

### **Experiment List (IOT Lab)**

1. Familiarise with Arduino UNO and interface LED/Buzzer with Arduino and write a program to turn ON LED for 1 sec after every 2 seconds.
2. To interface the Push button with Arduino, write a program to turn ON the LED when a push button is pressed.
3. To interface the LDR sensor with Arduino, write a program to print LDR readings on a serial monitor.
4. According to the LDR readings, write a program to turn ON & OFF multiple LEDs.
5. To interface the DHT11 sensor with Arduino, write a program to print temperature and humidity readings on a serial monitor.
6. According to the DHT readings, write a program to turn ON & OFF multiple LEDs.
7. To interface the HC-05 Bluetooth module with Arduino, write a program to turn ON & OFF multiple LEDs with a smartphone using Bluetooth.
8. Write a program to send sensor data to a smartphone using the HC-05 Bluetooth module with Arduino.
9. Familiarise with the ESP32 Development board and write a program to set up the Bluetooth and Wi-Fi features of ESP32 MCU.
10. Write a program to control multiple LEDs using ESP32 and mobile devices with the help of MQTT Protocol.
11. Familiarisation with BLYNK Cloud Platform to set up BLYNK Cloud dashboard and control multiple LEDs using a mobile device using ESP32
12. Upload sensor data to the cloud and retrieve it on any mobile device using the BLYNK Cloud Platform.
13. Write a Program in ESP32 to control LEDs/Home appliances with the help of Google Assistant/Amazon Alexa.
14. Project Work based on various applications of IoT.

**Experiment 1: Familiarise with Arduino UNO and interface LED/Buzzer with Arduino and write a program to turn ON LED for 1 sec after every 2 seconds.**

### Components Required

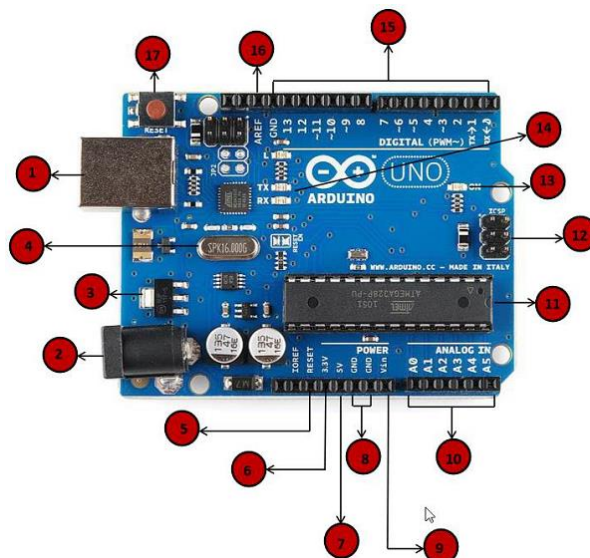
- 1 × Breadboard
- 1 × Arduino Uno R3
- 1 × LED
- 1 × 330Ω Resistor
- 2 × Jumper








Arduino is a prototype platform (open-source) based on easy-to-use hardware and software. It consists of a circuit board, which can be programmed (a microcontroller) and ready-made software called Arduino IDE (Integrated Development Environment), which is used to write and upload the computer code to the physical board.







The key features are –

- Arduino boards can read analog or digital input signals from different sensors and turn them into an output, such as activating a motor, turning LED on/off, connecting to the cloud and many other actions.
- You can control your board functions by sending instructions to the microcontroller on the board via Arduino IDE (referred to as uploading software).
- Unlike most previous programmable circuit boards, Arduino does not need an extra piece of hardware (a programmer) to load a new code onto the board. You can use a USB cable.
- Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program.
- Finally, Arduino provides a standard form factor that breaks the microcontroller's functions into a more accessible package.

**Arduino UNO** is the most popular board in the Arduino board family. It is the best board to get started with electronics and coding. Some panels look slightly different from the one below, but most Arduinos have most of these components in common.

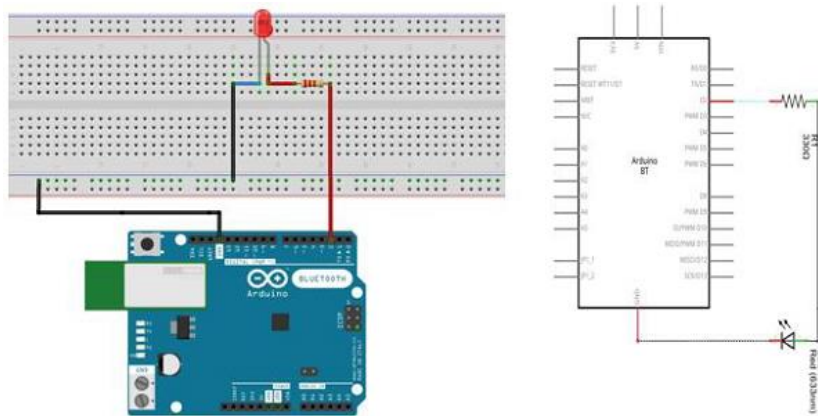


	<p><b>Power USB</b></p> <p>Arduino board can be powered by using the USB cable from your computer. You only need to connect the USB cable to the USB connection (1).</p>
	<p><b>Power (Barrel Jack)</b></p> <p>Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2).</p>
	<p><b>Voltage Regulator</b></p> <p>The voltage regulator controls the voltage given to the Arduino board and stabilizes the DC voltages used by the processor and other elements.</p>
	<p><b>Crystal Oscillator</b></p> <p>The crystal oscillator helps Arduino in dealing with time issues. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz.</p>
	<p><b>Arduino Reset</b></p> <p>You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways: first, by using the reset button (17) on the board. Second, connect an external reset button to the Arduino pin labelled RESET (5).</p>
	<p><b>Pins (3.3, 5, GND, Vin)</b></p> <ul style="list-style-type: none"> <li>• 3.3V (6) – Supply 3.3 output volt</li> <li>• 5V (7) – Supply 5 output volt</li> <li>• Most of the components used with the Arduino board work fine with 3.3 volts and 5 volts.</li> <li>• GND (8)(Ground) – There are several GND pins on the Arduino, which can be used to ground your circuit.</li> <li>• Vin (9) – This pin can also power the Arduino board from an external power source, like an AC mains power supply.</li> </ul>
	<p><b>Analog pins</b></p> <p>The Arduino UNO board has six analog input pins, A0 through A5. These pins can read the signal from an analog sensor like the humidity or temperature sensor and convert it into a digital value that the microprocessor can read.</p>

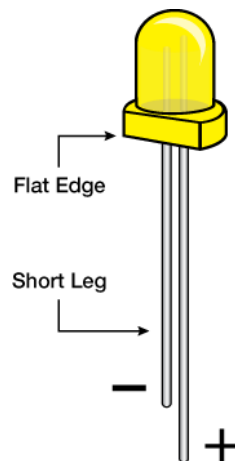
	<p><b>Main microcontroller</b></p> <p>Each Arduino board has its own microcontroller (11). You can assume it is the brain of your board. The Arduino's main IC (integrated circuit) differs slightly from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. Refer to the datasheet for details about the IC construction and functions.</p>
	<p><b>ICSP pin</b></p> <p>ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often called an SPI (Serial Peripheral Interface), which could be considered an “expansion” of the output. You are slaving the output device to the master of the SPI bus.</p>
	<p><b>Power LED indicator</b></p> <p>This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, the connection is wrong.</p>
	<p><b>TX and RX LEDs</b></p> <p>You will find two labels on your board: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, indicate the pins responsible for serial communication. Second, the TX and RX led (13). The TX LED flashes at different speeds while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.</p>
	<p><b>Digital I/O</b></p> <p>The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labelled “~” can be used to generate PWM.</p>
	<p><b>AREF</b></p> <p>AREF stands for Analog Reference. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.</p>

## Procedure

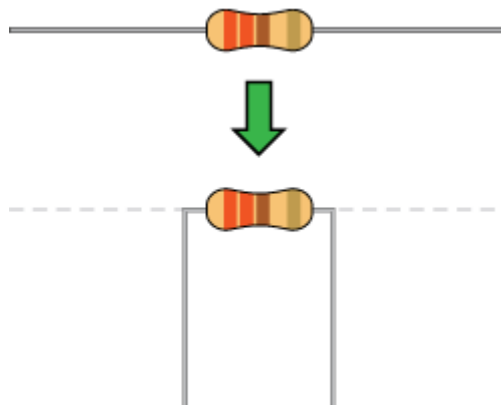
Follow the circuit diagram and hook up the components on the breadboard, as shown in the image below.



**Note** – To find out the polarity of an LED, look at it closely. The shorter of the two legs, towards the flat edge of the bulb, indicates the negative terminal.

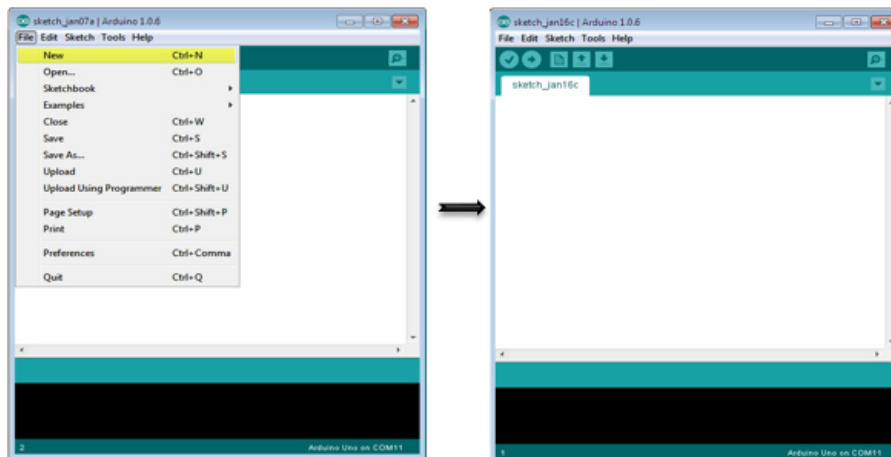


Components like resistors must have their terminals bent into 90° angles to fit the breadboard sockets properly. You can also cut the terminals shorter.



## Sketch

Open the Arduino IDE software on your computer. Coding in the Arduino language will control your circuit. Open the new sketch File by clicking New.



## Arduino Code

// The setup function runs once when you press reset or power the board

```
void setup()
{
  pinMode(2, OUTPUT);    // initialize digital pin 13 as an output.
}
```

//The loop function runs over and over again forever

```
void loop()
{
  digitalWrite(2, HIGH);    // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(2, LOW);    // turn the LED off by making the voltage LOW
  delay(2000);              // wait for a second
}
```

## Code to Note

**pinMode(2, OUTPUT)** – Before using one of Arduino’s pins, you must tell Arduino Uno R3 whether it is an INPUT or OUTPUT. To do this, we use a built-in “function” called pinMode().

**digitalWrite(2, HIGH)** – When you are using a pin as an OUTPUT, you can command it to be HIGH (output 5 volts) or LOW (output 0 volts).

## Result

You should see your LED turn on and off. If the required output is not seen, ensure you have assembled the circuit correctly and verified and uploaded the code to your board.

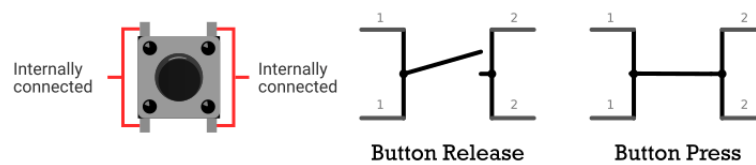
**Experiment 2:** To interface the Push button with Arduino, write a program to turn ON the LED when a push button is pressed.

### Components Required

- 1 × Breadboard
- 1 × Arduino Uno R3
- 1 × LED
- 1 × Push button
- 1 × 10K $\Omega$  Resistor
- 1 × 330 $\Omega$  Resistor
- 3 × Jumper

Push buttons come in different shapes and sizes. However, the internal structure and working principle are the same for most push buttons.

Push buttons usually have four pins. But these pins are connected internally in pairs. So, you have two connections. When you press the button, these two terminals will connect. And when you release the button, these terminals will disconnect.

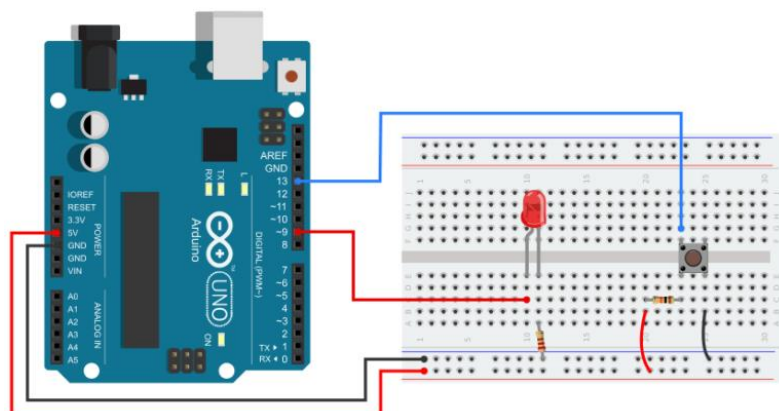


In the above image, you can see the schematic of a push button. It has two terminals marked 1 and 2. When you press the button, terminals 1 and 2 will connect.

### How to Connect a Push Button with Arduino

Connecting a push button with an Arduino is very simple. Connect one push button terminal to the ground pin and another to any Arduino digital pins. You must use a pull-up resistor (10k  $\Omega$ ) to keep the voltage HIGH when not pressing the button.

The pull-up resistor is a high-value resistor connecting to the Arduino digital pin you are using with the HIGH (5v) voltage.



Note that – If you use a pull-up configuration, the button input pin gets logic HIGH input when you are not pressing the button. And when you press the button, it gets logic LOW input.

### **Arduino Code**

```
const int buttonPin = 13;
const int ledPin = 9;
void setup()
{
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}
void loop()
{
  currentState = digitalRead(buttonPin);
  if (currentState == LOW)
  {
    digitalWrite(ledPin, HIGH);
  }
  else if (currentState == HIGH)
  {
    digitalWrite(ledPin, LOW);
  }
  delay(50);
}
```

### **Code to Note**

When the switch is open (the pushbutton is not pressed), there is no connection between the two pushbutton terminals, so the pin is connected to the VCC (through the pull-up resistor), and we read a HIGH. When the switch is closed (pushbutton is pressed), it combines its two terminals, connecting the pin to the ground, so we read a LOW.

### **Result**

LED is turned ON when the pushbutton is pressed and OFF when released.



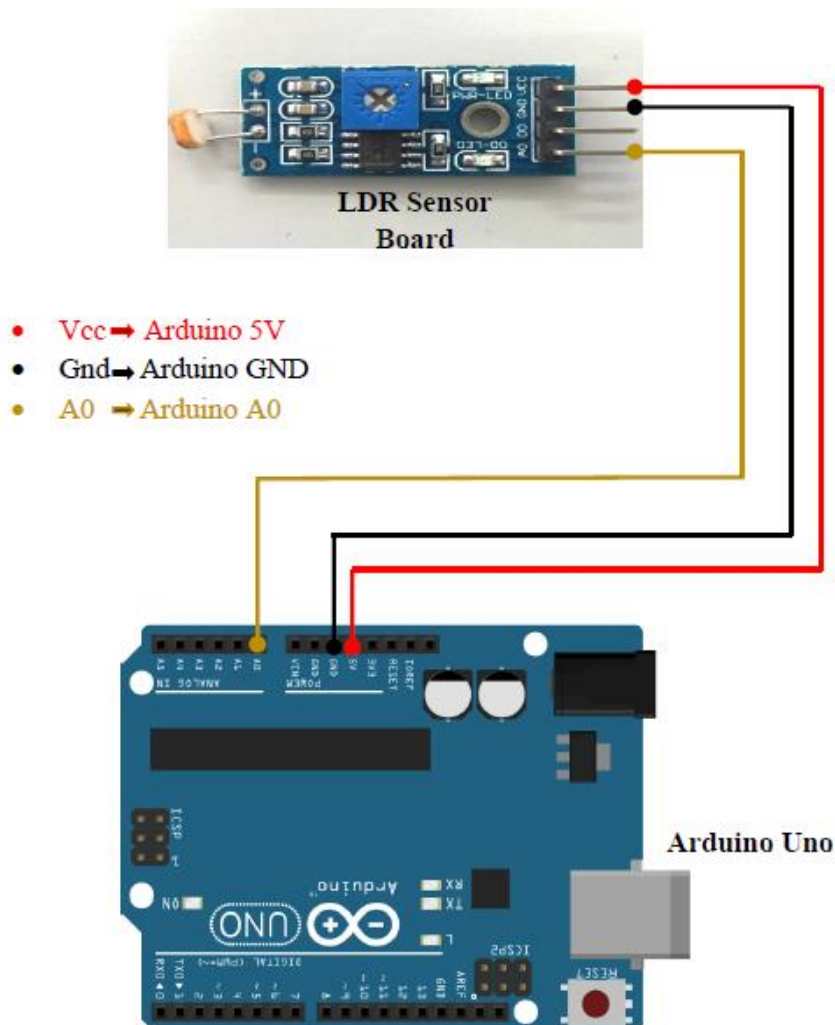
**Experiment 3: To interface the LDR sensor with Arduino, write a program to print LDR readings on a serial monitor.**

### Components Required

- 1 × Arduino Uno R3
- 1 × LDR Module
- 3 × Jumper

LDR sensor module is used to detect the intensity of light. It is associated with analog output pin and digital output pin labelled AO and DO on the board, respectively. When there is light, the resistance of LDR will become low according to light intensity. The greater the intensity of light, the lower the resistance of LDR. The sensor has a potentiometer knob that can be adjusted to change the sensitivity of LDR towards light.

### How to Connect an LDR Module with Arduino



## Arduino Code

```
void setup()

{

Serial.begin(9600);

}

void loop()

{

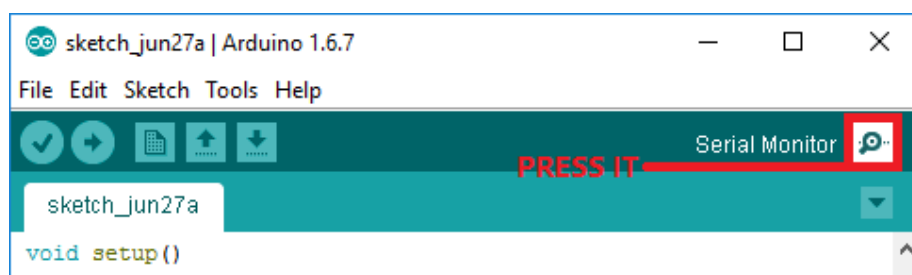
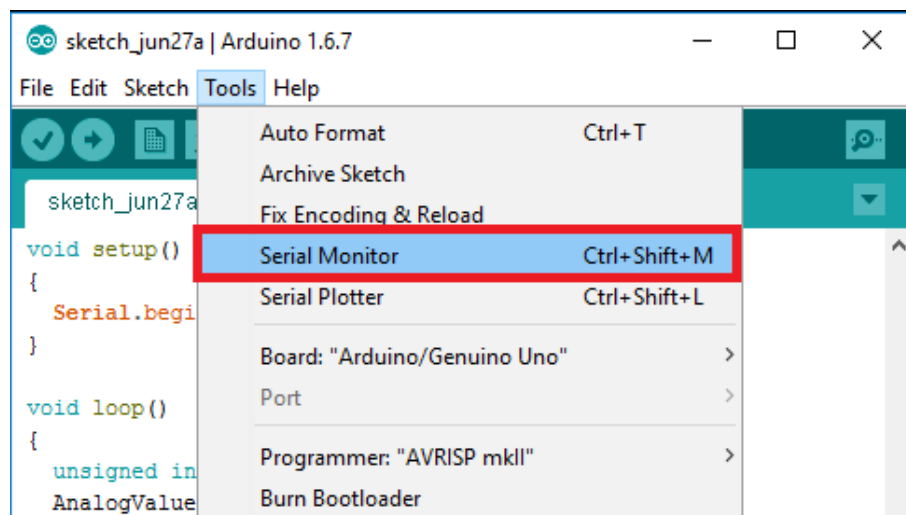
unsigned int AnalogValue;

AnalogValue = analogRead(A0);

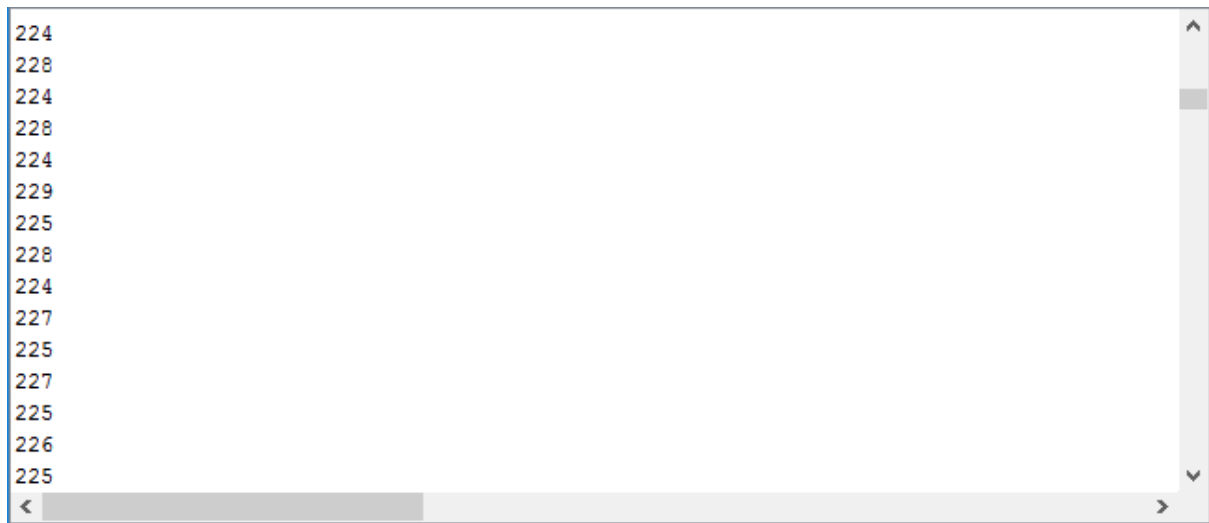
Serial.println(AnalogValue);

}
```

## How to open the serial monitor



## Result



A screenshot of a serial monitor window displaying a list of numerical readings. The readings are: 224, 228, 224, 228, 224, 229, 225, 228, 224, 227, 225, 227, 225, 226, 225. The window has a vertical scrollbar on the right and a horizontal scrollbar at the bottom.

224
228
224
228
224
229
225
228
224
227
225
227
225
226
225

The reading is shown on the Serial Monitor when the LDR sensor board is exposed to light.



A screenshot of a serial monitor window displaying a list of numerical readings. The readings are: 910, 909, 909, 910, 910, 909, 909, 910, 910, 909, 910, 910, 910, 909, 910. The window has a vertical scrollbar on the right and a horizontal scrollbar at the bottom.

910
909
909
910
910
909
909
910
910
909
910
910
910
909
910

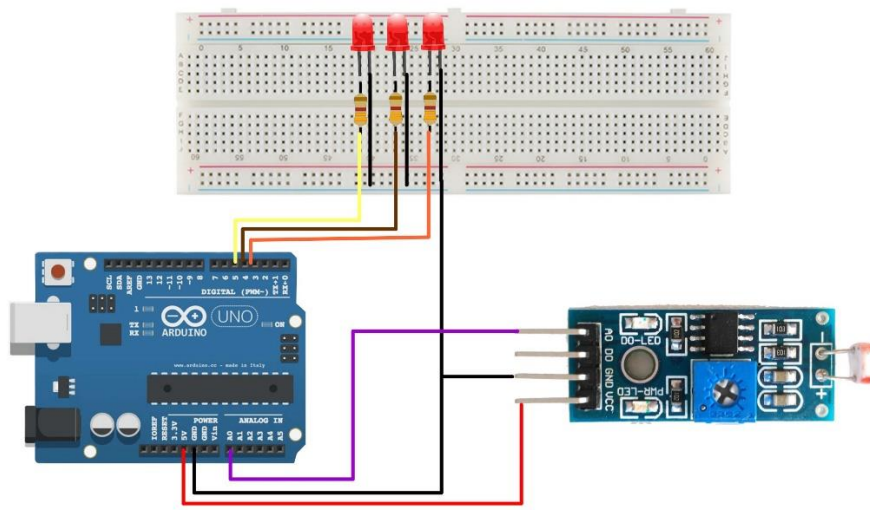
The reading is shown on the Serial Monitor when the LDR sensor module is kept in a room with very little light /no light.

**Experiment 4:** According to the LDR readings, write a program to turn ON & OFF multiple LEDs.

### Components Required

- 1 × Breadboard
- 1 × Arduino Uno R3
- 1 × LDR Module
- 3 × LED
- 3 × 330Ω Resistor
- 10 × Jumper

### Circuit Connections



### Arduino Code

```
int redLED1 = 3;
int redLED2 = 4;
int redLED3 = 5;

void setup()
{
  pinMode(A0, INPUT);
  pinMode(redLED1, OUTPUT);
  pinMode(redLED2, OUTPUT);
  pinMode(redLED3, OUTPUT);
  Serial.begin(9600);
}
```

```

void loop()
{
  long LDRReading = analogRead(A0);
  Serial.print("LDR Reading: ");
  Serial.println(LDRReading);
  if(LDRReading >= 700)
  {
    digitalWrite(redLED1, HIGH);
    digitalWrite(redLED2, HIGH);
    digitalWrite(redLED3, HIGH);
  }
  else if (LDRReading >= 350)
  {
    digitalWrite(redLED1, LOW);
    digitalWrite(redLED2, HIGH);
    digitalWrite(redLED3, HIGH);
  }
  else if (LDRReading >= 150)
  {
    digitalWrite(redLED1, LOW);
    digitalWrite(redLED2, LOW);
    digitalWrite(redLED3, HIGH);
  }
  else
  {
    digitalWrite(redLED1, LOW);
    digitalWrite(redLED2, LOW);
    digitalWrite(redLED3, LOW);
  }
}

```

## Result

LEDs are turning ON and OFF according to LDR readings.

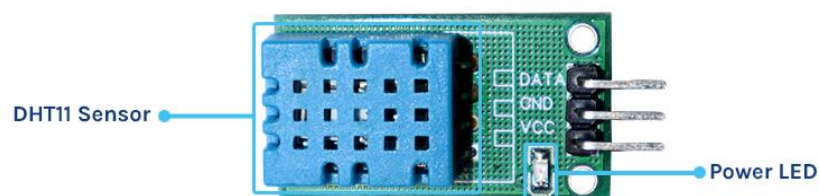
**Experiment 5:** To interface the DHT11 sensor with Arduino, write a program to print temperature and humidity readings on a serial monitor.

### Components Required

- 1 × Arduino Uno R3
- 1 × DHT11 Module
- 3 × Jumper

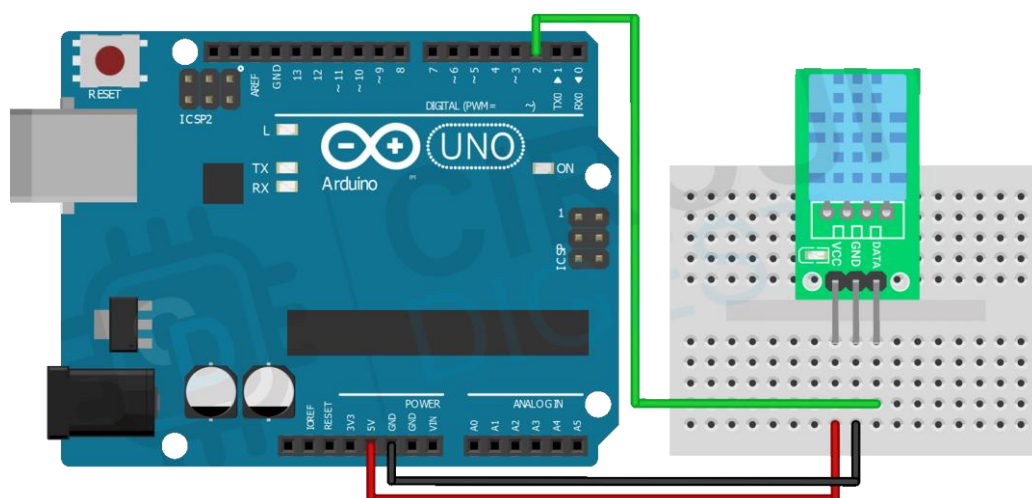
DHT11 Module features a temperature & humidity sensor complex with a calibrated digital signal output. The exclusive digital signal acquisition technique and temperature & humidity sensing technology ensure high reliability and long-term stability. This sensor includes an NTC for temperature measurement and a resistive-type humidity measurement component for humidity measurement. These are connected to a high-performance 8-bit microcontroller, offering excellent quality, fast response, anti-interference ability, and cost-effectiveness.

The DHT11 module has a total of 3 pins. Two are for power, and one is for communication. The pinout of a DHT11 Sensor module is as follows:



### Circuit Diagram for Interfacing DHT11 Sensor with Arduino

Now that we have completely understood how a DHT11 Sensor works, we can connect all the required wires to Arduino and write the code to get all the data from the sensor. The following image shows the circuit diagram for interfacing the DHT11 sensor module with Arduino.



Connections are pretty simple and only require three wires. Connect the module's VCC and GND to the Arduino's 5V and GND pins. Then, connect the DATA pin to the Arduino's digital pin 2. We communicate with DHT11 through this pin.

## Arduino Code

```
#include <dht11.h>

#define DHT11PIN 2

dht11 DHT11;

void setup()

{

    Serial.begin(9600);

}

void loop()

{

    Serial.println();

    int chk = DHT11.read(DHT11PIN);

    Serial.print("Humidity (%): ");

    Serial.println((float)DHT11.humidity, 2);

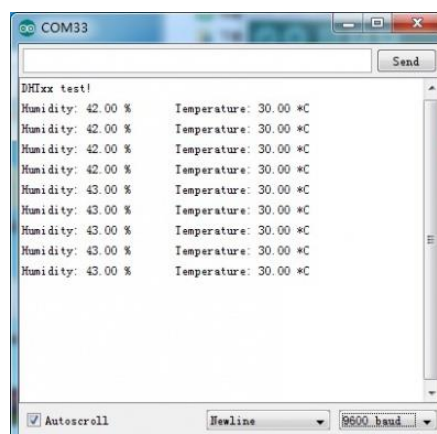
    Serial.print("Temperature (C): ");

    Serial.println((float)DHT11.temperature, 2);

    delay(2000);

}
```

## Result

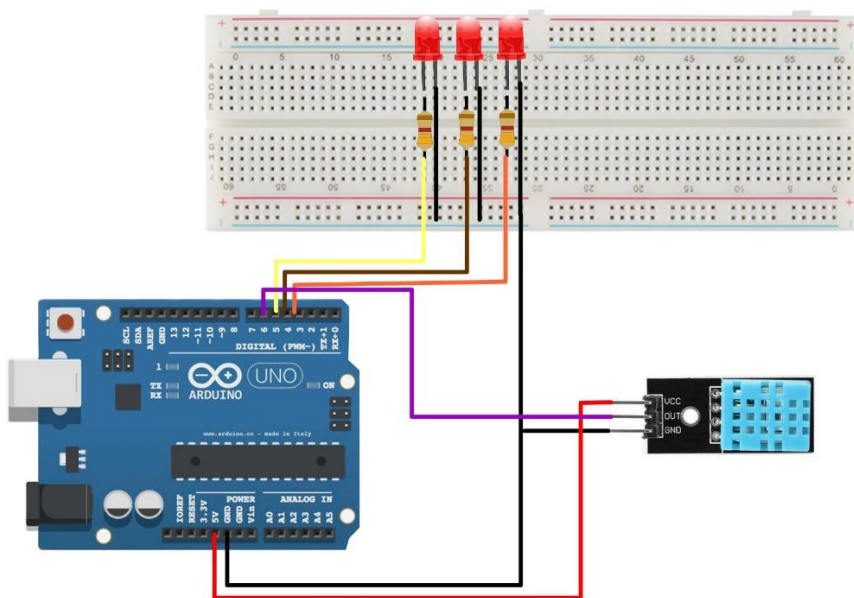


**Experiment 6:** According to the DHT readings, write a program to turn ON & OFF multiple LEDs.

### Components Required

1 × Breadboard  
1 × Arduino Uno R3  
1 × DHT11 Module  
3 × LED  
3 × 330Ω Resistor  
10 × Jumper

### Circuit Connections



### Arduino Code

```
#include <DFRobot_DHT11.h>
```

```
DFRobot_DHT11 DHT;
```

```
#define DHT11_PIN 6
```

```
int redLED1 = 3;
```

```
int redLED2 = 4;
```

```
int redLED3 = 5;
```



```
void setup()

{

Serial.begin(9600);

pinMode(DHT11_PIN, INPUT);

pinMode(redLED1, OUTPUT);

pinMode(redLED2, OUTPUT);

pinMode(redLED3, OUTPUT);

}

void loop()

{

DHT.read(DHT11_PIN);

Serial.print("temperature:");

Serial.print(DHT.temperature);

Serial.print(", humidity:");

Serial.println(DHT.humidity);

if(DHT.humidity >= 80 || DHT.temperature >= 30)

{

digitalWrite(redLED1, HIGH);

digitalWrite(redLED2, HIGH);

digitalWrite(redLED3, HIGH);

}

else if(DHT.humidity >= 70 || DHT.temperature

< 26)

{
```

```
digitalWrite(redLED1, LOW);  
  
digitalWrite(redLED2, HIGH);  
  
digitalWrite(redLED3, HIGH);  
  
}  
  
else  
  
{  
  
digitalWrite(redLED1, LOW);  
  
digitalWrite(redLED2, LOW);  
  
digitalWrite(redLED3, HIGH);  
  
}  
  
delay(1000);  
  
}
```

### **Result**

LEDs are turning ON and OFF according to DHT readings.

**Experiment 7:** To interface the HC-05 Bluetooth module with Arduino, write a program to turn ON & OFF multiple LEDs with a smartphone using Bluetooth.

### Components Required

1 × Breadboard  
1 × Arduino Uno R3  
1 × HC-05 Bluetooth Module  
1 × LED  
1 × 220Ω Resistor  
1 × Smartphone with Bluetooth compatibility  
10 × Jumper

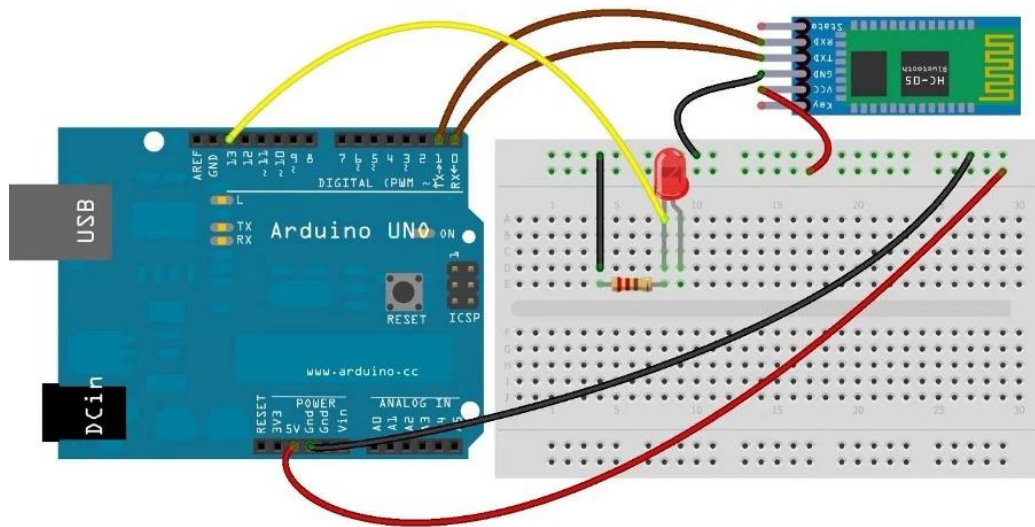
HC-05 is a Bluetooth device for wireless communication with Bluetooth-enabled devices (like smartphones). It communicates with microcontrollers using serial communication (USART). The default settings of the HC-05 Bluetooth module can be changed using specific AT commands.

As the HC-05 Bluetooth module has a 3.3 V level for RX/TX and the microcontroller can detect a 3.3 V level, there is no need to shift the TX voltage level of the HC-05 module. But, we need to turn the transmit voltage level from the microcontroller to the RX of the HC-05 module.

To send data from a smartphone to Arduino via Bluetooth, you will need a Bluetooth module, such as the HC-05, and a smartphone with Bluetooth capabilities. Here is a general overview of the process:

1. Connect the HC-05 Bluetooth module to the Arduino. The module should be connected to the Arduino's serial pins (RX, TX) and powered by the Arduino's voltage supply.
2. Upload a sketch to the Arduino that sends data via the serial connection. You can use Arduino's `Serial.begin()` and `Serial.print()` functions to get data from the HC-05 module.
3. Pair the HC-05 module with your smartphone. Go to the Bluetooth settings on your smartphone and search for available devices. Once the HC-05 is found, pair it with the default password "1234" or "0000."
4. Use software or an application on your smartphone to transmit the data to Arduino. You can use any Bluetooth home automation app from the Play Store.
5. Configure the software or app to transmit data to the HC-05 module.
6. Once everything is set up, the smartphone should be able to send data to the Arduino via the HC-05 Bluetooth module. The data can be displayed on the Serial monitor or used to control other functions on the Arduino.

## Connection Diagram



## Arduino Code

```
char data = 0; //declaring Variable for storing received data

void setup()
{
  Serial.begin(9600); // baud rate for serial data transmission
  pinMode(13, OUTPUT);
}

void loop()
{
  if(Serial.available() > 0) // checks if any data is received
  {
    data = Serial.read(); /*Reading receiving data and storing it into the variable named data*/
    Serial.print(data);
    Serial.print("\n"); //adding space by giving a New line
    if(data == '1')
      digitalWrite(13, HIGH);
    else if(data == '0')
      digitalWrite(13, LOW);
```

```
}  
}
```

### **Code to Note**

In this code, we have changed the state of the LED by sending data to the Bluetooth module. The code for interfacing the Bluetooth is compiled so that we have first declared the variable in which the data will be stored. After that, we gave the baud rate for the serial communication and assigned the mode and pin to the LED in the setup function.

In the loop function, we used the `Serial.available()` to check if any data is entered using the serial monitor. After that, to read the data, if received, we use the function `Serial.read()` and then this data is printed in the serial monitor using the function `Serial.print()`.

Next, we have used the if conditions using the `digitalWrite()` function. If 1 is entered in the serial monitor of COM6, on which the Bluetooth is connected, turn on the LED, and if 0 is entered in the serial monitor of COM6, the LED will turn off.

Note: There might be different ports for each computer connected to the Bluetooth module. Here, in our case, it is COM6.

### **Result**

LEDs can be controlled by Bluetooth application installed in the smartphone.

**Experiment 8:** Write a program to send sensor data to a smartphone using the HC-05 Bluetooth module with Arduino.

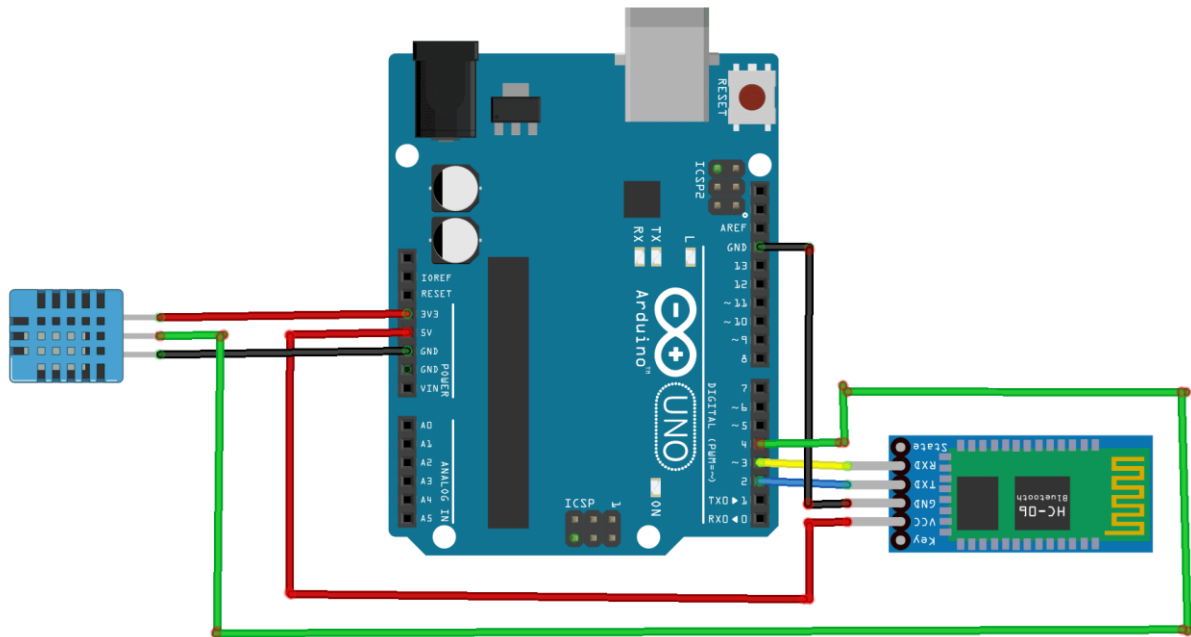
### **Components Required**

1 × Breadboard  
1 × Arduino Uno R3  
1 × HC-05 Bluetooth Module  
1 × LED  
1 × 220Ω Resistor  
1 × Smartphone with Bluetooth compatibility  
10 × Jumper

To send data from an Arduino to a smartphone via Bluetooth, you will need a Bluetooth module, such as the HC-05, and a smartphone with Bluetooth capabilities. Here is a general overview of the process:

1. Connect the HC-05 Bluetooth module to the Arduino. The module should be connected to the Arduino's serial pins (RX, TX) and powered by the Arduino's voltage supply.
2. Upload a sketch to the Arduino that sends data via the serial connection. You can use the `Serial.begin()` and `Serial.print()` functions in Arduino to send data from the Arduino to the HC-05 module.
3. Pair the HC-05 module with your smartphone. Go to the Bluetooth settings on your smartphone and search for available devices. Once the HC-05 is found, pair it with the default password "1234" or "0000."
4. Use software or an application on your smartphone that can receive the data sent by the Arduino. One of the most common ones is the "Serial Bluetooth Terminal." This app allows you to see the data transmitted by the Arduino.
5. Configure the software or app to receive data from the HC-05 module.
6. Once everything is set up, the Arduino should be able to send data to the smartphone via the HC-05 Bluetooth module. The data can be displayed on the smartphone's screen or used to control other functions on the smartphone.

## Connection Diagram



## Arduino Code

```
#include <SoftwareSerial.h>

#include <dht11.h>

#define DHT11PIN 4      // broche DATA -> broche 4

dht11 DHT11;

SoftwareSerial hc05(2,3);

void setup()
{
  hc05.begin(9600);
}

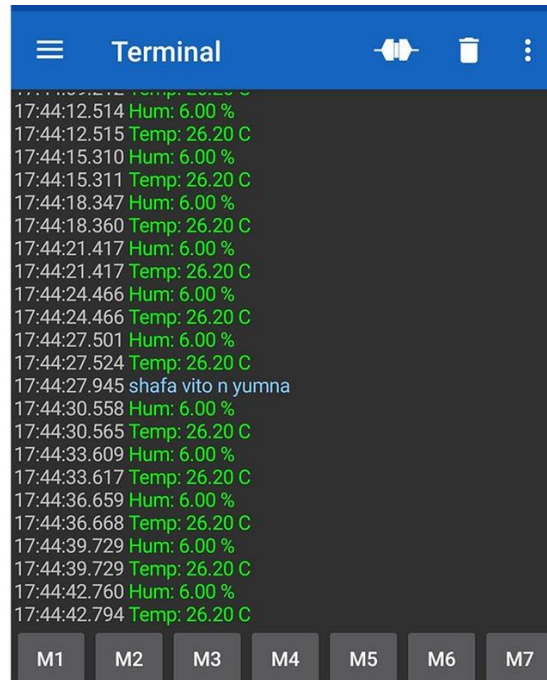
void loop()
{
  DHT11.read(DHT11PIN);
```

```
delay(1000);

hc05.print((float)DHT11.temperature);    //sends the temperature value to the smartphone

}
```

## Result

A screenshot of a mobile application interface titled "Terminal". The interface has a blue header bar with a hamburger menu icon on the left, the title "Terminal" in the center, and three icons (a plug, a trash can, and a vertical ellipsis) on the right. The main area is a dark grey terminal window displaying a list of sensor readings. Each line shows a timestamp, humidity, and temperature. The humidity is consistently 6.00%, and the temperature is consistently 26.20 C. One line contains the text "shafa vito n yumna". At the bottom of the screen, there is a row of seven grey buttons labeled M1, M2, M3, M4, M5, M6, and M7.

```
17:44:12.512 Temp: 26.20 C
17:44:12.514 Hum: 6.00 %
17:44:12.515 Temp: 26.20 C
17:44:15.310 Hum: 6.00 %
17:44:15.311 Temp: 26.20 C
17:44:18.347 Hum: 6.00 %
17:44:18.360 Temp: 26.20 C
17:44:21.417 Hum: 6.00 %
17:44:21.417 Temp: 26.20 C
17:44:24.466 Hum: 6.00 %
17:44:24.466 Temp: 26.20 C
17:44:27.501 Hum: 6.00 %
17:44:27.524 Temp: 26.20 C
17:44:27.945 shafa vito n yumna
17:44:30.558 Hum: 6.00 %
17:44:30.565 Temp: 26.20 C
17:44:33.609 Hum: 6.00 %
17:44:33.617 Temp: 26.20 C
17:44:36.659 Hum: 6.00 %
17:44:36.668 Temp: 26.20 C
17:44:39.729 Hum: 6.00 %
17:44:39.729 Temp: 26.20 C
17:44:42.760 Hum: 6.00 %
17:44:42.794 Temp: 26.20 C
```

M1 M2 M3 M4 M5 M6 M7





## Arduino Code

```
#include "BluetoothSerial.h"

BluetoothSerial SerialBT;

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

  SerialBT.begin("ESP32test");    //Bluetooth device name

  Serial.println("The device started, now you can pair it with Bluetooth!");
}

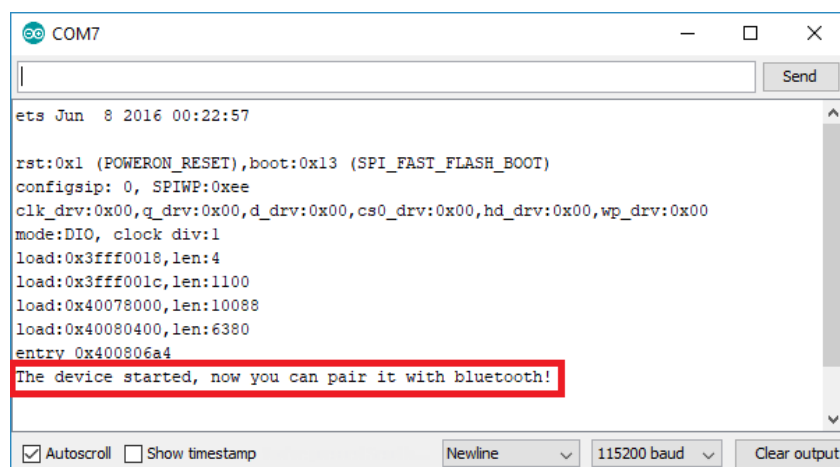
void loop() {
  if (Serial.available())
  {
    SerialBT.write(Serial.read());
  }

  if (SerialBT.available())
  {
    Serial.write(SerialBT.read());
  }

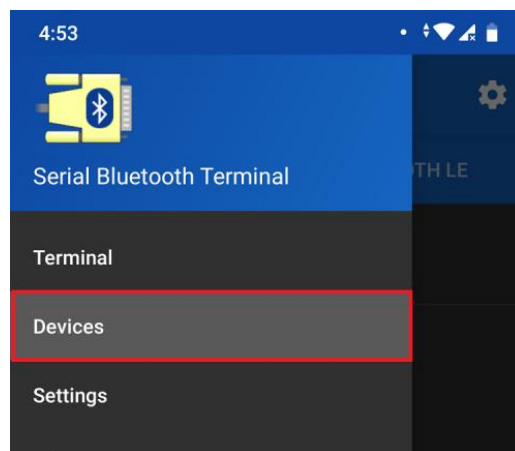
  delay(20);
}
```

Upload the previous code to the ESP32. Make sure you have the right board and COM port selected. After uploading the code, open the Serial Monitor at a baud rate 115200. Press the ESP32 Enable button.

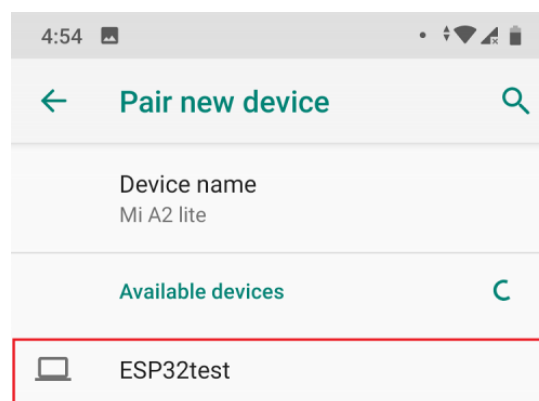
## Result



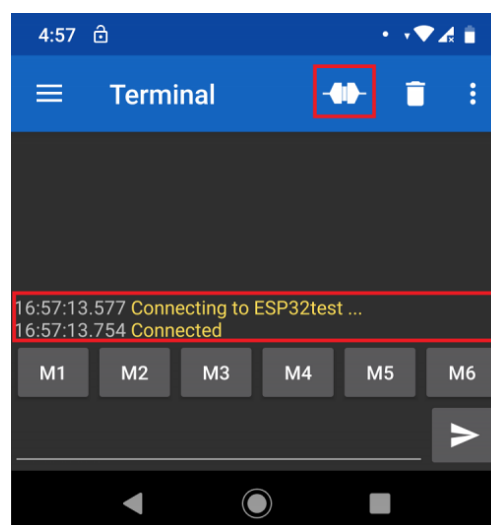
Go to your smartphone and open the “Serial Bluetooth Terminal” app. Make sure you’ve enabled your smartphone’s Bluetooth. You must pair a new device to connect to the ESP32 for the first time. Go to Devices.



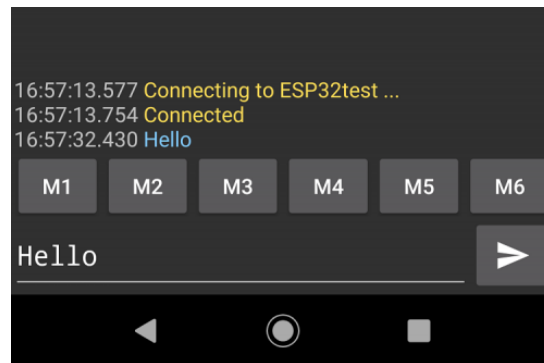
Click the settings icon, and select **Pair new device**. You should get a list of the available Bluetooth devices, including the **ESP32 test**. Pair with the **ESP32 test**.



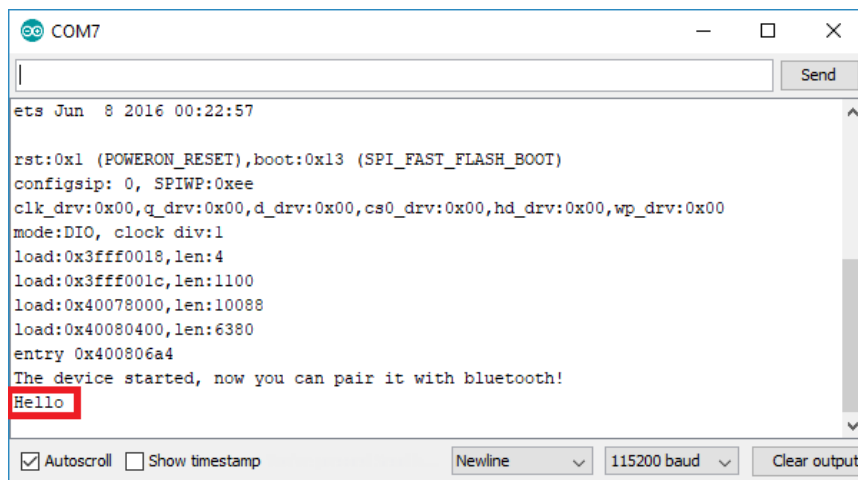
Then, go back to the Serial Bluetooth Terminal. Click the icon at the top to connect to the ESP32. You should get a “**Connected**” message.



After that, type something in the Serial Bluetooth Terminal app. For example, “Hello”.



You should instantly receive that message in the Arduino IDE Serial Monitor.

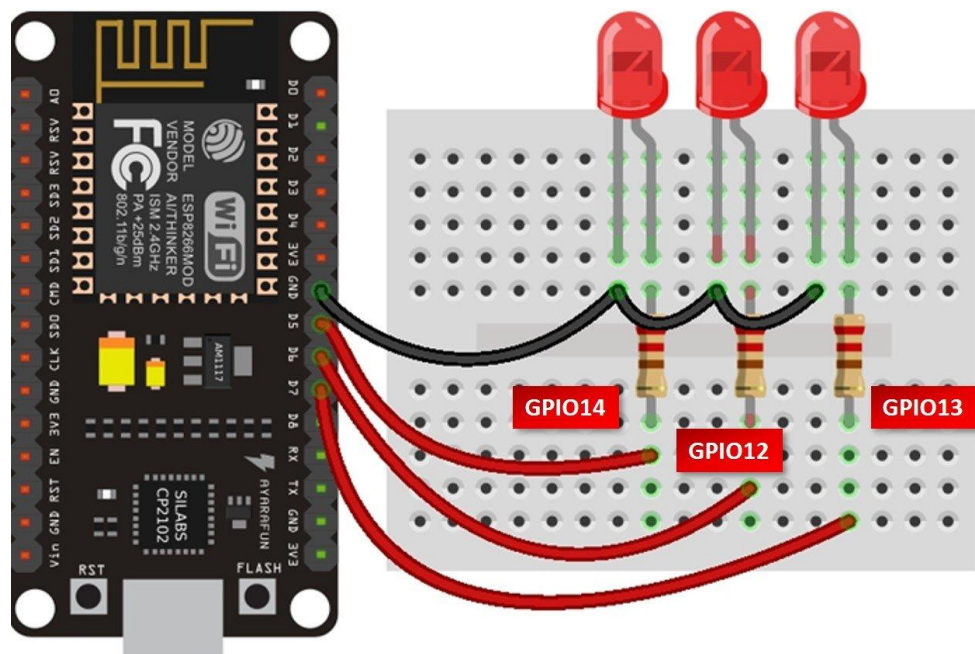


**Experiment 10:** Write a program to control multiple LEDs using ESP32 and mobile devices with the help of MQTT Protocol.

### Components Required

1 × Breadboard  
1 × ESP32  
3 × LED  
3 × 330Ω Resistor  
10 × Jumper

### Connection Diagram



### ESP32 Code

```
#include <WiFi.h>
#include <PubSubClient.h>

// WiFi
const char *ssid = "xxxxx";           // Enter your Wi-Fi name
const char *password = "xxxxx";       // Enter Wi-Fi password
```

**// MQTT Broker**

const char \*mqtt\_broker = "Broker Address";

const char \*topic = "test/esp32";

const char \*mqtt\_username = "username";

const char \*mqtt\_password = "password";

const int mqtt\_port = 1883;

WiFiClient espClient;

PubSubClient client(espClient);

void setup()

{

Serial.begin(115200);

**// Connecting to a Wi-Fi network**

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED)

{

delay(500);

Serial.println("Connecting to WiFi..");

}

Serial.println("Connected to the Wi-Fi network");

**//connecting to a mqtt broker**

client.setServer(mqtt\_broker, mqtt\_port);

client.setCallback(callback);

while (!client.connected())

{

String client\_id = "esp32-client-";

client\_id += String(WiFi.macAddress());

Serial.printf("The client %s connects to the public MQTT broker\n", client\_id.c\_str());

if (client.connect(client\_id.c\_str(), mqtt\_username, mqtt\_password))

{

Serial.println("Public EMQX MQTT broker connected");

```

}
else
{
Serial.print("failed with state ");
Serial.print(client.state());
delay(2000);
}
}

// Publish and subscribe
client.publish(topic, "Hi, I'm ESP32 ^^");
client.subscribe(topic);
}

void callback(char *topic, byte *payload, unsigned int length) {
Serial.print("Message arrived in topic: ");
Serial.println(topic);
Serial.print("Message:");
for (int i = 0; i < length; i++) {
Serial.print((char) payload[i]);
}
Serial.println();
Serial.println("-----");
}

void loop()
{
client.loop();
}

```

**Note: Write the necessary conditions for LED in the loop section.**

**Experiment 11:** Familiarisation with BLYNK Cloud Platform to set up BLYNK Cloud dashboard and control multiple LEDs using a mobile device using ESP32.

### **Components Required**

1 × Breadboard  
1 × ESP32  
3 × LED  
3 × 330Ω Resistor  
10 × Jumper

BLYNK is the most famous app for IoT-related projects. Here, we can receive data, send data and give commands through the Blynk cloud. We can create very cool dashboards using our smartphone, tablet, or computer. This dashboard lets us get sensor values, control electrical devices, etc. It depends on your project.

Also, we can use the basic widgets in this app for free, and we need to buy widgets for advanced projects. It depends on your requirements. But these widgets are more than enough for educational projects. You can download it from Play Store or App Store if you are a mobile or tablet user. Otherwise, you can visit the Blynk website if you are a PC user. Then, create a new account for the Blynk app.