<u>Guru Gobind Singh Indraprastha University</u>
<u>Shahdara, Delhi</u>

University School of Automation & Robotics (USAR)

IOT Manual

For

IIOT STUDENTS

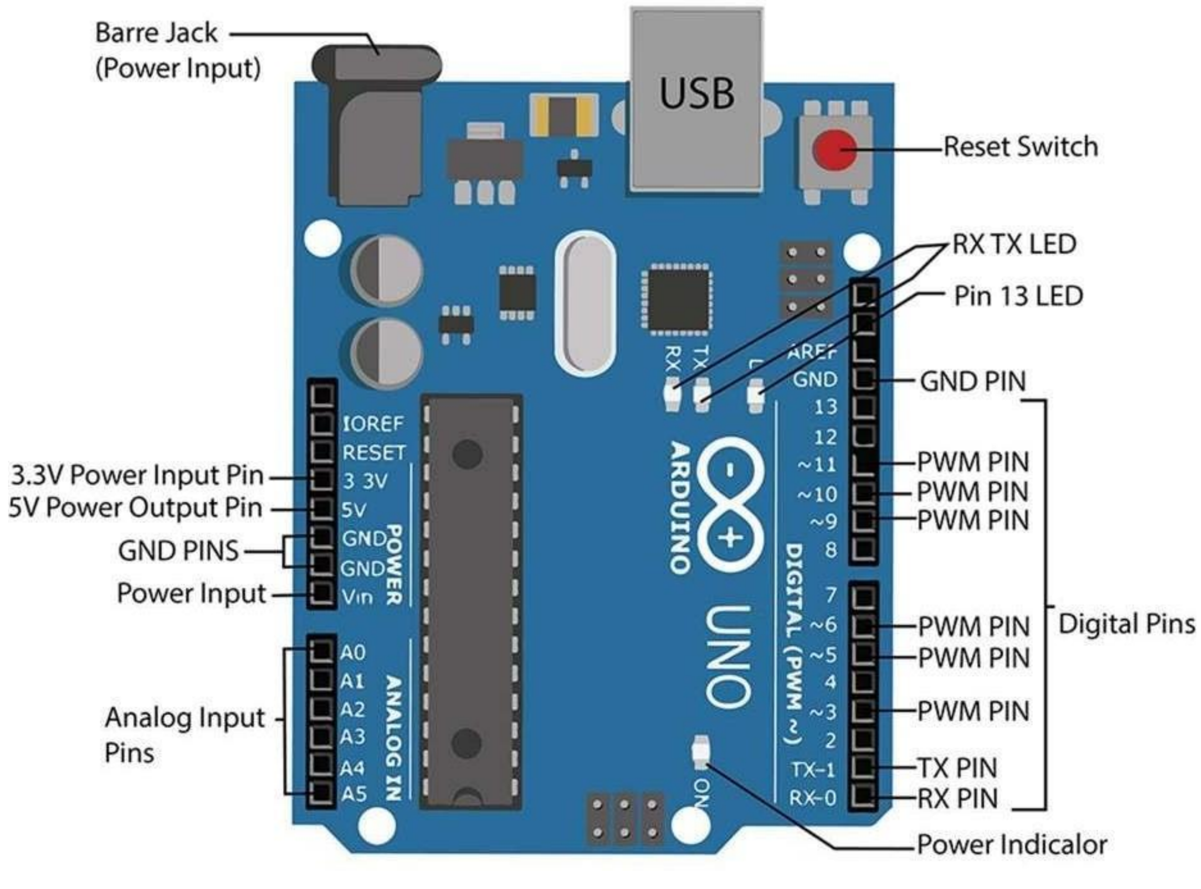
Experiment No. 1
Familiarization of Arduino IDE

Aim:

a) Familiarization of Arduino IDE

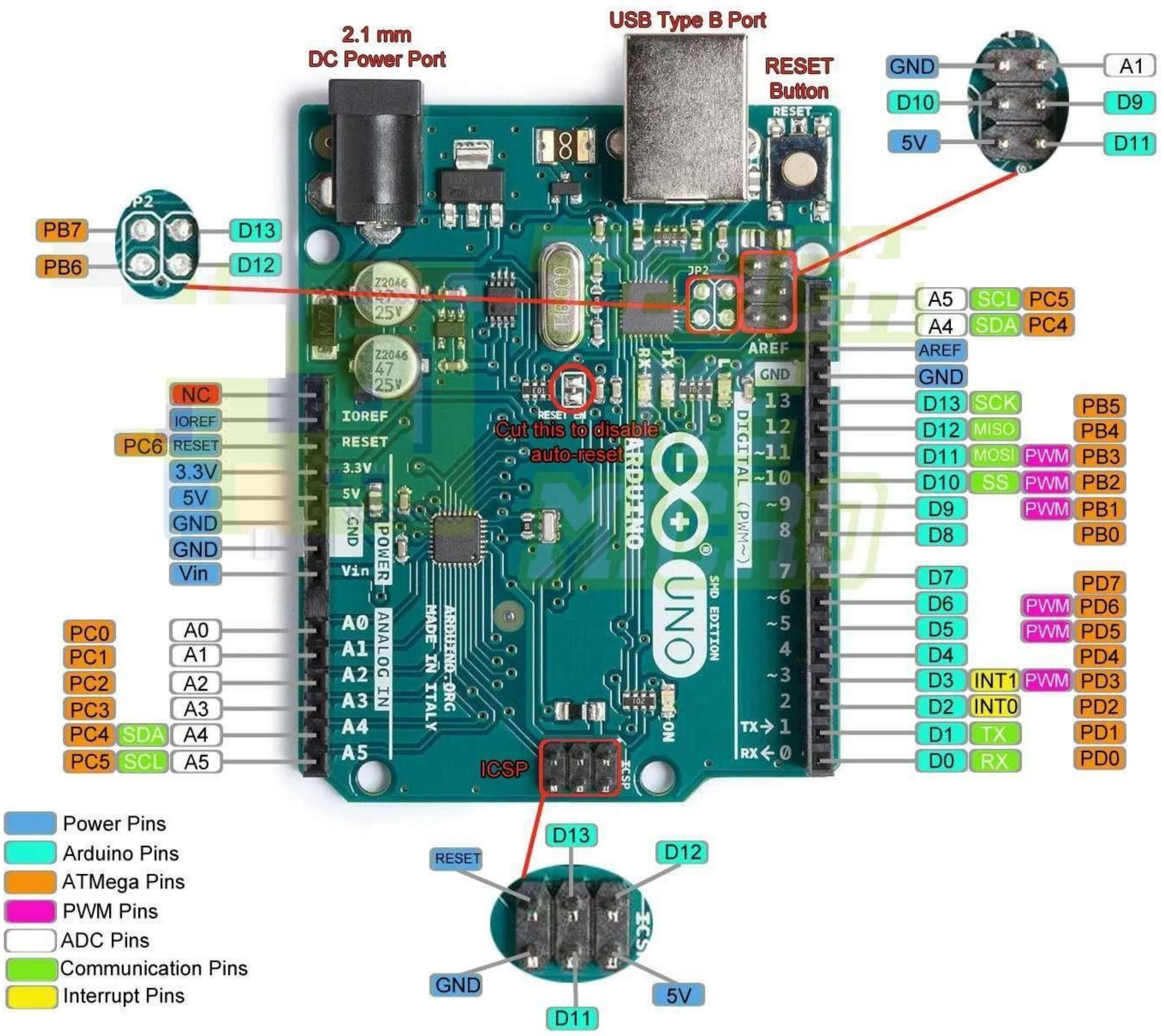
Theory: Introduction to Arduino board

Arduino is an open-source prototyping platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.



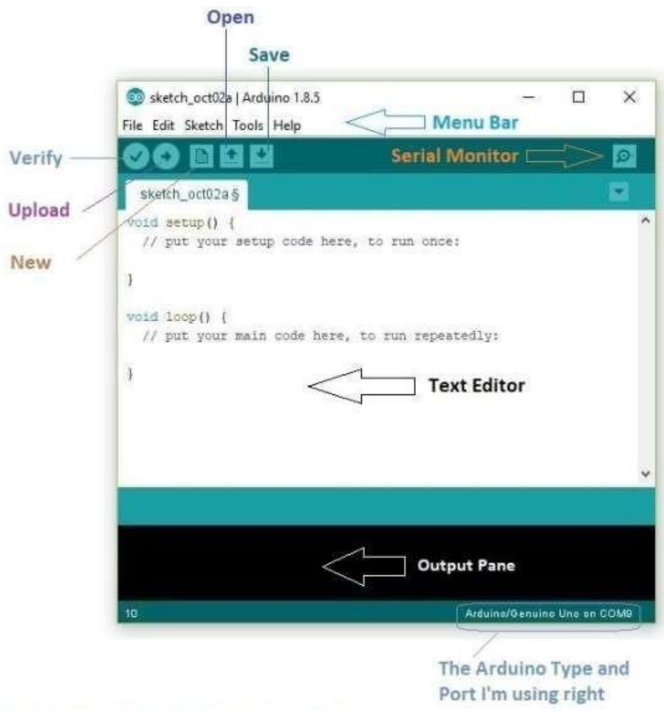
Arduino Uno

Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems which are inexpensive , cross-platform , Simple, clear programming environment , Open source and extensible hardware etc.



Pin Diagram of Arduino Uno

The basic structure of the Arduino programming language is fairly simple and runs in at least two parts. These two required parts, or functions, enclose blocks of statements. Where setup() is the preparation, loop() is the execution. Both functions are required for the program to work.


Open
Save
sketch_oct02a | Arduino 1.8.5
File Edit Sketch Tools Help
Menu Bar
Verify
Serial Monitor
Upload
New
sketch_oct02a §
Text Editor
Output Pane
10
Arduino/Genuino Uno on COM9
The Arduino Type and Port I'm using right

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

The setup function should follow the declaration of any variables at the very beginning of the program. It is the first function to run in the program, is run only once, and is used to set **pinMode** or initialize serial communication.

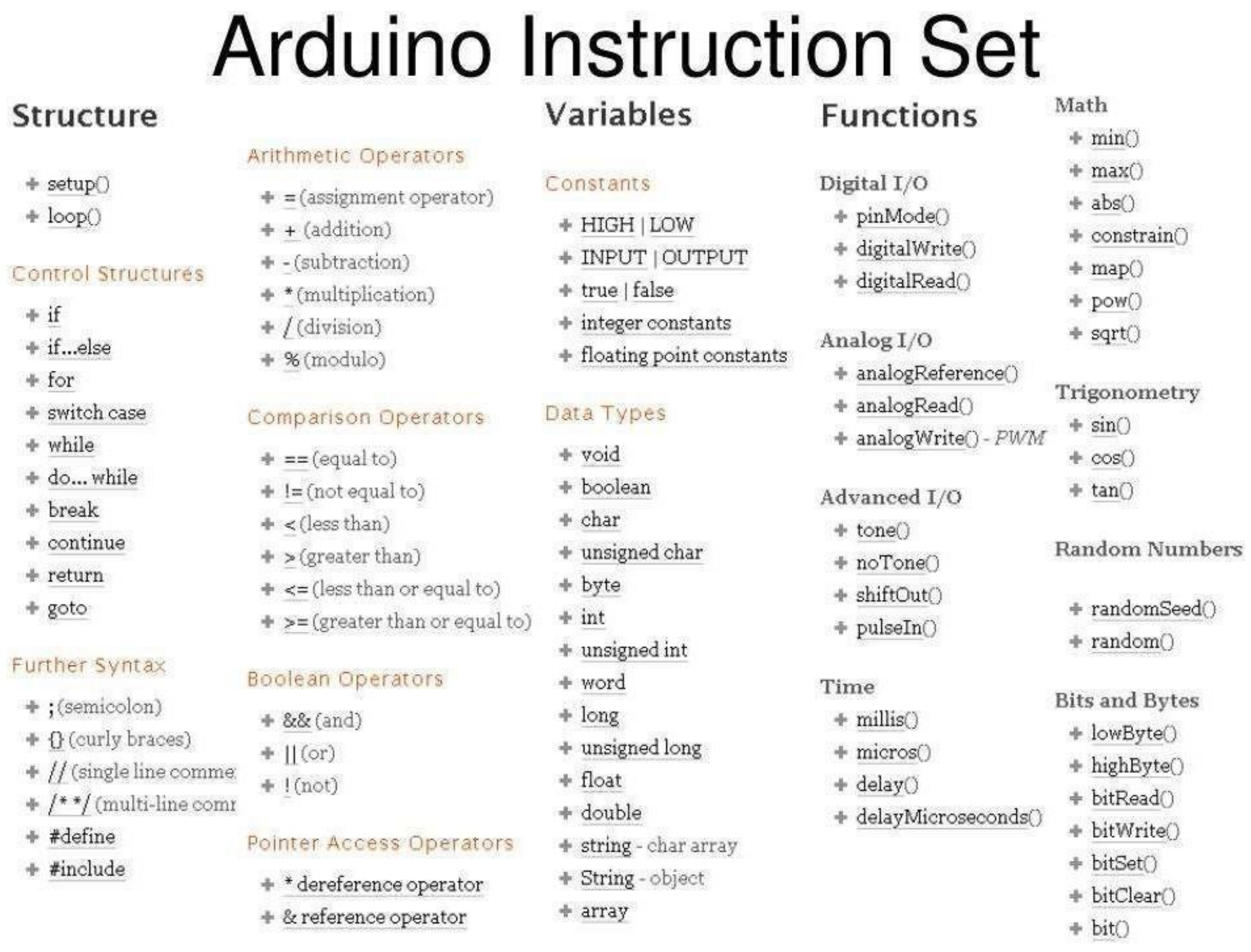
The loop function follows next and includes the code to be executed continuously-reading inputs, triggering outputs etc. This function is the core of all Arduino programs and does the bulk of the work. Curly braces (**{}**)define the beginning and the end of function blocks and statement blocks such as the void loop() function and the for and if statements.

Programming using Arduino IDE

Step 1 − Download Arduino IDE Software: You can get different versions of Arduino IDE from the Download page on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.

Step 2 − Power up your board: The Arduino Uno automatically draw power from either, the USB connection to the computer or an external power supply. Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

Step 4 − Launch Arduino IDE



Arduino Instruction Set

Structure

- setup()
- loop()

Control Structures

- if
- if...else
- for
- switch case
- while
- do... while
- break
- continue
- return
- goto

Further Syntax

- ; (semicolon)
- {} (curly braces)
- // (single line comme
- /* */ (multi-line comr
- #define
- #include

Arithmetic Operators

- = (assignment operator)
- + (addition)
- - (subtraction)
- * (multiplication)
- / (division)
- % (modulo)

Comparison Operators

- == (equal to)
- != (not equal to)
- < (less than)
- > (greater than)
- <= (less than or equal to)
- >= (greater than or equal to)

Boolean Operators

- && (and)
- || (or)
- ! (not)

Pointer Access Operators

- * dereference operator
- & reference operator

Variables

Constants

- HIGH | LOW
- INPUT | OUTPUT
- true | false
- integer constants
- floating point constants

Data Types

- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int
- word
- long
- unsigned long
- float
- double
- string - char array
- String - object
- array

Functions

Digital I/O

- pinMode()
- digitalWrite()
- digitalRead()

Analog I/O

- analogReference()
- analogRead()
- analogWrite() - *PWM*

Advanced I/O

- tone()
- noTone()
- shiftOut()
- pulseIn()

Time

- millis()
- micros()
- delay()
- delayMicroseconds()

Math

- min()
- max()
- abs()
- constrain()
- map()
- pow()
- sqrt()

Trigonometry

- sin()
- cos()
- tan()

Random Numbers

- randomSeed()
- random()

Bits and Bytes

- lowByte()
- highByte()
- bitRead()
- bitWrite()
- bitSet()
- bitClear()
- bit()

Step 5 − Open your first project: Once the software starts, you have two options −

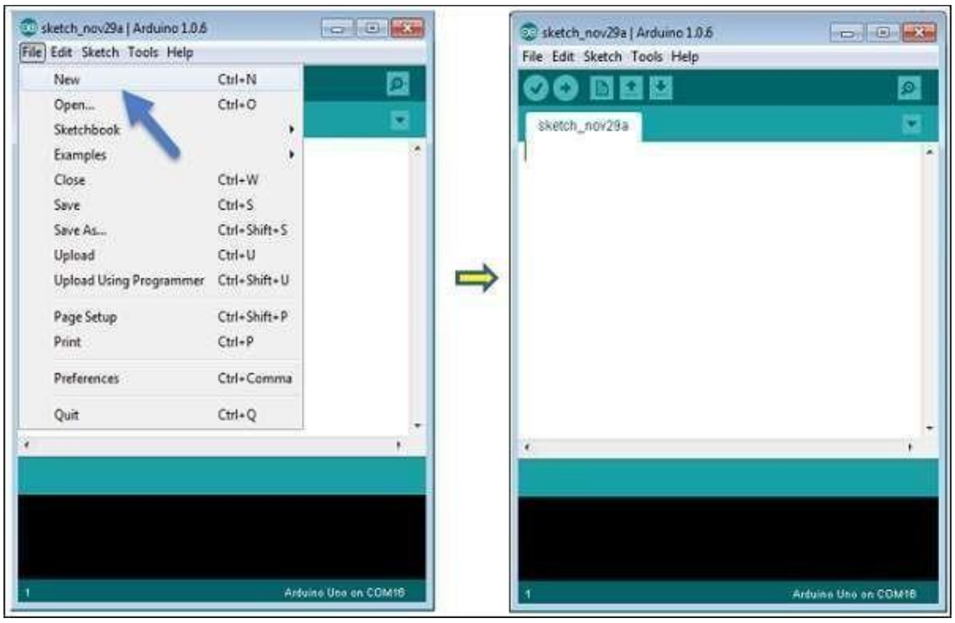
- Create a new project.
- Open an existing project

example. To create a new project,

select File → **New**.

To open an existing project example, select File → Example → Basics → Blink.

Here, we are selecting just one of the examples with the name **Blink**. It turns the LED on and off with some time delay. You can select any other example from the list.

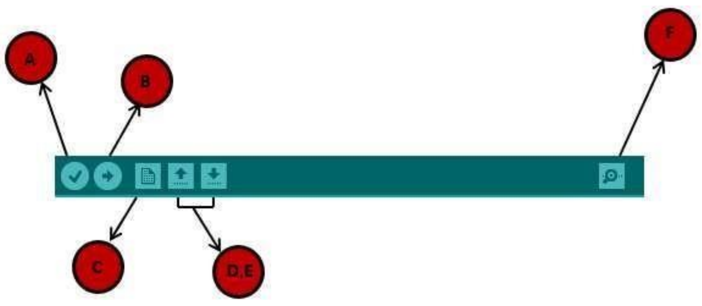


Step 6 − Select your Arduino board: To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.

Go to Tools → Board and select your board

Step 7 − Select your serial port: Select the serial device of the Arduino board. Go to **Tools → Serial Port** menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.

Step 8 − Upload the program to your board: Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar. Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.



A − Check if there is any compilation error.	**B** − Upload a program to the Arduino board.
C − Shortcut to create a new sketch.	**D** − Open one of the example sketch.
E − Used to save your sketch.	**F** − Serial monitor for serial data transf

Experiment :-2

Aim:- Experiment to blink an LED using an Arduino board:

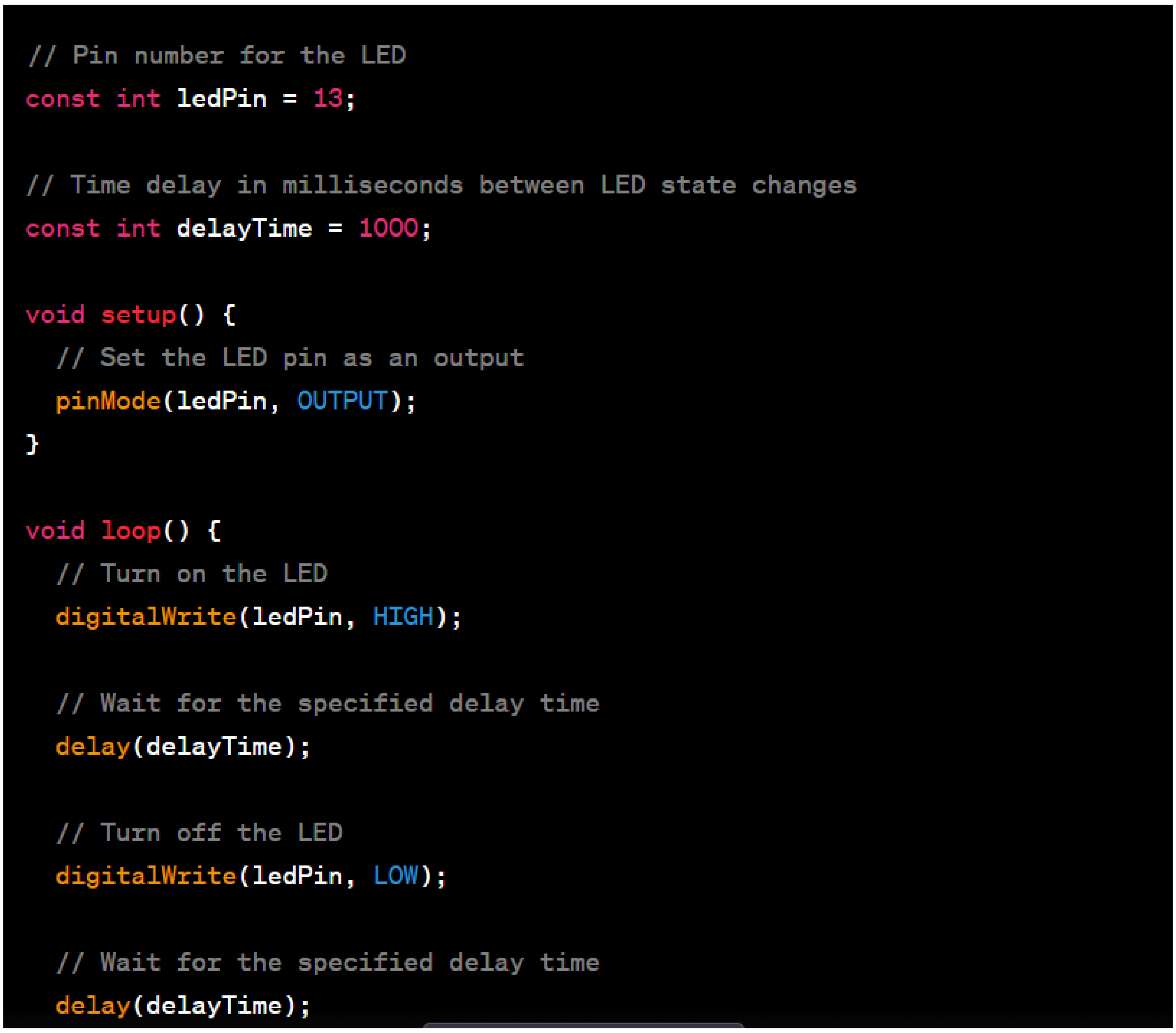
Components needed:

1. Arduino board (e.g., Arduino Uno)
2. LED (any color)
3. Resistor (220-470 ohms)
4. Breadboard
5. Jumper wires

Circuit setup:

1. Connect the longer leg (anode) of the LED to digital pin 13 on the Arduino using a resistor.
2. Connect the shorter leg (cathode) of the LED to the ground (GND) pin on the Arduino.

Arduino code:



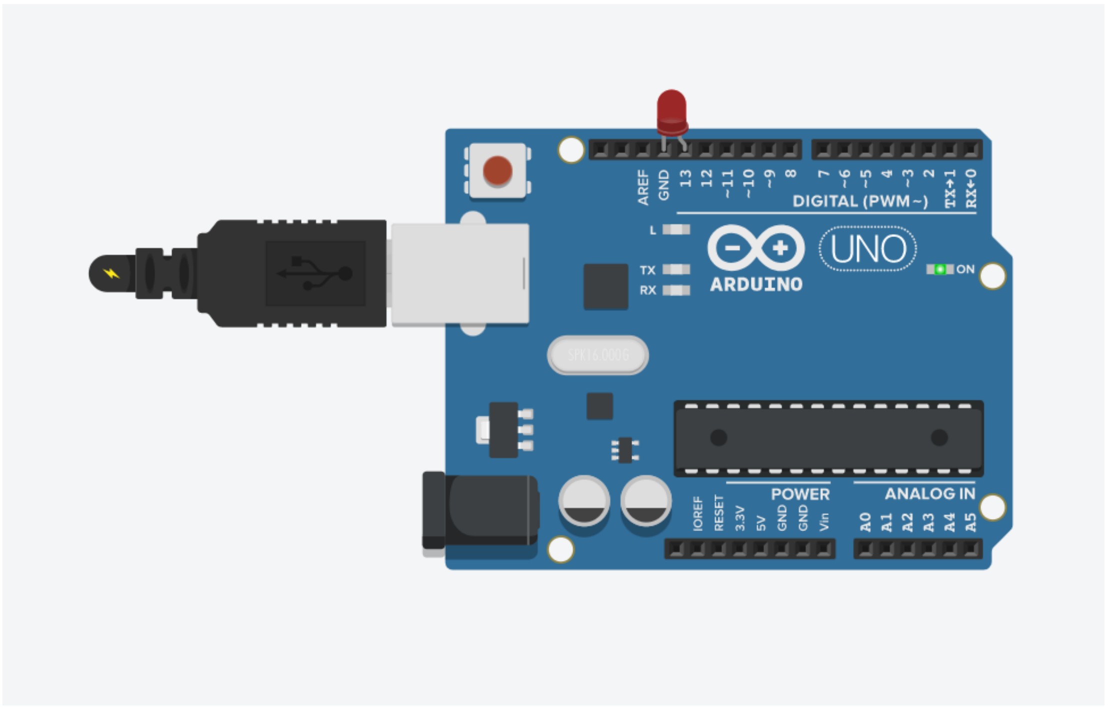
```
// Pin number for the LED
const int ledPin = 13;

// Time delay in milliseconds between LED state changes
const int delayTime = 1000;

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

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

  // Wait for the specified delay time
  delay(delayTime);

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

  // Wait for the specified delay time
  delay(delayTime);
```

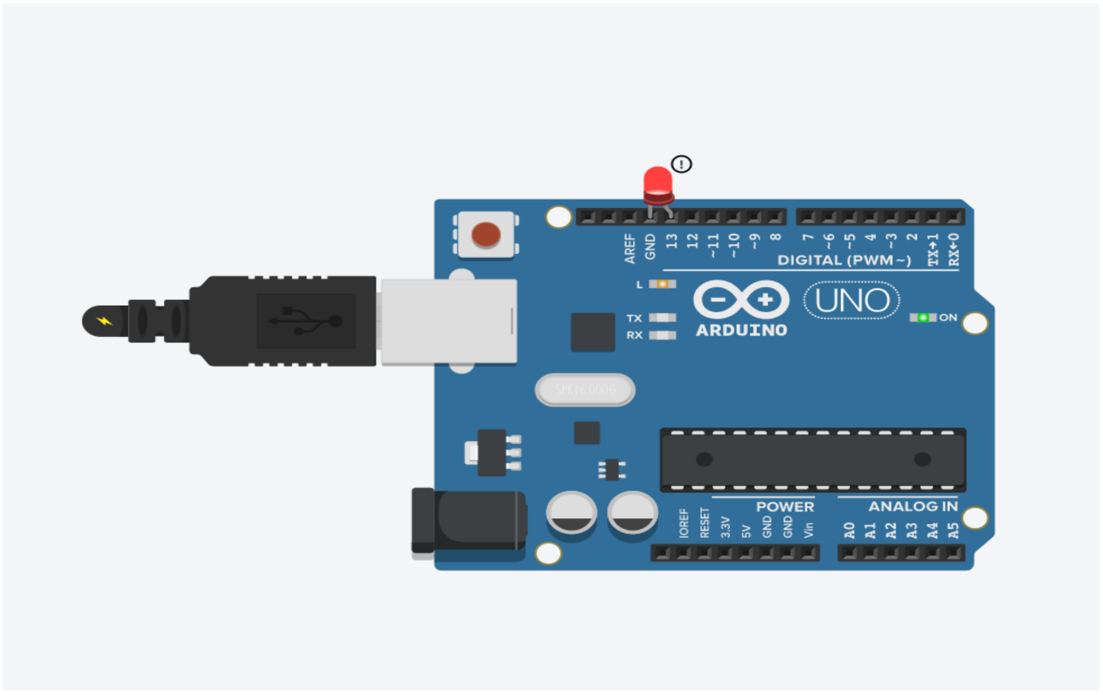
Explanation:

1. We define a constant variable `ledPin` with the value 13 to represent the pin to which the LED is connected.
2. Another constant variable `delayTime` is set to the desired delay (in milliseconds) between LED state changes (on and off).

3. In the `setup()` function, we set the `ledPin` as an output using the `pinMode()` function.
4. The `loop()` function is where the LED blinking logic resides.
5. We first turn on the LED by setting the `ledPin` to a HIGH state using `digitalWrite()`.
6. We then wait for the specified delay time using the `delay()` function.
7. Next, we turn off the LED by setting the `ledPin` to a LOW state.
8. We wait for the specified delay time again.
9. The `loop()` function continues to run indefinitely, resulting in the LED blinking on and off repeatedly.

Upload the code to your Arduino board, and you should see the LED connected to pin 13 blinking at the specified interval.





Experiment-3

AIM : Introduction to various types of Sensors and Actuators.

Description:

We live in a World of Sensors. You can find different types of Sensors in our homes, offices, cars etc. working to make our lives easier by turning on the lights by detecting our presence, adjusting the room temperature, detect smoke or fire, make us delicious coffee, open garage doors as soon as our car is near the door and many other tasks.All these and many other automation tasks are possible because of Sensors. Before going in to the details of What is a Sensor, What are the Different Types of Sensors and Applications of these different types of Sensors, we will first take a look at a simple example of an automated system, which is possible because of Sensors (and many other components as well).An Automatic Flight Control System consists of several sensors for various tasks like speed control, height monitoring, position tracking, status of doors, obstacle detection, fuel level, maneuvering and many more. A Computer takes data from all these sensors and processes them by comparing them with pre-designed values.The computer then provides control signals to different parts like engines, flaps, rudders, motors etc. that help in a smooth flight. The combination of Sensors, Computers and Mechanics makes it possible to run the plane in Autopilot Mode.

Different Types of Sensors

The following is a list of different types of sensors that are commonly used in various applications. All these sensors are used for measuring one of the physical properties like Temperature, Resistance, Capacitance, Conduction, Heat Transfer etc.

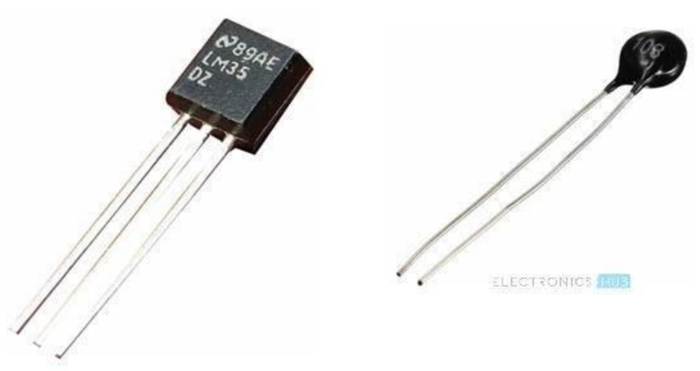
1. Temperature Sensor
2. Proximity Sensor
3. Accelerometer
4. IR Sensor (Infrared Sensor)
5. Pressure Sensor
6. Light Sensor
7. Ultrasonic Sensor
8. Smoke, Gas and Alcohol Sensor
9. Touch Sensor
10. Color Sensor
11. Humidity Sensor
12. Position Sensor
13. Magnetic Sensor (Hall Effect Sensor)
14. Microphone (Sound Sensor)
15. Tilt Sensor

Flow and Level Sensor

1. PIR Sensor
2. Touch Sensor
3. Strain and Weight Sensor

We will see about few of the above-mentioned sensors in brief. More information about the sensors will be added subsequently. A list of projects using the above sensors is given at the end of the page.

Temperature Sensor: One of the most common and most popular sensors is the Temperature Sensor. A Temperature Sensor, as the name suggests, senses the temperature i.e., it measures the changes in the temperature.



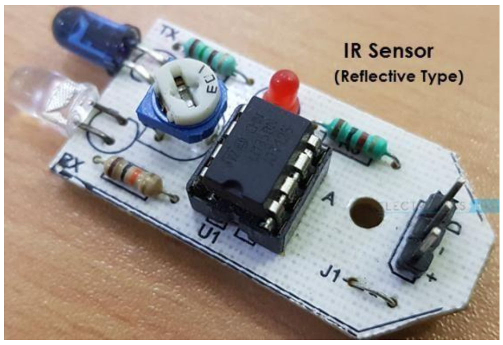
LM35 - Temperature Sensor IC 10KΩ NTC Thermistor

There are different types of Temperature Sensors like Temperature Sensor ICs (like LM35, DS18B20), Thermistors, Thermocouples, RTD (Resistive Temperature Devices), etc.Temperature Sensors can be analog or digital. In an Analog Temperature Sensor, the changes in the Temperature correspond to change in its physical property like resistance or voltage. LM35 is a classic Analog Temperature Sensor.Coming to the Digital Temperature Sensor, the output is a discrete digital value (usually, some numerical data after converting analog value to digital value). DS18B20 is a simple Digital Temperature Sensor.Temperature Sensors are used everywhere like computers, mobile phones, automobiles, air conditioning systems, industries etc.

Proximity Sensors:A Proximity Sensor is a non-contact type sensor that detects the presence of an object. Proximity Sensors can be implemented using different techniques like Optical (like Infrared or Laser), Sound (Ultrasonic), Magnetic (Hall Effect), Capacitive, etc.

Some of the applications of Proximity Sensors are Mobile Phones, Cars (Parking Sensors), industries (object alignment), Ground Proximity in Aircrafts, etc.

Infrared Sensor (IR Sensor):IR Sensors or Infrared Sensor are light based sensor that are used in various applications like Proximity and Object Detection. IR Sensors are used as proximity sensors in almost all mobile phones.



There are two types of Infrared or IR Sensors: Transmissive Type and Reflective Type. In Transmissive Type IR Sensor, the IR Transmitter (usually an IR LED) and the IR Detector (usually a Photo Diode) are positioned facing each other so that when an object passes between them, the sensor detects the object.The other type of IR Sensor is a Reflective Type IR Sensor. In this, the transmitter and the detector are positioned adjacent to each other facing the object. When an object comes in front of the sensor, the infrared light from the IR Transmitter is reflected from the object and is detected by the IR Receiver and thus the sensor detects the object.Different applications where IR Sensor is implemented are Mobile Phones, Robots, Industrial assembly, automobiles etc.

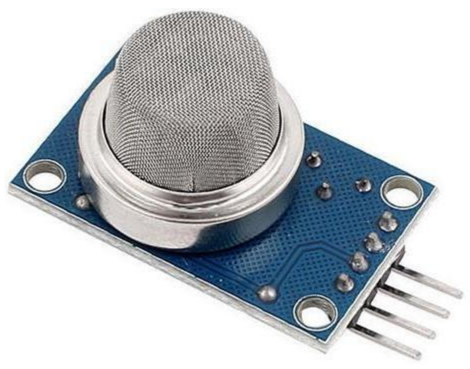
Ultrasonic Sensor: An Ultrasonic Sensor is a non-contact type device that can be used to measure distance as well as velocity of an object. An Ultrasonic Sensor works based on the properties of the sound waves with frequency greater than that of the human audible range.

Using the time of flight of the sound wave, an Ultrasonic Sensor can measure the distance of the object (similar to SONAR). The Doppler Shift property of the sound wave is used to measure the velocity of an object.

Light Sensor:Sometimes also known as Photo Sensors, Light Sensors are one of the important sensors. A simple Light Sensor available today is the Light Dependent Resistor or LDR. The property of LDR is that its resistance is inversely proportional to the intensity of the ambient light i.e., when the intensity of light increases, its resistance decreases and vise-versa.By using LDR is a circuit, we can calibrate the changes in its resistance to measure the intensity of Light. There are two other Light Sensors (or Photo Sensors) which are often used in complex electronic system design. They are Photo Diode and Photo Transistor. All these are Analog Sensors.

There are also Digital Light Sensors like BH1750, TSL2561, etc., which can calculate intensity of light and provide a digital equivalent value.Check out this simple LIGHT DETECTOR USING LDR project.

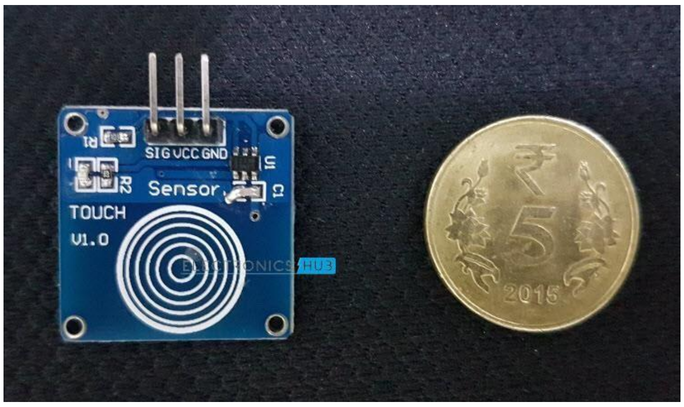
Smoke and Gas Sensors:One of the very useful sensors in safety related applications are Smoke and Gas Sensors. Almost all offices and industries are equipped with several smoke detectors, which detect any smoke (due to fire) and sound an alarm.Gas Sensors are more common in laboratories, large scale kitchens and industries. They can detect different gases like LPG, Propane, Butane, Methane (CH4), etc.



Now-a-days, smoke sensors (which often can detect smoke as well gas) are also installed in most homes as a safety measure.The "MQ" series of sensors are a bunch of cheap sensors for detecting CO, CO2, CH4, Alcohol, Propane, Butane, LPG etc. You can use these sensors to build your own Smoke Sensor Application.

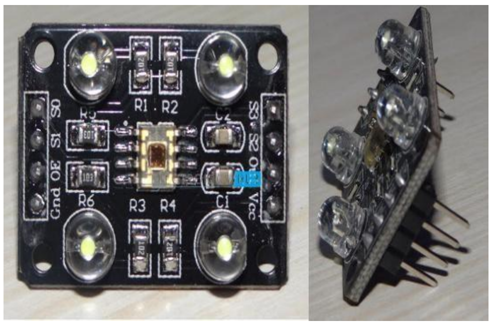
Alcohol Sensor:As the name suggests, an Alcohol Sensor detects alcohol. Usually, alcohol sensors are used in breathalyzer devices, which determine whether a person is drunk or not. Law enforcement personnel uses breathalyzers to catch drunk-and-drive culprits.

Touch Sensor: We do not give much importance to touch sensors but they became an integral part of our life. Whether you know or not, all touch screen devices (Mobile Phones, Tablets, Laptops, etc.) have touch sensors in them. Another common application of touch sensor is trackpads in our laptops.

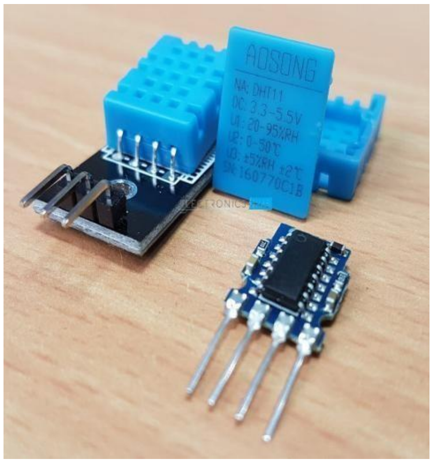


Touch Sensors, as the name suggests, detect touch of a finger or a stylus. Often touch sensors are classified into Resistive and Capacitive type. Almost all modern touch sensors are of Capacitive Types as they are more accurate and have better signal to noise ratio.

Color Sensor:A Color Sensor is an useful device in building color sensing applications in the field of image processing, color identification, industrial object tracking etc. The TCS3200 is a simple Color Sensor, which can detect any color and output a square wave proportional to the wavelength of the detected color.



Humidity Sensor: If you see Weather Monitoring Systems, they often provide temperature as well as humidity data. So, measuring humidity is an important task in many applications and Humidity Sensors help us in achieving this.Often all humidity sensors measure relative humidity (a ratio of water content in air to maximum potential of air to hold water). Since relative humidity is dependent on temperature of air, almost all Humidity Sensors can also measure Temperature.



Humidity Sensors are classified into Capacitive Type, Resistive Type and Thermal Conductive Type. DHT11 and DHT22 are two of the frequently used Humidity Sensors in DIY Community (the former is a resistive type while the latter is capacitive type).

Tilt Sensor: Often used to detect inclination or orientation, Tilt Sensors are one of the simplest and inexpensive sensors out there. Previously, tilt sensors are made up of Mercury (and hence they are sometimes called as Mercury Switches) but most modern tilt sensors contain a roller ball.

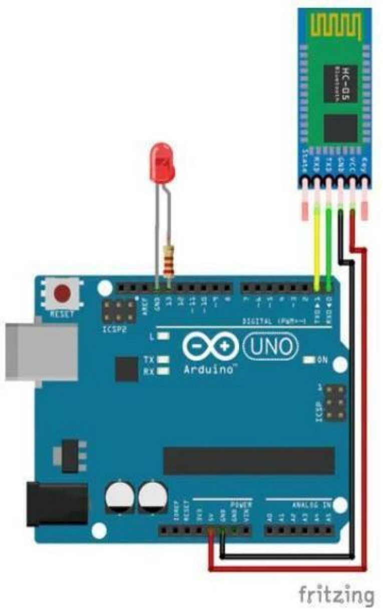
Experiment-4

AIM : To interface Bluetooth with Arduino/Raspberry Pi and write a program to send sensor data to smartphone using Bluetooth.

Component View :

NAME	**QUANTITY**	**COMPONENT**
U1	1	Arduino Uno R3
U2	1	HC-05 Bluetooth Module
R1	1	221 OHM Resistor
D1	1	Red LED
U3	1	Android Device

Circuit View



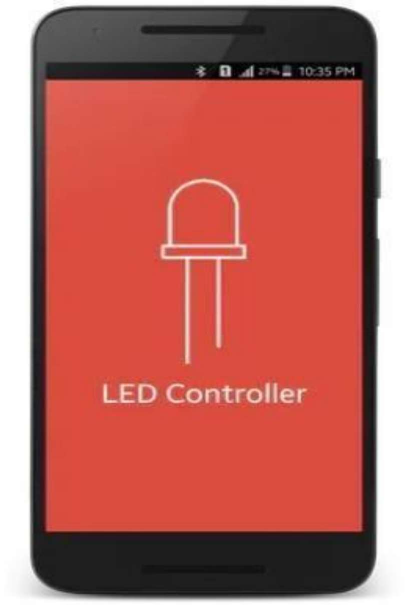
Code :

```
char Incoming_value = 0;

void setup() {

Serial.begin(9600);

pinMode(13, OUTPUT);

}

void loop(){

if(Serial.available() > 0) {

Incoming_value = Serial.read();

    Serial.print(Incoming_value);

    Serial.print("\n");

    if(Incoming_value == '1')

    digitalWrite(13, HIGH);

    else if(Incoming_value == '0')

    digitalWrite(13, LOW);

    }

}
```

__Simulation__ :

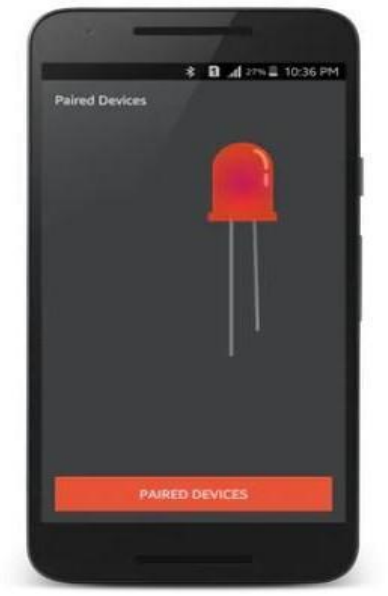
How to connect your andriod device with bluebooth module

- ✓ Pair your device with HC 05/06 Bluetooth module1) Turn ON HC 05/06 Bluetooth module2) Scan for available device3) Pair to HC 05/06 by entering default password 1234 OR 0000
- ✓ Install LED application on your android device
- ✓ Open the Application

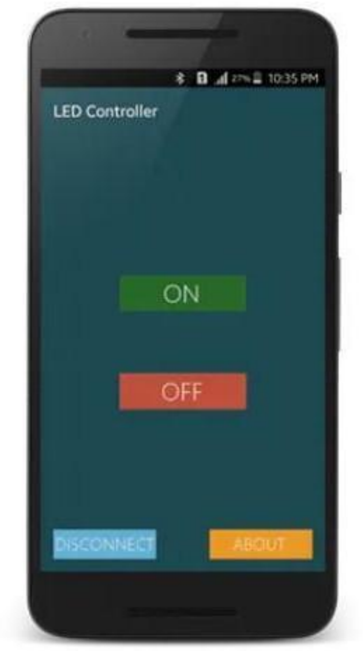


splash screen

- ✓ Press paired devices
- ✓ Select your Bluetooth module from the List (HC 05)



- ✓ After connecting successfully
- ✓ Press ON button to turn ON LED and OFF button to turn OFF the LE



Experiment - 5

Aim:- To interface a DHT11 sensor with an Arduino and print temperature and humidity readings using a program:

Materials Required:
- Arduino Uno board
- DHT11 temperature and humidity sensor
- Breadboard
- Jumper wires

Experiment Steps:

1. Connect the DHT11 sensor to the Arduino board as follows:
   - VCC pin of the sensor to the 5V pin of the Arduino
   - GND pin of the sensor to the GND pin of the Arduino
   - Data pin of the sensor to digital pin 2 of the Arduino

2. Open the Arduino IDE on your computer and create a new sketch.

3. In the sketch, include the DHT library by adding the following code to the top of the sketch:

```
#include <DHT.h>
```

4. Declare the DHT11 sensor as an object by adding the following code:

\`\`\`

```
#define DHTPIN 2          // Pin which is connected to the DHT sensor.
#define DHTTYPE DHT11     // DHT 11
DHT dht(DHTPIN, DHTTYPE);
```

5. In the `setup()` function, initialize the serial communication by adding the following code:

```
Serial.begin(9600);
```

6. In the `loop()` function, read the temperature and humidity values from the DHT11 sensor by adding the following code:

```
float humidity = dht.readHumidity();
float temperature = dht.readTemperature();
```

7. Print the temperature and humidity values to the serial monitor using the following code:

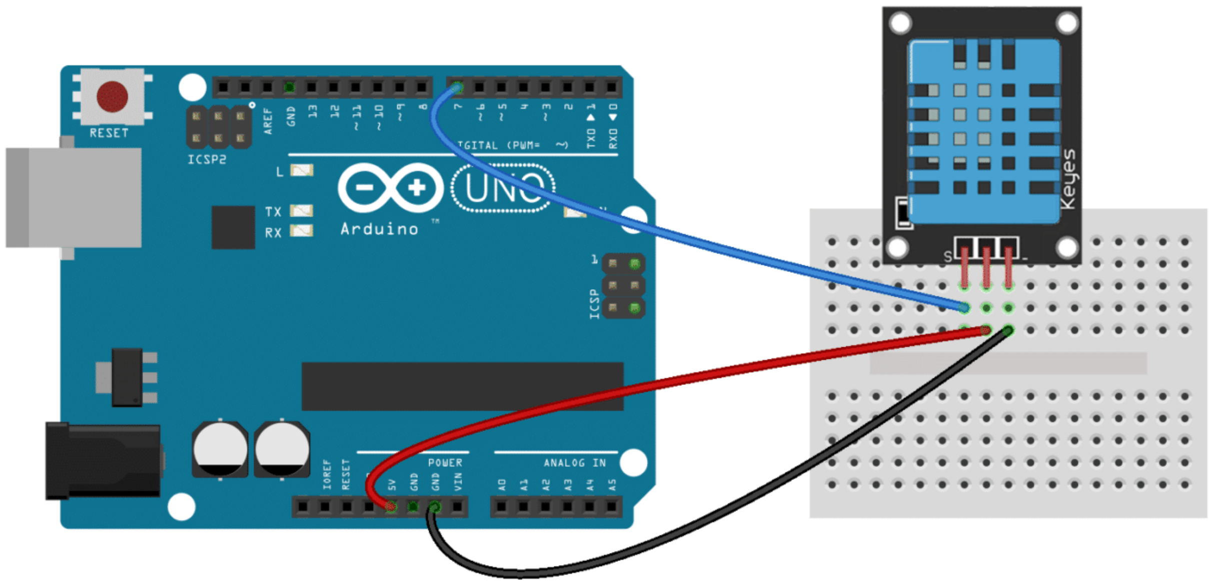
```
Serial.print("Humidity: ");
Serial.print(humidity);
Serial.print("%  Temperature: ");
Serial.print(temperature);
Serial.println("°C");
```

8. Upload the sketch to the Arduino board.

9. Open the serial monitor in the Arduino IDE by clicking on the magnifying glass icon in the toolbar or by selecting `Tools > Serial Monitor`.

10. Wait for a few seconds for the DHT11 sensor to stabilize, and then you should see the temperature and humidity readings printed to the serial monitor.

That's it! You have successfully interfaced the DHT11 sensor with the Arduino board and printed temperature and humidity readings using a program.



Experiment : 6

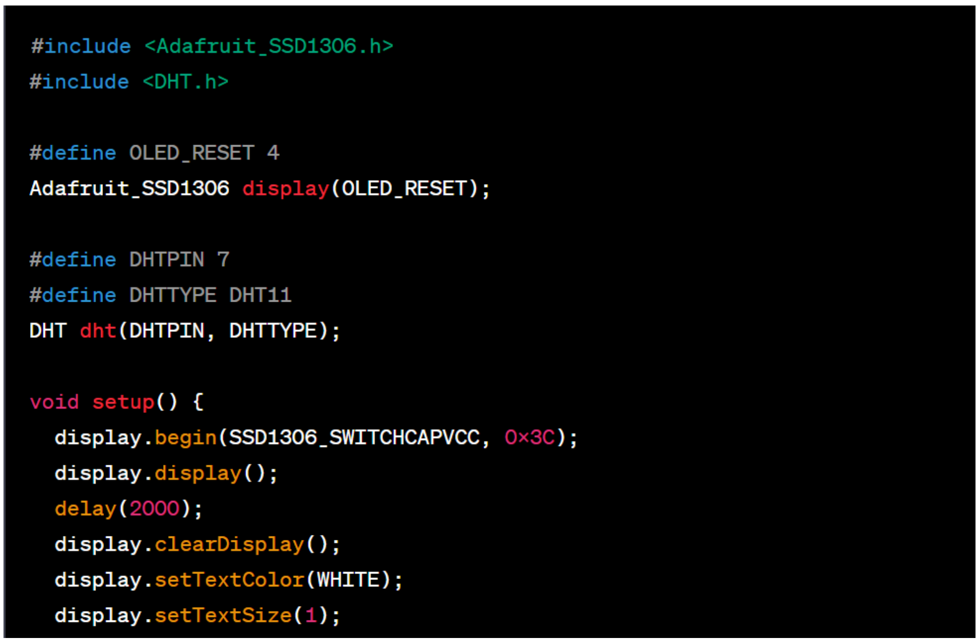
Aim : - To interface an OLED display with an Arduino and write a program to print temperature and humidity readings on it.

Materials needed:
1. Arduino board (e.g., Arduino Uno)
2. OLED display (e.g., SSD1306)
3. DHT11 or DHT22 temperature and humidity sensor
4. Jumper wires
5. Breadboard

Step 1: Hardware setup
1. Connect the VCC pin of the OLED display to the 5V pin on the Arduino.
2. Connect the GND pin of the OLED display to the GND pin on the Arduino.
3. Connect the SCL pin of the OLED display to the A5 pin on the Arduino.
4. Connect the SDA pin of the OLED display to the A4 pin on the Arduino.
5. Connect the VCC pin of the DHT sensor to the 5V pin on the Arduino.
6. Connect the GND pin of the DHT sensor to the GND pin on the Arduino.
7. Connect the data pin (either DHT11 pin 2 or DHT22 pin 1) of the DHT sensor to a digital pin on the Arduino (e.g., pin 7).

Step 2: Install libraries
1. Open the Arduino IDE on your computer.
2. Go to "Sketch" -> "Include Library" -> "Manage Libraries".
3. Search for "Adafruit SSD1306" and click the "Install" button to install the library for the OLED display.
4. Search for "DHT" and install the library for the DHT sensor.

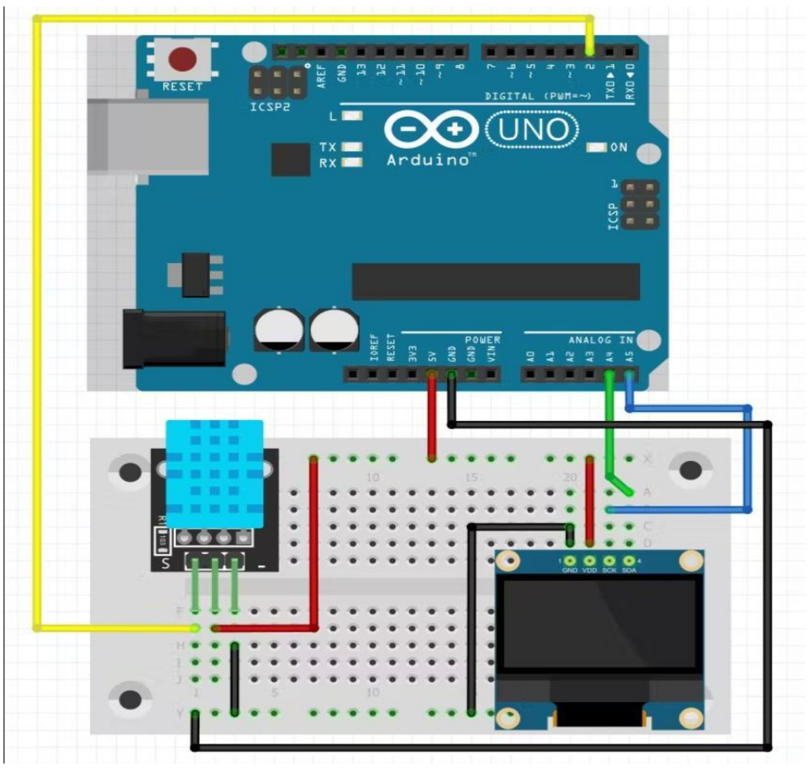
Step 3: Upload the code
Copy and paste the following code into the Arduino IDE:



```
#include <Adafruit_SSD1306.h>
#include <DHT.h>

#define OLED_RESET 4
Adafruit_SSD1306 display(OLED_RESET);

#define DHTPIN 7
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.display();
  delay(2000);
  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setTextSize(1);
  display.setTextSize(1);
  display.setCursor(0, 0);
  display.println("Temperature &");
  display.println("Humidity Readings");
  display.display();
  delay(2000);
}

void loop() {
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  display.clearDisplay();
  display.setCursor(0, 0);
  display.println("Temperature: " + String(temperature) + " C");
  display.println("Humidity: " + String(humidity) + " %");
  display.display();

  delay(2000);
}
```

Step 4: Upload and Run
1. Connect your Arduino to your computer using a USB cable.
2. Select the appropriate board and port in the Arduino IDE.
3. Click the "Upload" button to upload the code to the Arduino.
4. Once uploaded, you should see the temperature and humidity readings displayed on the OLED display.

That's it! You have successfully interfaced an OLED display with an Arduino and created a program to print temperature and humidity readings on it.



Experiment - 7

Aim :- Write a program on Arduino to send temperature and humidity data from thingspeak cloud.

Components Required :-

- Arduino Uno
- ESP8266 WiFi Module
- DHT11 Sensor
- Breadboard
- Jumper Wires

Experiment steps :-

Step 1: ThingSpeak Setup for Temperature and Humidity Monitoring

For creating your channel on Thingspeak, you first need to Sign up on Thingspeak. In case if you already have an account on Thingspeak, just sign in using your id and password.

For creating your account go to www.thinspeak.com

Step 2: Create a Channel for Your Data

Once you Sign in after your account verification, Create a new channel by clicking "New Channel" button.

After clicking on "New Channel", enter the Name and Description of the data you want to upload on this channel. For example, I am sending my DHT11 sensor data, so I named it DHT11 data.

Enter the name of your data 'Temperature' in Field1 and 'Humidity' in Field2. If you want to use more Fields, you can check the box next to Field option and enter the name and description of your data.

After this, click on the save channel button to save your details.

Step 3: API Key

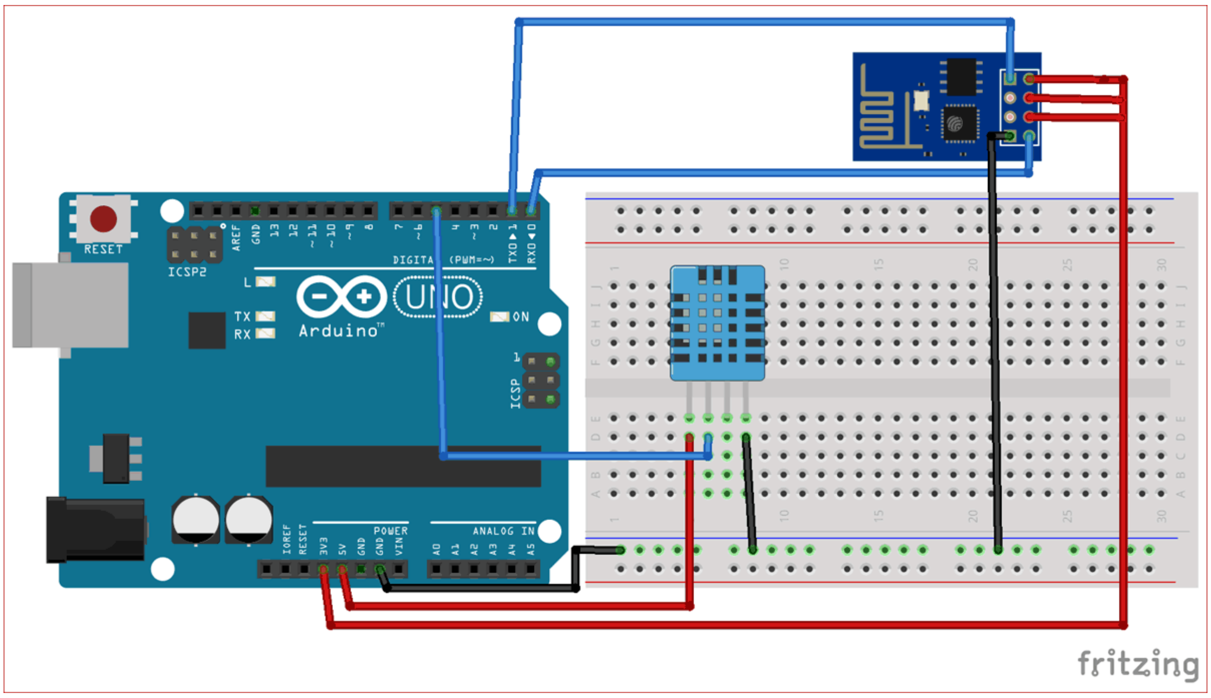
To send data to Thingspeak, we need a unique API key, which we will use later in our code to upload our sensor data to Thingspeak Website.

Click on "API Keys" button to get your unique API key for uploading your sensor data.

Programming Arduino for Sending data to ThingSpeak

To program Arduino, open Arduino IDE and choose the correct board and port from the *'tool'* menu.

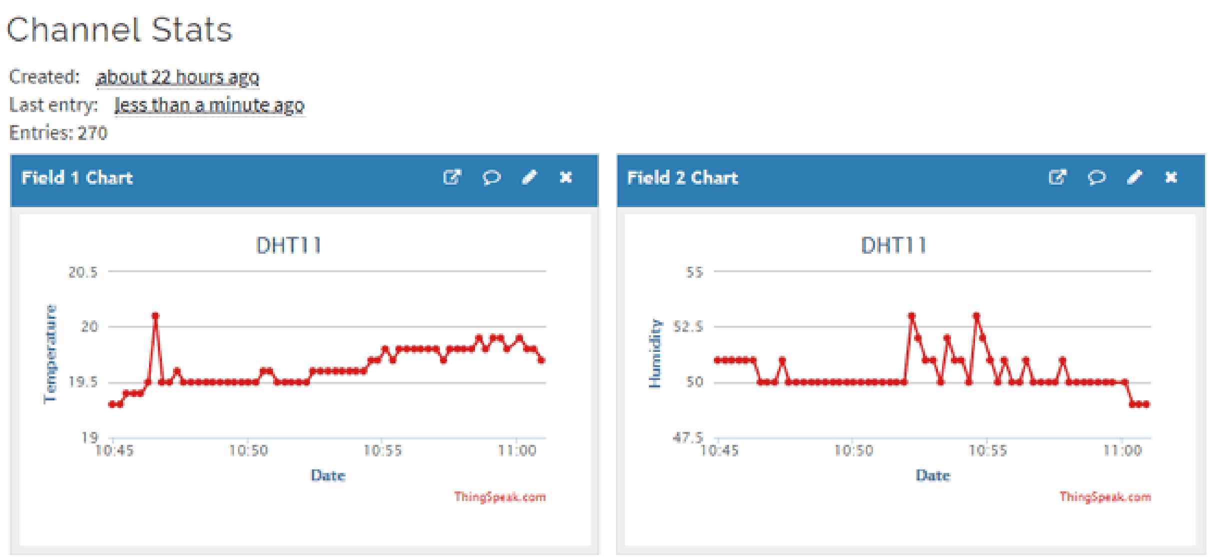
Complete code is given at the end of this tutorial. Upload it in Arduino UNO. If you successfully upload your program.



Connections are given in below table:

S.NO.	Pin Name	Arduino Pin
1	ESP8266 VCC	3.3V
2	ESP8266 RST	3.3V
3	ESP8266 CH-PD	3.3V
4	ESP8266 RX	TX
5	ESP8266 TX	RX
6	ESP8266 GND	GND
7	DHT-11 VCC	5V

8	DHT-11 Data	5
9	DHT-11 GND	GND


Channel Stats

Created: about 22 hours ago

Last entry: less than a minute ago

Entries: 270

Code:-

```
#include <stdlib.h>

#include <DHT.h>

#define DHTPIN 5      // DHT data pin connected to Arduino pin 5

#define DHTTYPE DHT11    // DHT11 (DHT Sensor Type )

DHT dht(DHTPIN, DHTTYPE); // Initialize the DHT sensor

#define SSID "WiFi Name"    // "WiFi Name"

#define PASS "WiFi Password"         // "Password"

#define IP "184.106.153.149"// thingspeak.com ip

String msg = "GET /update?key=Your API Key"; //change it with your key...

float temp;

int hum;

String tempC;

int error;

void setup()

{

  Serial.begin(115200); // use default 115200.

  Serial.println("AT");
  delay(5000);
  if(Serial.find("OK")){
            connectWiFi();
  }
}
void loop(){
   start:
  error=0;
  temp = dht.readTemperature();
  hum = dht.readHumidity();
  char buffer[10];
  tempC = dtostrf(temp, 4, 1, buffer);
  updateTemp();
  if (error==1){
            goto start;
  }

  delay(5000);
}
void updateTemp(){
  String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += IP;
  cmd += "\",80";
  Serial.println(cmd);
  delay(2000);
  if(Serial.find("Error")){
            return;
  }
  cmd = msg ;
  cmd += "&field1=";
  cmd += tempC;
  cmd += "&field2=";
  cmd += String(hum);
  cmd += "\r\n";
  Serial.print("AT+CIPSEND=");
  Serial.println(cmd.length());
  if(Serial.find(">")){
            Serial.print(cmd);
  }
  else{
            Serial.println("AT+CIPCLOSE");
            //Resend...
            error=1;
  }
}
  boolean connectWiFi(){
  Serial.println("AT+CWMODE=1");
  delay(2000);
  String cmd="AT+CWJAP=\"";
  cmd+=SSID;
  cmd+="\",\"";
  cmd+=PASS;
  cmd+="\"";
  Serial.println(cmd);
  delay(5000);
  if(Serial.find("OK")){
            return true;
  }else{
            return false;
  }
}
```