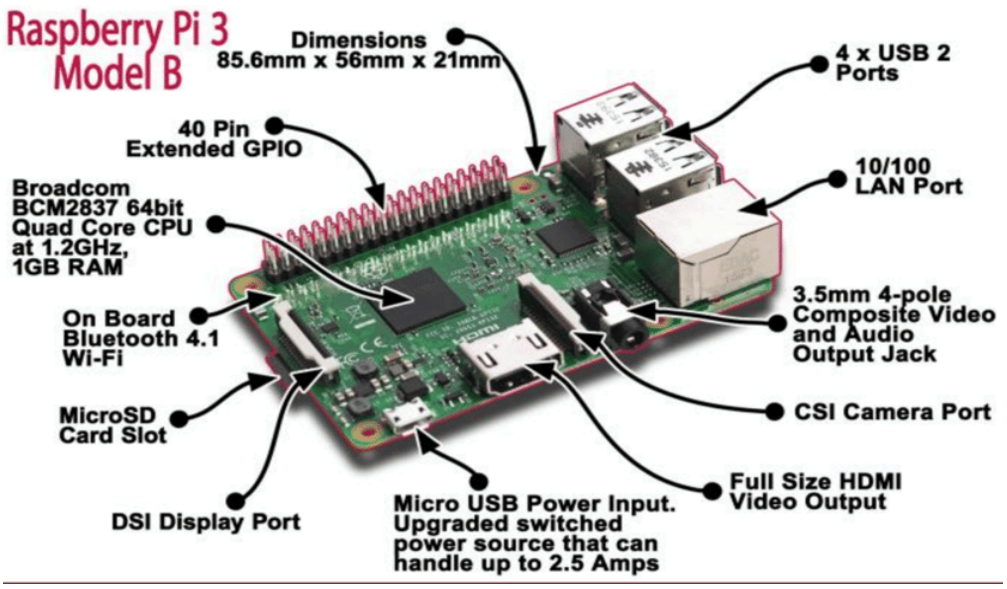
Raspberry Pi is a series of small single-board computers (SBCs) developed in the United Kingdom by the Raspberry Pi Foundation in association with Broadcom. By connecting peripherals like Keyboard, mouse, display to the Raspberry Pi, it will act as a mini personal computer.

It provides a set of GPIO (general purpose input/output) pins, allowing you to control electronic components for physical computing and explore the Internet of Things (IoT). Raspberry Pi is popularly used for real time Image/Video Processing, IoT based applications and Robotics applications. The Raspberry Pi operates in the open source ecosystem. It can run a variety of Linux OS, Windows IoT Core OS and its main supported official operating system, Raspberry Pi OS. Among these the Raspberry Pi OS is efficiently optimized to use with Raspberry Pi.

There have been many generations of the Raspberry Pi line: from Pi 1 to 4, and even a Pi 400. There has generally been a Model A and a Model B of most generations. Model A has been a less expensive variant, and tends to have reduced RAM and fewer ports (such as USB and Ethernet). The models are;

Pi 1 Model B (2012)
Pi 1 Model A (2013)
Pi 1 Model B+ (2014)
Pi 1 Model A+ (2014)
Pi 2 Model B (2015)
Pi Zero (2015)
Pi 3 Model B (2016)
Pi Zero W (2017)
Pi 3 Model B+ (2018)
Pi 3 Model A+ (2019)
Pi 4 Model A (2019)
Pi 4 Model B (2020)
Pi 400 (2021)

General Structure: Raspberry Pi Model B

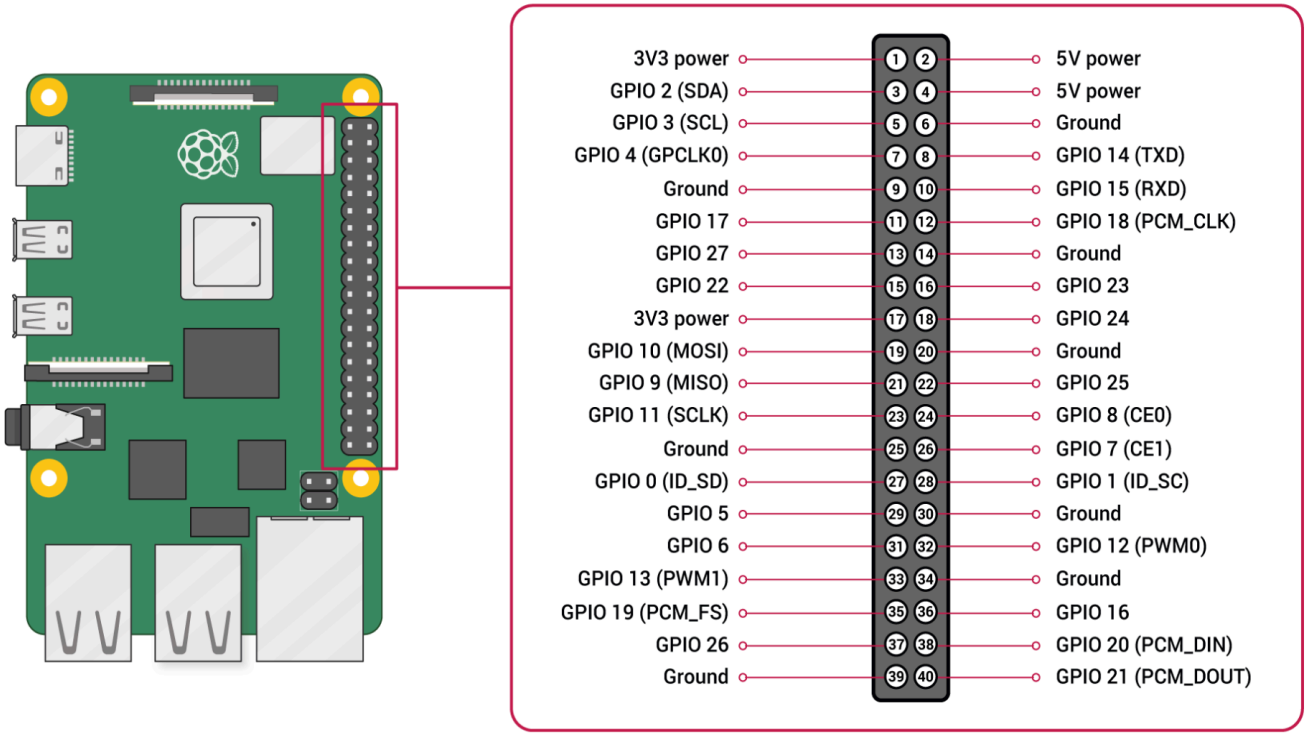


Components:

- **Central Processing Unit (CPU):**
  - ARM series Broadcom processors on its boards. All the models of the first generation make use of a 32bit, 700 MHz, single core chip that uses the ARM11 architecture.
  - Raspberry Pi 4 has a quad-core 64-bit processor. The clock frequency is also increased to 1.8 GHz.
- **Memory (RAM)**:
  - The initial Raspberry Pi had 256 MB RAM. Over the years, developers gradually and significantly improved the size. Now the Raspberry Pi 4 with 8GB RAM is available.
- **Graphic Processing Unit (GPU)**:
  - Its primary purpose is to increase the speed of image calculations and is located inside the main chip.
- **HDMI port:**
  - An HDMI interface (and an RCA port in some models) for display purposes.
- **Audio Port:**
  - The Pi Model B+, Pi 2, Pi 3 and Pi 4 features a 4-pole 3.5mm audio jack which also includes the composite video signal.
- **Camera Module Port:**
  - All current models of Raspberry Pi have a port for connecting the Camera Module.
  - It is called the Camera Serial Interface(CSI) port.
- **Ethernet port:**
  - The Ethernet port is a connectivity hardware feature available on B models of Raspberry Pi.
  - Pi 4 supports Gigabit Ethernet.
- **SD card slot**:
  - The Raspberry Pi board has a Secure Digital card or SD card slot where users must insert SD cards for the computer to function.
  - The SD card functions like a hard drive as it contains the operating system necessary for turning the system on.
  - It also serves to store data.
- **USB ports**:
  - They allow the computer to connect to a keyboard, mouse, hard drives, etc.
  - The newer models come with USB 2.0 and USB 3.0 ports.
- **Wifi and Bluetooth:**
  - Supports 2.4/5 GHz dual-band Wi-Fi.
  - The newer version of Raspberry Pi supports Bluetooth 5 and BLE connectivity.
- **Power source**:
  - Raspberry Pi has a power source connector that typically uses a 5V micro USB/Type C power cable.
  - The amount of electricity any Raspberry Pi consumes depends on what it’s used for and the number of peripheral hardware devices connected.
- **LEDs:**
  - These are a group of five light-emitting diodes. They signal the user on the present status of the Raspberry Pi unit. Their function covers:

- PWR (Red): When the unit is on, it emits a red light and only goes off when the unit is switched off.
- ACT (Green): This flashes to indicate any form of SD card activity.
- LNK (Orange): LNK LED gives off an orange light to signify that active Ethernet connectivity has been established.
- FDX (Orange): FDX light also comes during Ethernet connection. It shows that the connection is a full-duplex.

- **General Purpose Input and Output (GPIO) pins**:
  - Most Raspberry Pi models have 40 GPIO pins. These pins are used to interact, control and monitor the outside world by being connected to electronic circuits. They can read and control the electric signals from other boards or devices based on how the user programs them. Of the 40 pins, 26 are GPIO pins and the rest are power, ground, or others. A GPIO pin designated as an **output** pin can be set to high (3.3V) or low (0V). A GPIO pin that is designated as an **input** will allow a signal to be received by the Raspberry Pi. A voltage between 1.8V and 3.3V will be read by the Raspberry Pi as **high**; anything lower than 1.8V will be read as **low**. Raspberry Pi does not have analog input pins. We need to use an Analog to Digital Converter separately for interfacing analog sensors.



Interfacing Basic Sensors with Arduino Uno

Basic Materials needed:

1. Arduino Uno
2. USB connector to connect Arduino with PC
3. A Computer with Arduino IDE installed.

4. Breadboard
5. LEDs and current limiting resistors for testing purposes.
6. Connecting wires

Also we need the sensors as per requirement.

Steps for interfacing

1. Choose the I/O pin to be used for the sensor. The selection must be based on whether the input/output is digital or analog. For digital I/O the pins D0 to D13 can be used. For Analog input, pins A0 to A5 can be used. For analog output, the PWM pins 3, 5, 6, 9, 10, 11 can be used.
2. Connect the sensor to the pin/pins.
3. If the sensor needs power supply (5V or 3.3V), it can be given from the board itself. Ground connection is also there on the board.
4. Connect the Arduino board to the PC using the USB cable.
5. Open the Arduino IDE and select the board and port.
6. Do the Programming. It can be done using C++ language in the IDE.
7. The coding screen is divided into two blocks: the *setup()* function and the *loop()* function. The setup is considered as the preparation block, while the loop is considered as the execution block. The code which needs to be executed while the board is booted up (eg: initializing the parameters/variables) is included in the setup() function. The code which is to be run always is included in the loop() function.
8. After the code is written, compile it and upload it into the Arduino and check the working.

Interfacing sensors with NodeMCU are similar, but the pin names will be different.

Coding Basics:

The setup is considered as the preparation block, while the loop is considered as the execution block.

```
void setup( )
{
  // statements to be executed once during startup
}
void loop( )
{
  // statements to be executed continuously as loop
}
```

In the setup function, the pin mode can be set as INPUT or OUTPUT, and the serial monitor can be started. The functions that can be helpful here are given below.

- **pinMode( )**: To set the specific *pin* number is to the INPUT or OUTPUT *mode*.
    - Syntax: **pinMode (pin, mode)**
    - Eg: pinMode (12, OUTPUT);

The void loop () would include digitalWrite( ) and delay ( ) as the main functions.

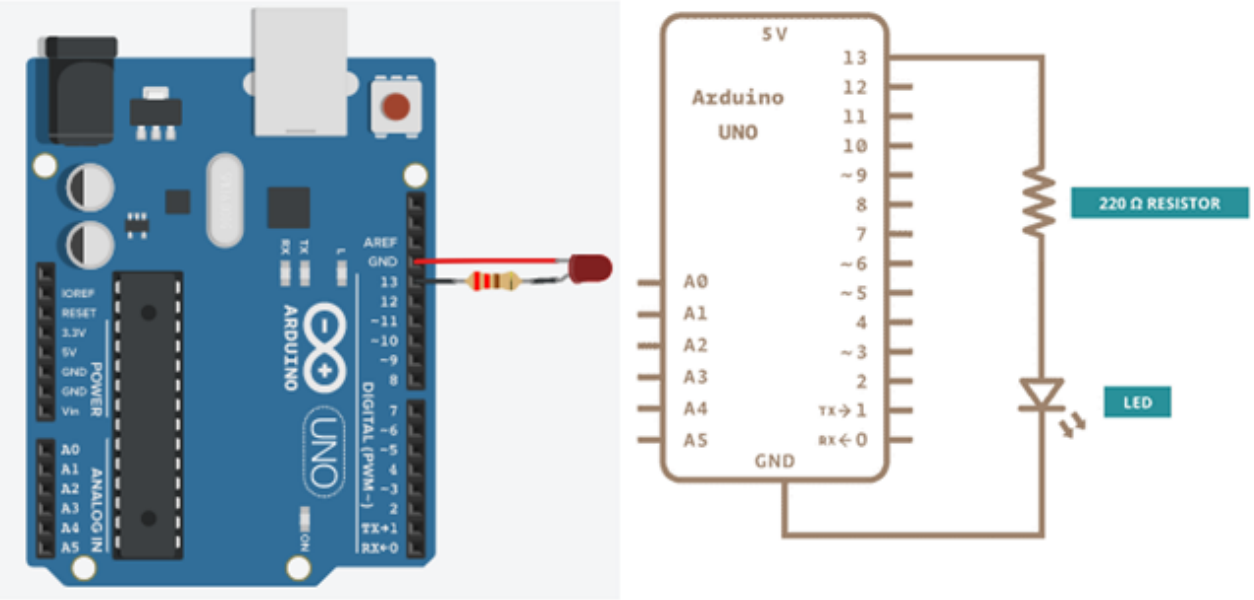
- **digitalWrite( )**: To set the value of a *pin* as HIGH or LOW.

- Syntax: **digitalWrite(*pin*, value HIGH/LOW)**
- Eg: digitalWrite (13, HIGH);

- **digitalRead( ):** To read digital data (HIGH/LOW) from a *pin*.
  - Syntax: **digitalRead(pin);**
  - Eg: digitalRead(13);
- **delay ( ):** To pause a program from doing a task during the specified duration in milliseconds.
  - Syntax: **delay(n);** where n is the time in milliseconds
  - Eg: delay (2000);
- **analogRead( )**: To read the value from the specified analog *pin*. It will map input voltages between 0 and the operating voltage (5V or 3.3V) into integer values between 0 and 1023.
  - Syntax: **analogRead(pin);**
  - Eg: analogRead(A0);
- **analogWrite( )**: To write an analog *value* (PWM wave) to a digital *pin*. The *value* should be between 0 and 255.
  - Syntax: **analogWrite(pin, value);**
  - Eg: analogWrite(11, 200);

(For full function reference: https://www.arduino.cc/reference/en/)

LED Blinking Program using Arduino UNO




```
int LEDpin = 13;
void setup() {
  // put your setup code here, to run once:
pinMode(LEDpin, OUTPUT);
}
void loop() {
  // put your main code here, to run repeatedly:
digitalWrite(LEDpin, HIGH); // turn on the LED
```

```
delay(1000); // delay for 1 sec
digitalWrite(LEDpin, LOW); // turn off the LED
delay(1000); // delay for 1 sec
}
```

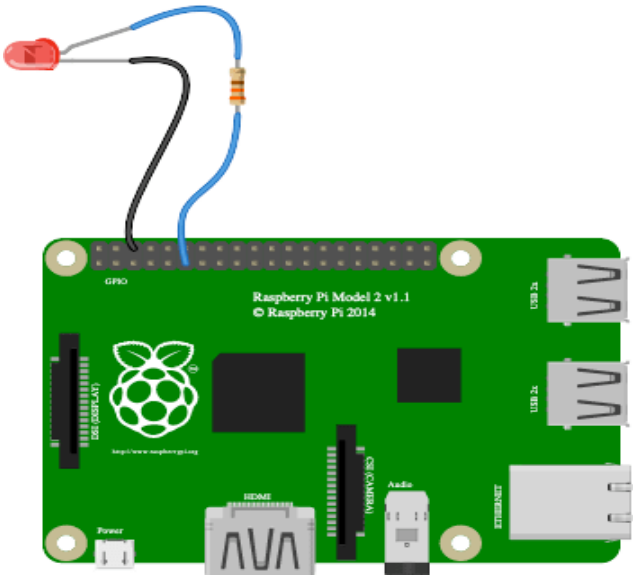
Programming Raspberry Pi with python

The General Purpose Input/Output (GPIO) pins on a Raspberry Pi are one of its most powerful features, allowing you to interact with the physical world. GPIO pins can be used to control external devices, such as LEDs, motors, sensors, and much more. They can also receive input from various sensors, buttons, and switches. The most commonly used models, such as Raspberry Pi 3 and Raspberry Pi 4, have 40 GPIO pins. Make sure to check the pinout diagram specific to the Raspberry Pi model. GPIO pins have physical numbering on the Raspberry Pi board. GPIO pins can be configured as either input or output pins. Input pins are used to read the state of external devices, while output pins are used to send signals to control those devices.

Python is the most popular programming language for Raspberry Pi projects, and it provides excellent support for GPIO control. The official Raspberry Pi Foundation provides a Python library called **RPi.GPIO**, which simplifies GPIO programming. There are some other libraries such as GPIO Zero (good for beginners) and WiringPi (good for experts).

LED Blinking Program using RPi

1. We need an LED, a 100 ohm resistor, and connecting wires.
2. Connect the long leg (anode) of the LED to a GPIO pin of the Raspberry Pi (say GPIO 17, i.e. pin 11 on the board).
3. Connect the short leg (cathode) of the LED to the ground pin of RPi through a 100 ohm resistor in between for limiting the current.
4. Open any text editor in RPi and save it as a .py file (say program.py). In this file, type the following program. There are many ways to program. Two ways are given below.



By Using RPi.GPIO Library:

```
import RPi.GPIO as GPIO          # Import Raspberry Pi GPIO library
from time import sleep           # Import the sleep function

pinLED = 17                       # LED GPIO Pin

GPIO.setmode(GPIO.BCM)           # Use GPIO pin number
GPIO.setwarnings(False)          # Ignore warnings in our case
GPIO.setup(pinLED, GPIO.OUT)     # GPIO pin as output pin

while True:                           # Endless Loop
   GPIO.output(pinLED, GPIO.HIGH)    # Turn on
   print(LED on)                     # Prints state to console
   sleep(1)                          # Pause 1 second
   GPIO.output(pinLED, GPIO.LOW)     # Turn off
   print(LED off)                    # Prints state to console
   sleep(1)                          # Pause 1 second
```

By Using gpiozero Library:

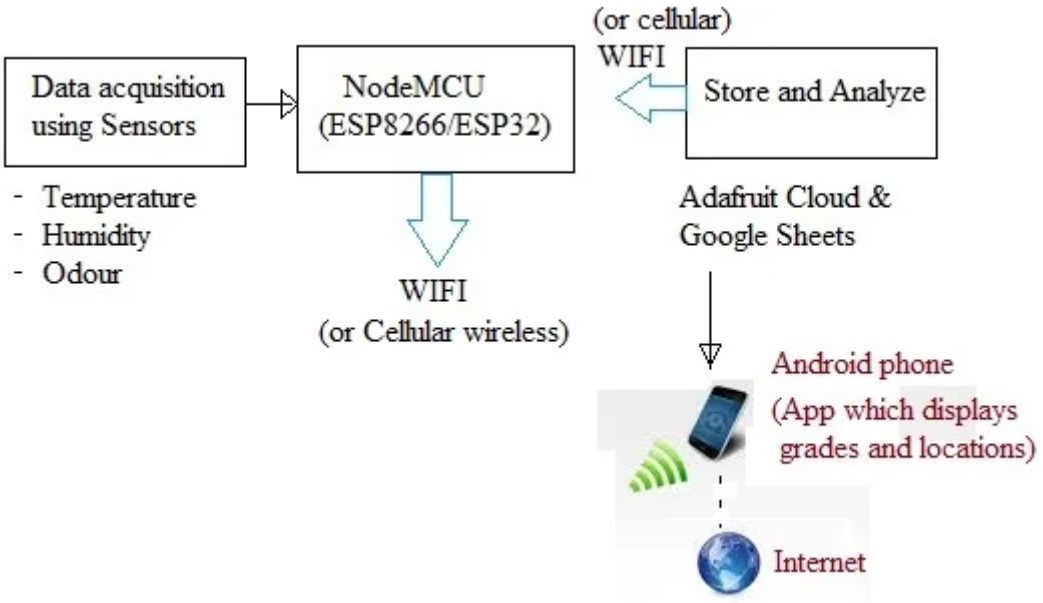
```
from gpiozero import LED
from time import sleep

led = LED(17)

while True:
    led.on()
    sleep(1)
    led.off()
    sleep(1)
```

Here LED(17), LED("GPIO17"), LED("BCM17") and LED("BOARD11") are the same.

Save the program.
Go to Terminal and execute the program by *python program.py*

Case Study - Applications building with IoT

Smart Perishable Tracking
(https://www.rfwireless-world.com/Articles/IoT-based-Perishable-Food-and-Vegetables-Tracking-System-Architecture.html)



Here are three main parts of the system viz. data acquisition, data storage and analysis and grade data monitoring. The heart of the system is a microcontroller based board ESP32. There are various other boards which can also be used.

Data Acquisition: The data acquisition from food and vegetables boxes have been carried out using sensors such as temperature, humidity and odor. The data from these sensors are passed to MCU through the ADC (analog pin). The data is read by MCU and passed to the cloud server for storage.

Cloud data Storage and Analysis: The popular cloud data storage providers include Google cloud platform, Microsoft Azure, AWS from Amazon, Adafruit cloud, etc. The dashboards in adafruit help in visualizing sensor data. The data is transferred to google sheets for further data analysis. Google sheets compare real time data with optimum thresholds set for temperature, humidity and odor. Based on this, each box is assigned a grade in sheets.

Data monitoring: Then the data is moved from google sheets to applications running on android mobile phones. There should be internet connectivity in the mobile phone to monitor the data in real time. The grade and delivery location of each of the boxes are monitored before they get spoiled.

Smart Transportation
Smart transportation uses new and emerging technologies such as IoT devices and 5G communication technology to make moving around a city more convenient, more cost effective (for both the city and the individual), and safer. The former provides for inexpensive sensors and controllers that can be embedded into nearly any physical machine to be controlled and managed remotely. The latter provides the high speed communications needed for managing and controlling transportation systems in real time with minimal latency. IoT provides drivers in a smart city with many benefits, including

traffic management, improved logistics, efficient parking systems, and enhanced safety measures. Smart transportation is the integration of all these benefits into applications for transportation systems.

Assignment 3:

Case Study - Applications building with IoT-, Smart Healthcare, Smart Lavatory maintenance, Smart water through IoT, Smart warehouse monitoring, Smart Retail, Smart Driver assistance system.