What is the Internet of Things (IoT)?

The term IoT, or Internet of Things, refers to the collective network of connected devices and the technology that facilitates communication between devices and the cloud, as well as between the devices themselves. Thanks to the advent of inexpensive computer chips and high bandwidth telecommunication, we now have billions of devices connected to the internet. This means everyday devices like toothbrushes, vacuums, cars, and machines can use sensors to collect data and respond intelligently to users.

The Internet of Things integrates everyday "things" with the internet. Computer Engineers have been adding sensors and processors to everyday objects since the 90s. However, progress was initially slow because the chips were big and bulky. Low power computer chips called RFID tags were first used to track expensive equipment. As computing devices shrank in size, these chips also became smaller, faster, and smarter over time.

The cost of integrating computing power into small objects has now dropped considerably. For example, you can add connectivity with Alexa voice services capabilities to MCUs with less than 1MB embedded RAM, such as for light switches. A whole industry has sprung up with a focus on filling our homes, businesses, and offices with IoT devices. These smart objects can automatically transmit data to and from the Internet. All these "invisible computing devices" and the technology associated with them are collectively referred to as the Internet of Things.

How does IoT work?

A typical IoT system works through the real-time collection and exchange of data. An IoT system has three components:

Smart devices

This is a device, like a television, security camera, or exercise equipment that has been given computing capabilities. It collects data from its environment, user inputs, or usage patterns and communicates data over the internet to and from its IoT application.

IoT application

An IoT application is a collection of services and software that integrates data received from various IoT devices. It uses machine learning or artificial intelligence (AI) technology to analyze this data and make informed decisions. These decisions are communicated back to the IoT device and the IoT device then responds intelligently to inputs.

A graphical user interface

The IoT device or fleet of devices can be managed through a graphical user interface. Common examples include a mobile application or website that can be used to register and control smart devices.

What are examples of IoT devices?

Let's look at some examples of IoT systems in use today:

Connected cars

There are many ways vehicles, such as cars, can be connected to the internet. It can be through smart dashcams, infotainment systems, or even the vehicle's connected gateway. They collect data from the accelerator, brakes, speedometer, odometer, wheels, and fuel tanks to monitor both driver performance and vehicle health. Connected cars have a range of uses:

- Monitoring rental car fleets to increase fuel efficiency and reduce costs.
- Helping parents track the driving behavior of their children.
- Notifying friends and family automatically in case of a car crash.
- Predicting and preventing vehicle maintenance needs.

Connected homes

Smart home devices are mainly focused on improving the efficiency and safety of the house, as well as improving home networking. Devices like smart outlets monitor electricity usage and smart thermostats provide better temperature control. Hydroponic systems can use IoT sensors to manage the garden while IoT smoke detectors can detect tobacco smoke. Home security systems like door locks, security cameras, and water leak detectors can detect and prevent threats, and send alerts to homeowners.

Connected devices for the home can be used for:

- Automatically turning off devices not being used.
- Rental property management and maintenance.
- Finding misplaced items like keys or wallets.
- Automating daily tasks like vacuuming, making coffee, etc.

Smart cities

IoT applications have made urban planning and infrastructure maintenance more efficient. Governments are using IoT applications to tackle problems in infrastructure, health, and the environment. IoT applications can be used for:

- Measuring air quality and radiation levels.
- Reducing energy bills with smart lighting systems.
- Detecting maintenance needs for critical infrastructures such as streets, bridges, and pipelines.
- Increasing profits through efficient parking management.

Smart buildings

Buildings such as college campuses and commercial buildings use IoT applications to drive greater operational efficiencies. IoT devices can be use in smart buildings for:

- Reducing energy consumption.
- Lowering maintenance costs.
- Utilizing work spaces more efficiently.

What are IoT technologies?

Technologies used in IoT systems may include:

Edge computing

Edge computing refers to the technology used to make smart devices do more than just send or receive data to their IoT platform. It increases the computing power at the edges of an IoT network, reducing communication latency and improving response time.

Cloud computing

Cloud technology is used for remote data storage and IoT device management – making the data accessible to multiple devices in the network.

Machine learning

Machine learning refers to the software and algorithms used to process data and make real-time decisions based on that data. These machine learning algorithms can be deployed in the cloud or at the edge.

1) BLE

Bluetooth Low Energy (BLE) is a wireless communication technology designed for short-range communication between devices. It is a power-efficient version of the classic Bluetooth technology and is also known as Bluetooth Smart. BLE was introduced with Bluetooth 4.0 and is a significant enhancement over traditional Bluetooth, especially in terms of energy consumption.

How BLE Works:

1. Advertising:BLE devices operate in two modes: advertiser and scanner/peripheral. Advertisers broadcast packets containing information about themselves, such as their identity or available services. Scanners listen for these advertising packets.

2. Connection Establishment:When a scanner or peripheral decides to connect to an advertiser, a connection is established. This connection is short-lived and is initiated only when data needs to be transferred.

3. GATT (Generic Attribute Profile): BLE uses the Generic Attribute Profile for organizing and describing the data that devices exchange. It defines roles for devices, such as server (providing data) and client (consuming data), and a hierarchical structure for organizing data into services, characteristics, and descriptors.

4. Characteristic Value Notification/Indication:BLE uses characteristic value notification or indication mechanisms to transfer data between devices. Notifications are unacknowledged, while indications require an acknowledgment from the receiving device.

5. Security: BLE incorporates security measures, including encryption and authentication, to ensure secure communication between devices.

Uses of BLE:

1. IoT Devices: BLE is widely used in Internet of Things (IoT) devices due to its low power consumption. Devices like smart home sensors, fitness trackers, and health monitors often use BLE to communicate with smartphones or other gateways.

2. Wearable Devices: Many wearable devices, such as smartwatches and fitness trackers, use BLE for communication with smartphones. The low energy consumption of BLE is crucial for these battery-powered devices.

3. Beacon Technology: BLE beacons are small, low-cost devices that broadcast information to nearby smartphones or other BLE-enabled devices. They are commonly used for location-based services, indoor navigation, and proximity marketing.

4. Healthcare Devices: BLE is utilized in various healthcare applications, including medical sensors, remote patient monitoring, and communication between medical devices and smartphones.

5. Asset Tracking: BLE is employed in asset tracking systems, allowing businesses to monitor and manage the location of their assets in real-time.

6. Home Automation: BLE is used in smart home devices, enabling communication between smart bulbs, thermostats, and other home automation devices with central control systems like smartphones or smart hubs.

In summary, Bluetooth Low Energy is a versatile technology that excels in applications where low power consumption is crucial, making it a key enabler for a wide range of wireless communication scenarios, especially in the context of IoT and wearable devices.

Radio-Frequency Identification (RFID)

Radio-Frequency Identification (RFID) is a technology that uses radio waves to wirelessly identify and track objects. It typically consists of two main components: an RFID tag and an RFID reader. The tag contains information about the item to which it is attached, and the reader uses radio waves to retrieve that information.

How RFID Works:

1.RFID Tags: RFID tags come in various forms, but they generally consist of a microchip and an antenna. The microchip stores data, and the antenna allows the tag to communicate with RFID readers. Tags can be active (with a built-in power source) or passive (powered by the energy from the reader).

2. RFID Readers: Readers emit radio waves in the form of electromagnetic fields. When an RFID tag passes through this field, it receives power from it and sends the stored data back to the reader.

3. Frequency Bands: RFID systems operate in different frequency bands, including low-frequency (LF), high-frequency (HF), and ultra-high-frequency (UHF). The choice of frequency depends on factors such as the application, the required read range, and environmental considerations.

4. Read Range: The read range of an RFID system varies depending on factors such as the frequency used, the power of the reader, and the type of RFID tags. LF systems typically have shorter read ranges, while UHF systems can achieve longer distances.

5. Data Storage: RFID tags can store varying amounts of data, from a unique identifier to more extensive information about the tagged item. The data on the tag can be read, modified, or updated by an RFID reader.

Uses of RFID:

1. Supply Chain Management: RFID is widely used in supply chain management for tracking and managing the movement of goods. It helps improve inventory accuracy, reduce errors, and streamline logistics processes.

2. Asset Tracking: RFID is employed for tracking and managing assets in various industries, including manufacturing, healthcare, and IT. It provides real-time visibility into the location and status of assets.

3. Access Control and Security: RFID technology is used for access control systems, allowing secure entry to buildings or restricted areas. RFID cards or key fobs are commonly used for employee access.

4. Payment Systems: RFID is utilized in contactless payment systems, such as credit cards or mobile payment solutions. The technology enables quick and convenient transactions by simply waving or tapping the RFID-enabled card or device.

5. Passport and ID Cards: Many modern passports and identification cards use RFID technology to store and transmit information securely. This allows for efficient border control and identity verification.

6. Livestock Tracking: In agriculture, RFID tags are used to track and manage livestock. Each animal can be uniquely identified, allowing farmers to monitor health, movement, and other vital information.

7. Retail: RFID is employed in retail for inventory management, reducing out-of-stock situations, and improving the overall efficiency of the supply chain.

Transmission code (example):

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(9, 8);  // CE, CSN
const byte address[6] = "00001";

int xAxis = A0;
int yAxis = A1;
int SW = 2;

int range = 12;
int threshold = range / 12;
int center = range / 2;

void setup() {
  pinMode(SW, INPUT);
  digitalWrite(SW, HIGH);
  Serial.begin(115200);
```

```
  radio.begin();
  radio.openWritingPipe(address);
  radio.stopListening();
}

void loop() {
  // Initialize xReading to 0
  int xReading = 0;

  // Read the x-axis value
  xReading = readAxis(A0);

  int yReading = readAxis(A1);

  // Create a data structure to send both X and Y readings
  struct Readings {
    int x;
    int y;
  };

  Readings data;
  data.x = xReading;
  data.y = yReading;

  // Send the data structure
  radio.write(&data, sizeof(data));

  Serial.print("X: ");
  Serial.println(data.x);
```

```
  Serial.print("Y: ");
  Serial.println(data.y);
   // Adjust the delay as needed
}

int readAxis(int thisAxis) {
  int reading = analogRead(thisAxis);
  reading = map(reading, 0, 1023, 0, range);
  int distance = reading - center;

  if (abs(distance) < threshold) {
    distance = 0;
  }

  return distance;
}
```

Receiver code (Example):-

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(9, 10); // CE, CSN pins on your Arduino

const byte address[6] = "00001";

void setup() {
  Serial.begin(115200);
  radio.begin();
  radio.openReadingPipe(1, address);
```

```
  radio.startListening();
}

struct Readings {
  int x;
  int y;
};

void loop() {
  if (radio.available()) {
    Readings data;
    radio.read(&data, sizeof(data));

    Serial.print("Received X: ");
    Serial.println(data.x);
    Serial.print("Received Y: ");
    Serial.println(data.y);
  }
}
```

TCP – IP address , Mac address, keys

Meet IoT Boards: ESP8266 & ESP32

The **ESP8266** is an **on-chip system** type microcontroller **SOC** of a Chinese manufacturer called the **ESPRESSIF.**

The module includes capacsystem-typea processing, reads, and control of GPIOs, and has a **TCP/IP** protocols communication and it has wireless **WiFi** of **2.4 GHz** in the **802.11 bgn** standard, with support for **WPA** and **WPA2** security encryption.

Uses

1. Internet of Things or **IOT**;

2. AUTOMATION & CONTROL Be industrial or residential, automotive or whatever field you're working in;

3. this module performs **DATA TRANSMISSION** By having inside an integrated WiFi technology we can transmit data to a router wirelessly easily and without much effort; the stack is build-in;

4. DATA PROCESSING This processing goes from reading of analog and digital sensors to complex calculations to process results comparators there may have addition, multipliers, or any other algorithm that the driver can be run onto;

5. DATA MANAGEMENT In this mode ESP receives the data as an example of a sensor and manages what to do, delivering results depending on the data type and processing which it has been programmed;

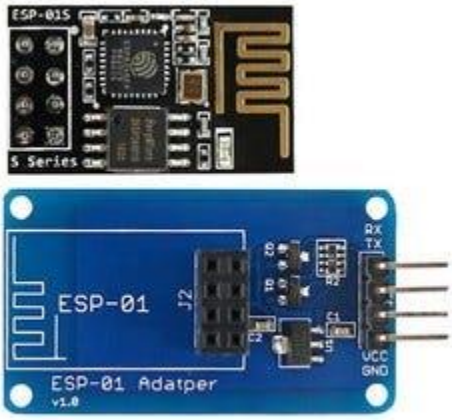
6. NETWORK CONNECTIONS With this module we can connect to any network available computer or make our own P2P net mesh;

7. WEB SERVER This module allows us to server WEB and we can access a page written in **HTML**, **PHP, Python, Ruby** or any other language of development supported by it; our skills is the limit;

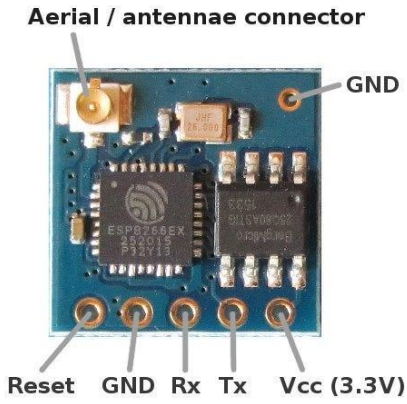
8. ACESS POINT We can also make communication between ESPs directly, no need of external network in a Point-to-Point (**P2P**) communication or access it with a SMARTPHONE, Computer or whatever device that connected directly with the ESP8266 without needing of an external network.

TYPES of ESPs

ESP-01



ESP-05



ESP-07

ESP-12E & ESP-12F

Node MCU ESP8266 (ESP-12)

ESP-32

ESP8266 (ESP-12) Characteristics

ESP8266	Description
Core	1
Arquitecture	32 bits
Clock	Xtensa LX106 80-160MHz
WiFi	IEEE802.11 b/g/n support for WPA and WPA2
Bluetooth	No
RAM	160KB - 64KB Instruction - 96KB Data
Flash	Extern QSPI - 512KB A 4MB
GPIO	16
DAC	0
ADC	1
Interfaces	SPI-I2C-UART-I2S

ESP32 Characteristics

ESP-32	Description
Core	2
Arquitecture	32 bits
Clock	Tensilica Xtensa LX106 160-240MHz
WiFi	IEEE802.11 b/g/n
Bluetooth	Yes - classic & BLE
RAM	520KB
Flash	Extern QSPI - 16MB
GPIO	22
DAC	2
ADC	18
Interfaces	SPI-I2C-UART-I2S-CAN

Sender(8266) :

```
#include <ESP8266WiFi.h>
#include <espnow.h>

uint8_t broadcastAddress[] = {0x2C, 0x3A, 0xE8, 0x43, 0xB9, 0xBC};

#define BOARD_ID 2
#define BOARD_ID 2

// Structure example to send data
// Must match the receiver structure
typedef struct struct_message {
```

```
    int id;
    int x;
    int y;
} struct_message;

// Create a struct_message called test to store variables to be sent
struct_message myData;

unsigned long lastTime = 0;
unsigned long timerDelay = 10000;

void OnDataSent(uint8_t *mac_addr, uint8_t sendStatus) {
  Serial.print("\r\nLast Packet Send Status: ");
  if (sendStatus == 0){
    Serial.println("Delivery success");
  }
  else{
    Serial.println("Delivery fail");
  }
}
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);

  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  if (esp_now_init() != 0) {
```

```
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  esp_now_set_self_role(ESP_NOW_ROLE_CONTROLLER);

  // Once ESPNow is successfully init, we will register for Send CB to
  // get the status of Trasnmitted packet
  esp_now_register_send_cb(OnDataSent);

  // Register peer
  esp_now_add_peer(broadcastAddress,  ESP_NOW_ROLE_SLAVE, 1, NULL, 0);

}

void loop() {
  // put your main code here, to run repeatedly:
  if ((millis() - lastTime) > timerDelay) {
    // Set values to send
    myData.id = BOARD_ID;
    myData.x = random(1, 50);
    myData.y = random(1, 50);

    // Send message via ESP-NOW
    esp_now_send(0, (uint8_t *) &myData, sizeof(myData));
    lastTime = millis();
  }
}
```

Receiver (8266)-

```
#include <ESP8266WiFi.h>
#include <espnow.h>

// Structure example to receive data
// Must match the sender structure
typedef struct struct_message {
    int id;
    int x;
    int y;
} struct_message;
```

```
// Create a struct_message called myData
struct_message myData;
void OnDataRecv(uint8_t * mac_addr, uint8_t *incomingData, uint8_t len) {
  char macStr[18];
  Serial.print("Packet received from: ");
  snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
           mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3], mac_addr[4], mac_addr[5]);
  Serial.println(macStr);
  memcpy(&myData, incomingData, sizeof(myData));

  Serial.printf("x value: %d \n", myData.x);
  Serial.printf("y value: %d \n", myData.y);
  Serial.println();
}
void setup() {
  // Initialize Serial Monitor
  Serial.begin(115200);

  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();

  // Init ESP-NOW
  if (esp_now_init() != 0) {
    Serial.println("Error initializing ESP-NOW");
    return;
```

```
  }
// Once ESPNow is successfully Init, we will register for recv CB to
  // get recv packer info
  esp_now_set_self_role(ESP_NOW_ROLE_SLAVE);
  esp_now_register_recv_cb(OnDataRecv);
}
void loop(){

}
```

ARDUINO(intro)

Hi Friends! Hope you are doing great. Today, I am going to give you a detailed **Introduction to Arduino Uno**. It is a microcontroller board developed by Arduino.cc and is based on **Atmega328 Microcontroller**. The first Arduino project was started in Interaction Design Institute Ivrea in **2003 by David Cuartielles and Massimo Banzi** with the intention of providing

a cheap and flexible way for students and professionals to learn embedded programming.

Arduino UNO is a very valuable addition in electronics that consists of a USB interface, 14 digital I/O pins(of which 6 Pins are used for PWM), 6 analog pins and an Atmega328 microcontroller. It also supports 3 communication protocols named Serial, I2C and SPI protocol. You should also have a look at this video presentation on Arduino UNO:

- Few main features of Arduino UNO are shown in the below figure:

No.	**Parameter Name**	**Parameter Value**
1	Microcontroller	Atmega328
2	Crystal Oscillator	16MHz
3	Operating Voltage	5V
4	Input Voltage	5-12V
5	Digita0l I/O Pins	14 (D0 to D13)
6	Analog I/O Pins	6 (A0 to A5)
7	PWM Pins	6 (Pin # 3, 5, 6, 9, 10 and 11)
8	Power Pins	5V, 3.3V, Vin, GND
9	Communication	UART(1), SPI(1), I2C(1)
10	Flash Memory	32 KB (0.5KB is used by bootloader)

11	SRAM	2 KB
12	EEPROM	1 KB
13	ICSP Header	Yes
14	Power sources	DC Power Jack & USB Port

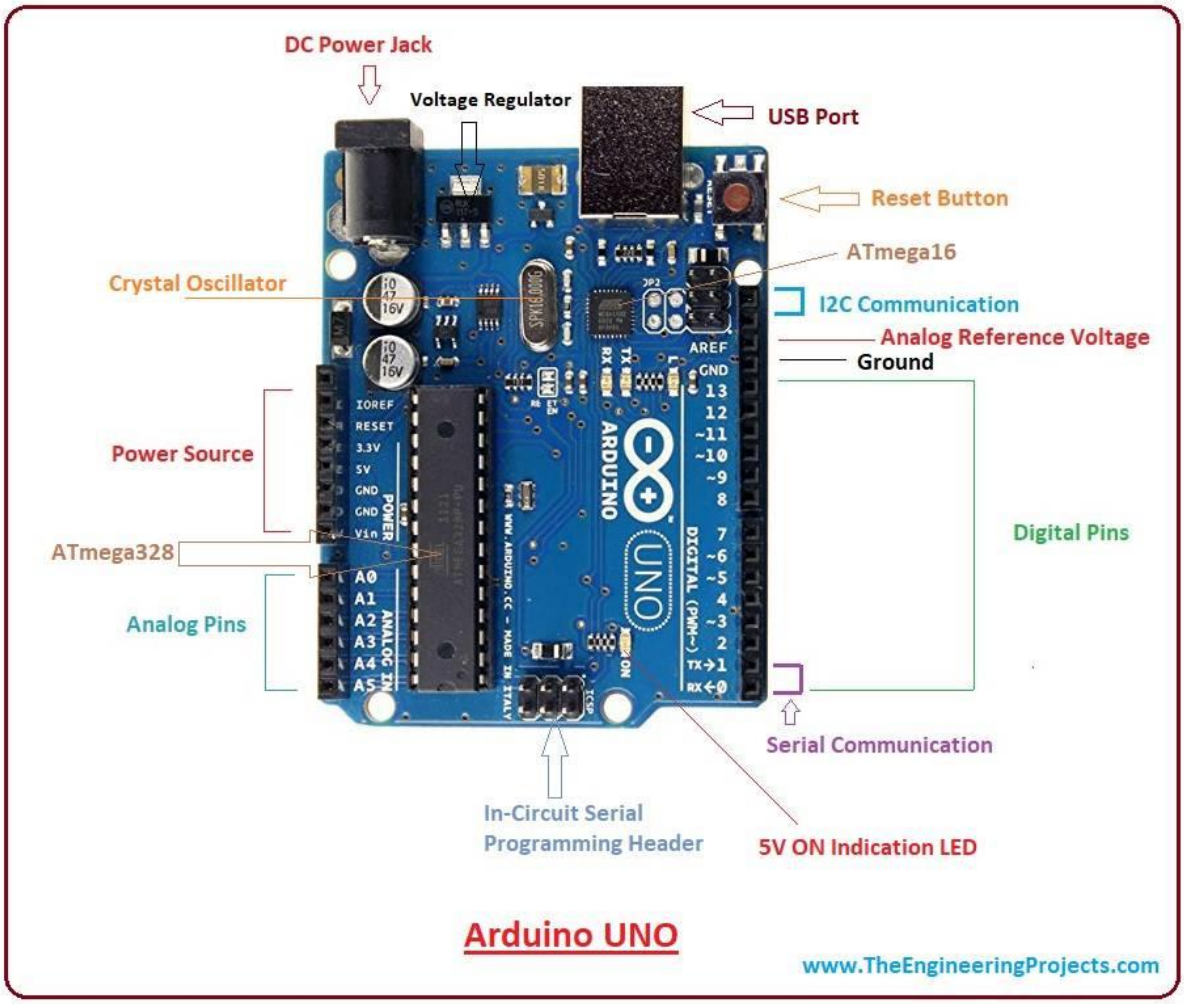
I'll try to cover each and everything related to Arduino Uno, so you get a clear idea of what it does, its main features, working and everything you need to know. Let's get started.

Introduction to Arduino-

- **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.
- It comes with a **crystal oscillator of 16MHz**, which is its operating frequency.

- **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**.
- 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**
- You can download the Arduino UNO datasheet by clicking the below button:

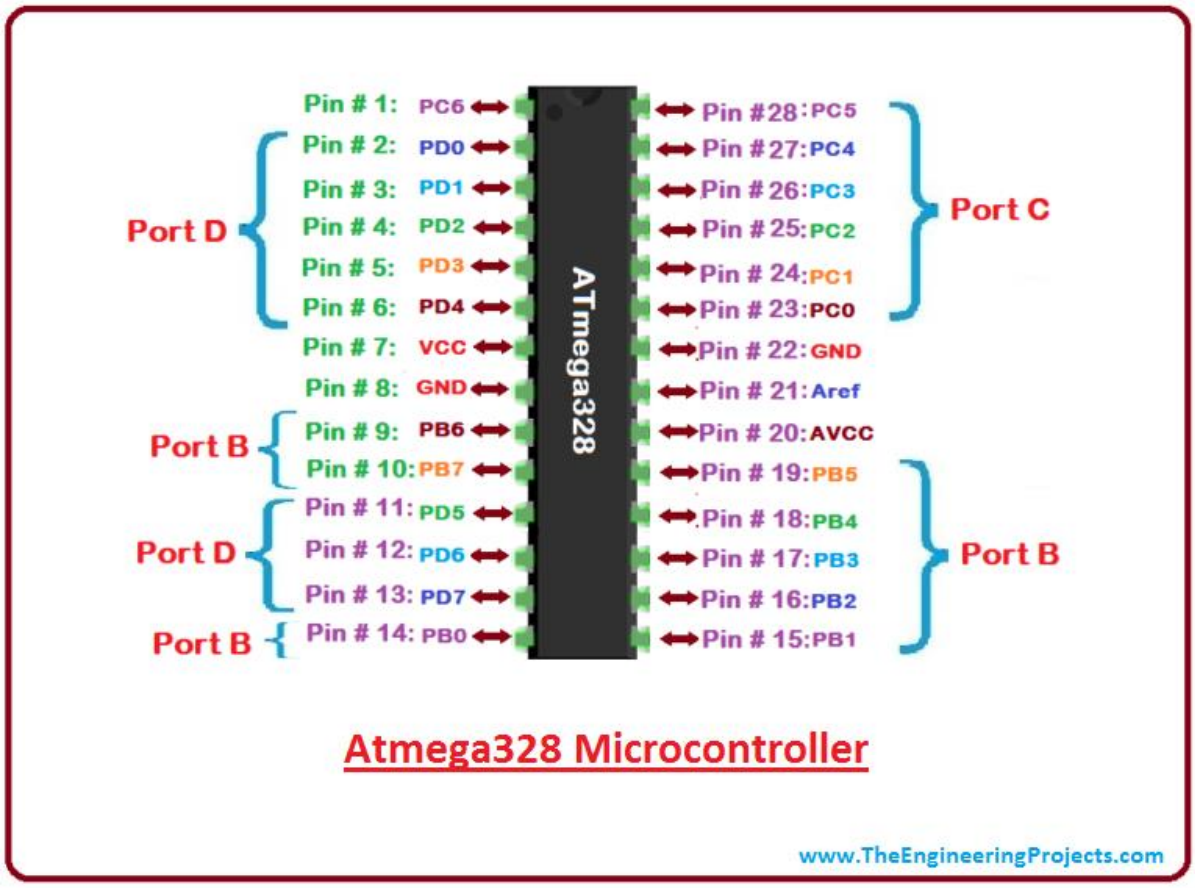
Download Arduino UNO Datasheet(for more details about arduino)



- Apart from USB, a battery or AC to DC adopter can also be used to power the board.

- Arduino Uno comes with a USB interface i.e. USB port is added on the board to develop serial communication with the computer.
- Atmega328 microcontroller is placed on the board that comes with a number of features like timers, counters,

interrupts, PWM, CPU, I/O pins and based on a 16MHz clock that helps in producing more frequency and number of instructions per cycle.



- It is an open-source platform where anyone can modify and optimize the board based on the number of instructions and tasks they want to achieve.
- This board comes with a built-in regulation feature that keeps the voltage under control when the device is connected to the external device.
- A reset pin is present in the board that resets the whole board and takes the running program in the initial stage. This pin is useful when the board hangs up in the middle

of the running program; pushing this pin will clear everything up in the program and starts the program right from the beginning.

- There are 14 I/O digital and 6 analog pins incorporated in the board that allows the external connection with any circuit with the board. These pins provide flexibility and ease of use to the external devices that can be connected through these pins. There is no hard and fast interface required to connect the devices to the board. Simply plug the external device into the pins of the board that are laid out on the board in the form of the header.
- The 6 analog pins are marked as A0 to A5 and come with a resolution of 10bits. These pins measure from 0 to 5V, however, they can be configured to the high range using analogReference() function and AREF pin.
- Only 5 V is required to turn the board on, which can be achieved directly using a USB port or external adopter, however, it can support an external power source up to 12 V which can be regulated and limit to 5 V or 3.3 V based on the requirement of the project.

Arduino Uno Pinout

Arduino Uno is based on an AVR microcontroller called Atmega328. This controller comes with 2KB SRAM, 32KB of flash memory, 1KB of EEPROM. Arduino Board comes with 14 digital pins and 6 analog pins. ON-chip ADC is used to sample these pins. A 16 MHz frequency crystal oscillator is equipped on the

board. The following figure shows the pinout of the Arduino Uno Board.

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.

- **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 provide 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.

Arduino Uno comes with the ability of interfacing with other Arduino boards, microcontrollers and computers. **The Atmega328 placed on the board provides serial communication using pins like Rx and Tx.** The Atmega16U2 incorporated on the board provides a pathway for serial communication using USB com drivers. A serial monitor is provided on the IDE software which is used to send or receive text data from the board. If LEDs placed on the Rx and Tx pins will flash, they indicate the transmission of data. Arduino Uno is programmed using Arduino Software which is a cross-platform application called IDE written in Java. The AVR microcontroller Atmega328 laid out on the base comes with built-in bootloader that sets you free from using a separate burner to upload the program on the board.

Arduino Codes:

1) Led Blink-

```
// Define the LED pin
const int ledPin = 2;

void setup() {
  // Set the LED pin as an output
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // Turn on the LED
  digitalWrite(ledPin, HIGH);

  // Wait for 1 second
  delay(1000);
```

```
  // Turn off the LED
  digitalWrite(ledPin, LOW);

  // Wait for 1 second
  delay(1000);
}
```

2)ULTRASONIC SENSOR

```
#include <Servo.h>

// Define ultrasonic sensor pins
const int trigPin = 9; // Trigger pin
const int echoPin = 10; // Echo pin

// Define servo motor pin
const int servoPin = 6;
```

```
// Create servo object
Servo myservo;

void setup() {
  // Initialize Serial communication
  Serial.begin(9600);

  // Set up ultrasonic sensor pins
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  // Attach the servo to its pin
  myservo.attach(servoPin);
}

void loop() {
  // Measure distance using ultrasonic sensor
  long duration, distance;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
```

```
  distance = (duration / 2) / 29.1; // Convert distance to centimeters

  // Print the distance to Serial Monitor
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  // Map the distance to servo angle
  int angle = map(distance, 5, 30, 0, 180);

  // Limit the servo angle to prevent mechanical issues
  angle = constrain(angle, 0, 180);

  // Move the servo motor to the calculated angle
  myservo.write(angle);

  // Wait for a short time before the next measurement
  delay(100);
}
```

3.RX TX

Sender Arduino Code:

```
char Mymessage[5] = "Hello"; //String data
void setup() { // Begin the Serial at 9600 Baud
 Serial.begin(9600);
 }
 void loop()
{
Serial.write(Mymessage,5); //Write the serial data
delay(1000);
}
```

Receiver Arduino Code:

```
char Mymessage[10]; //Initialized variable to store recieved data
 void setup()
{
 // Begin the Serial at 9600 Baud
Serial.begin(9600);
}
 void loop() {
 Serial.readBytes(Mymessage,5); //Read the serial data and store in var
Serial.println(Mymessage); //Print data on Serial Monitor
delay(1000);
}
```

NODEMCU CODES

1) live ultrasonic sensor on webpage

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>

const char* ssid = "RohitV";
const char* password = "12345678";

ESP8266WebServer server(80);

const int trigPin = D5;
const int echoPin = D6;

long duration;
float distance;

void setup() {
pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
```

```
pinMode(echoPin, INPUT); // Sets the echoPin as an Input
Serial.begin(9600); // Starts the serial communication
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected..!");
  Serial.print("Got IP: ");  Serial.println(WiFi.localIP());

  server.on("/", handle_OnConnect);
  server.onNotFound(handle_NotFound);

  server.begin();
  Serial.println("HTTP server started");
}
void loop() {
  server.handleClient();
}

void handle_OnConnect(){
  digitalWrite(trigPin, LOW);
```

```
  delayMicroseconds(2);

  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);

  // Calculating the distance
  distance= duration*0.034/2;
  // Prints the distance on the Serial Monitor
  Serial.print("Distance: ");
  Serial.println(distance);
  server.send(200, "text/html", SendHTML(distance));
}
void handle_NotFound(){
  server.send(404, "text/plain", "Not found");
}
String SendHTML(float distance){
  String ptr = "<!DOCTYPE html> <html>\n";
  ptr +="<head><meta name=\"viewport\" content=\"width=device-width,
initial-scale=1.0, user-scalable=no\">\n";
```

```
  ptr +="<title>ESP8266 distance report</title>\n";
  ptr +="<script>function autoRefresh() {"
       " window.location = window.location.href;}"
       "setInterval('autoRefresh()', 2000);</script>";
  ptr +="<style>html { font-family: Helvetica; display: inline-block; margin: 0px
auto; text-align: center;}\n";
  ptr +="body{margin-top: 50px;} h1 {color: #444444;margin: 50px auto
30px;}\n";
  ptr +="p {font-size: 24px;color: #444444;margin-bottom: 10px;}\n";
  ptr +="</style>\n";
  ptr +="</head>\n";
  ptr +="<body>\n";
  ptr +="<div id=\"webpage\">\n";
  ptr +="<h1>Node MCU ultrasonic sensor</h1>\n";

  ptr +="<p>distance: ";
  ptr +=(int)distance;



  ptr +="</div>\n";
  ptr +="</body>\n";
  ptr +="</html>\n";
  return ptr;
}
```

2) Communication

Client code

```
#include <ESP8266WiFi.h>

const char* ssid = "iPhone "; // Replace with your Wi-Fi network name
const char* password = "12345678"; // Replace with your Wi-Fi password
const char* server_ip = "172.20.10.2"; // Replace with NodeMCU 1's IP address
WiFiClient client;

void setup() {
  Serial.begin(115200);
  delay(10);

  // Connect to Wi-Fi
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
```

```
  // Set up the client
  Serial.print("Connecting to server: ");
  Serial.println(server_ip);

  if (client.connect(server_ip, 80)) {
    Serial.println("Connected to server");
  } else {
    Serial.println("Connection failed");
  }
}

void loop() {
  // Connect to the server
  if (client.connected()) {
    String line = client.readStringUntil('\n');
    if (line != "") {
      Serial.println("Server says: " + line);
    }
  } else {
    Serial.println("Disconnected from server");
    client.stop();
```

```
    // Attempt to reconnect
    if (client.connect(server_ip, 80)) {
      Serial.println("Reconnected to server");
    } else {
      Serial.println("Reconnection failed");
    }
  }

  delay(1000);
}
```

Server Code

```
#include <ESP8266WiFi.h>

const char* ssid = "iPhone "; // Remove the space at the end of the SSID
const char* password = "12345678";
WiFiServer server(80);

void setup() {
  Serial.begin(115200);
  delay(10);
```

```
  // Connect to Wi-Fi
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
  Serial.println(WiFi.localIP());



  server.begin();
}

void loop() {
  WiFiClient client = server.available();
  if (client) {
    // Serial.println("Client connected");

    while (client.connected()) {
      if (Serial.available()) {
        char message = Serial.read(); // Read a character from the Serial Monitor
        client.print(message); // Send the character to the client
```

```
      Serial.print("Sent to client: ");
      Serial.println(message);
    }

    // if (client.available()) {
    //   char response = client.read(); // Read data from the client (if any)
    //   // Do something with the received data from the client (if needed)
    // }
  }

  // Client disconnected
  // Serial.println("Client disconnected");
 }
}
```

Introduction to Raspberry Pi(Rpi)

What is Raspberry Pi?

Short for RPi, it is a credit-card-sized single-board computer built by the **Raspberry Pi** Foundation in association with Broadcom in the United Kingdom. It can do most of the things your typical desktop does, such as handling spreadsheets, word-processing, and games. Raspberry Pi also has the capacity to play videos in high-definition. It can run several versions of Linux and is used to teach kids worldwide how to program. In fact, that was one of the biggest reasons why the idea of Raspberry Pi was conceived. But more on that later.

In addition, it has inputs and outputs for sensors so that you can attach all sorts of hardware to it. The General Purpose Input/Output (GPIO) pins on Raspberry Pi allow you to connect status lights, switches, analog signals, and more. You can use C++ or Python to control the board to sense or control devices attached to it.

Invented to stimulate the teaching of basic computer science in schools, Raspberry Pi has since then been used in many applications, including hobbyist projects, hardware platforms for electronics design, embedded devices, and robotics.

Raspberry Pi can run various operating systems such as Raspbian OS, Snappy Ubuntu Core, Ubuntu Mate, and Windows 10 IoT core.

Powering the Raspberry Pi

One of the most popular operating systems is the Raspbian Operating system, which is based on the Debian OS and is optimized for the Raspberry Pi hardware. The easiest way to deploy the Raspbian OS is to download NOOBS from its **official site**.

By the way, NOOBS is short for New Out Of Box Software. The Raspbian OS has a micro-SD card on which the entire operating system operates.

A typical Class 4 8GB micro-SD card suffices for most purposes, but if you have the opportunity to connect it to an external hard disk for extra storage, then grab that option with both hands.

Once you have installed the Raspbian OS, you can proceed to log in. While the default password is raspberry, the username is pi. Please change it to something else after the initial login.

IoT and the maker culture

The maker culture is rightfully a contemporary subculture, a technology-based extension of DIY culture intersecting with the hacker culture's hardware-oriented parts. It is a part of both new devices and existing ones.

IoT represents an emerging technology connecting devices wirelessly using communication protocols to exchange data. It can be as simple as a thermostat that can be operated remotely because both devices are connected via the internet.

Or as complex as large-scale connecting sensors, actuators, and controls that monitor and control physical devices in remote locations using wireless communications. Specific examples include

RFID tags, RFID readers, barcode scanners, GPS receivers, and other network-enabled smart devices.

Using the IoT technology with Raspberry Pi 3 allows you to monitor and control devices remotely, collect and exchange data and create automation systems with relative ease. Moreover, Raspberry Pi 3 can be expanded by adding different sensors and modules.

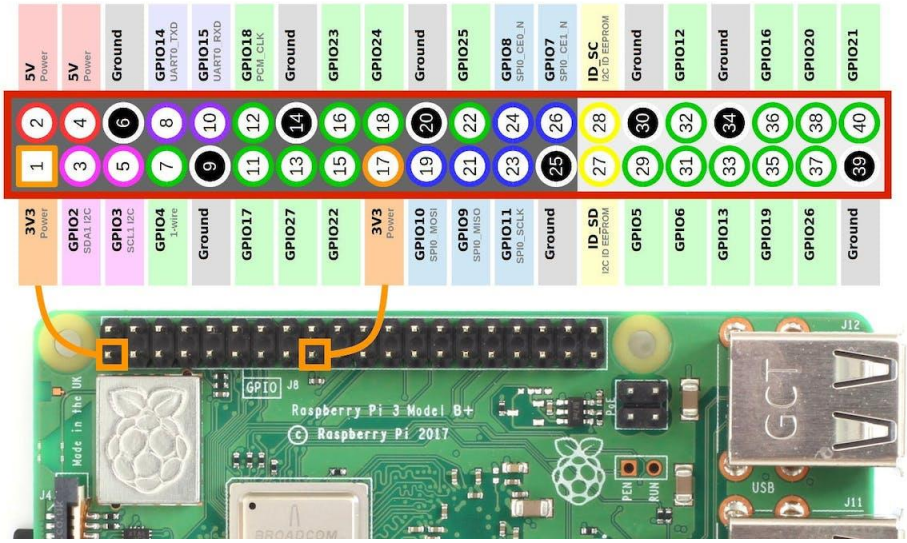
As we studied previously, the maker culture is also known as the "DIY movement." It offers a way for people to become creators by designing products from raw materials such as wood or metal with their own hands. You could say the culture shines a light on being self-sufficient and building projects for oneself.

If you consider yourself a maker, inventor, and tinkerer, that makes you a part of the maker culture. It gives you the freedom to create things from scratch and use tools to help you achieve your goals. That is precisely what Raspberry Pi allows in the IoT ecosystem.

GPIO Pins — Connecting Raspberry Pi to the outside world

GPIO stands for General Purpose Input Output. One of the most exciting features of Raspberry Pi is GPIO pins, which are uncommitted digital signal pins acting as the electronic interface between the Raspberry Pi board and various external electronic components.

We can use these GPIO pins to control or monitor other circuitry on a board. In other words, the GPIO pins in Raspberry Pi are unique because they allow direct access to the input/output circuitry of the computer.

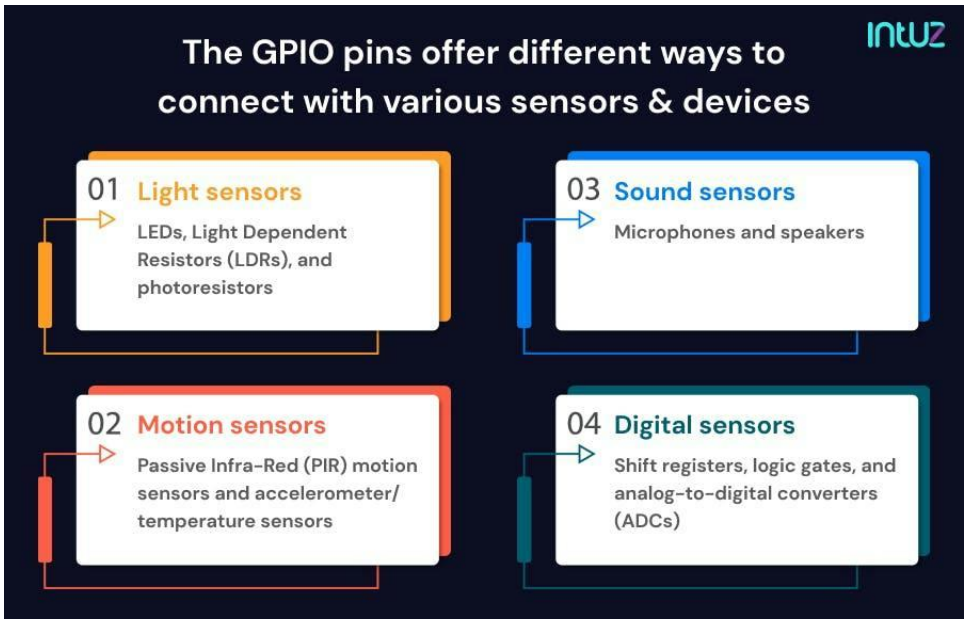


This functionality allows you to directly interface sensors and other components with your Raspberry Pi project. There are 26 GPIO pins on Raspberry Pi and two programmable I2C.

These pins are vital when creating an IoT device because they allow you to hook up components such as LEDs, buttons, and touch screens directly to your computer.

GPIO pins allow you to send and receive information out of and into your Raspberry Pi. Basically, the input pins will enable you to pull data into your Pi, and the output pins allow you to get information out of your Pi.

The GPIO pins provide several different ways for you to connect with various sensors and devices, including:



- Light sensors — LEDs, Light Dependent Resistors (LDRs), and photoresistors
- Motion sensors — Passive Infra-Red (PIR) motion sensors and accelerometer/temperature sensors
- Sound sensors — Microphones and speakers
- Digital sensors — Shift registers, logic gates, and analog-to-digital converters (ADCs)

You can connect these sensors directly to the GPIO pins. In fact, Raspberry Pi can control LEDs, turn them on or off, interact with many other objects present in the outside world, drive motors, and so on. It can also detect a change in temperature or light and the pressing of a switch with the help of sensors.

Raspberry Pi Uses

Now that you know what Raspberry Pi is, you may investigate its many applications.

Desktop

With just a Raspberry Pi, a microSD card, and some electricity, you can put together a basic desktop computer. You will also need an HDMI cable and a monitor or other appropriate display. A USB keyboard and mouse are also required.

Robotics Controller

Numerous robot-controller projects may be implemented using a Raspberry Pi. Pi has a dedicated robotics package that can communicate with and operate robots; it runs from the device's battery.

Printing By means of a Raspberry Pi

Basically anything can be printed with a Raspberry Pi. A Raspberry Pi and print server software are all that's required to set up a printing system in your house. To do this, first install the Samba file-sharing application, and then set up CUPS to use it. Printer drivers and a control panel are both part of the Common Unix Printing System (CUPS).

Game Servers

The Raspberry Pi's base OS comes with a customized build of the popular video game Minecraft already loaded. The Raspberry Pi can host games with the right software. The server is excellent for playing Minecraft. Using a large number of Raspberry Pis, a fantastic gaming environment may be built.

Gaming Machine

In this respect, the Raspberry Pi is ideal. It's one of the machine's most lightweight parts. One model, the Raspberry Pi Zero, is ideal for usage in compact settings for DIY video game development. Many classic 16-bit video game consoles may be brought back to life with the help of a Raspberry Pi.