DHANALAKSHMI SRINIVASAN ENGINEERING COLLEGE
(AUTONOMOUS)
(Approved by AICTE & Affiliated to Anna University, Chennai)
Re-Accredited by NAAC with 'A' Grade
Accredited by NBA for AERO, BME, CSE, ECE, EEE, IT & MECH.
PERAMBALUR-621212, TAMILNADU, INDIA.
Website: www.dsengg.ac.in

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

U23CSP62 -INTERNET OF THINGS LABORATORY

PREPARED BY

1.Mrs.R.GAYATHRI,AP,ECE

2.Mrs.PARIMALA,AP,ECE

EXP : 1 **Arduino Platform and Programming**

Aim:

Getting Started with IoT (Arduino) and perform necessary software installation

Arduino:

- Arduino is a platform that makes it easy for you to build projects using electronics.
- IoT is a way of using electronics - to make electronic modules talk to each other remotely and wirelessly (often using a Cloud) to solve problems.
- Now, Arduino can also help you easily build IoT projects in two ways: Using traditional Arduino boards and attaching communication breakout modules (like nRF, Bluetooth,WiFi,LoRA,GSM, etc) to them.
- Arduino is a micro controller that can be connected to one or more sensors and help you capture the data or information and then pass it on to processor. If you know the full stack of IoT then you shouldalso look at Raspberry.
- RaspPi is a microprocessor so the basic difference between Arduino and RasPi is that RaspPi is controller plus processor and Arduino is just a micro controller.
- They suit the need for different use cases. You can easily read online about this both.

Download and install the Arduino software (Arduino IDE 1.8.15)

- Go to the Arduino website and click the download link to go to the download page.
- After downloading, locate the downloaded file on the computer and extract the folder from thedownload zipped file. Copy the folder to a suitable place such as your desktop.

INSTALLING THE ARDUINO IDE ONWindows PCs

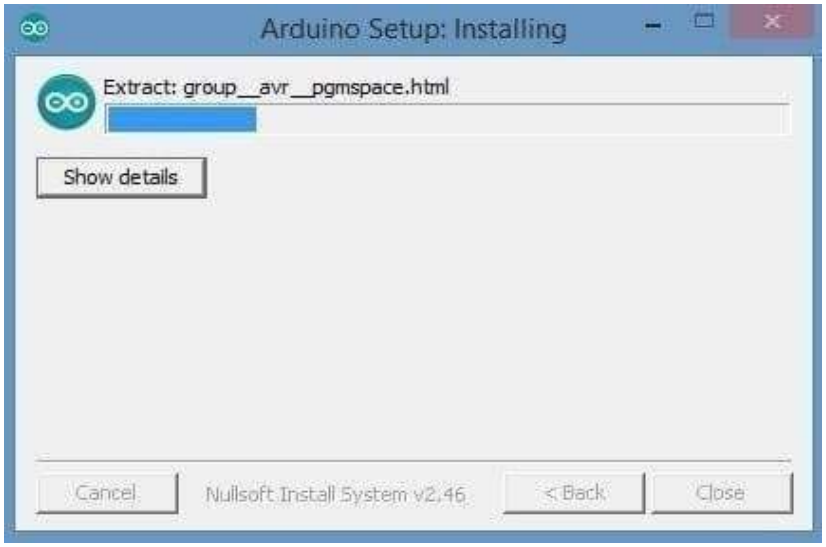
1. Visit http://www.arduino.cc/en/main/software to download the latest Arduino IDE version for your computer's operating system. There are versions for Windows, Mac, and Linux systems. At the download page, click on the "Windows Installer" option for the easiest installation.
2. Save the .exe file to your hard drive.
3. Open the .exe file.
4. Click the button to agree to the licensing agreement:

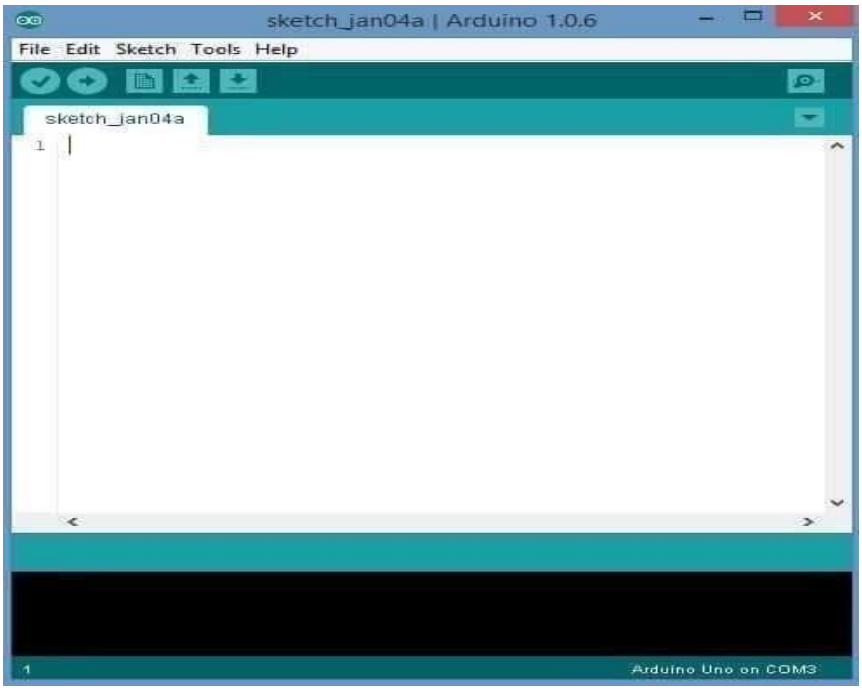
5. Decide which components to install, and then click "Next":

6. Select which folder to install the program to, then click “Install”:

7. Wait for the program to finish installing, and then click “Close”:

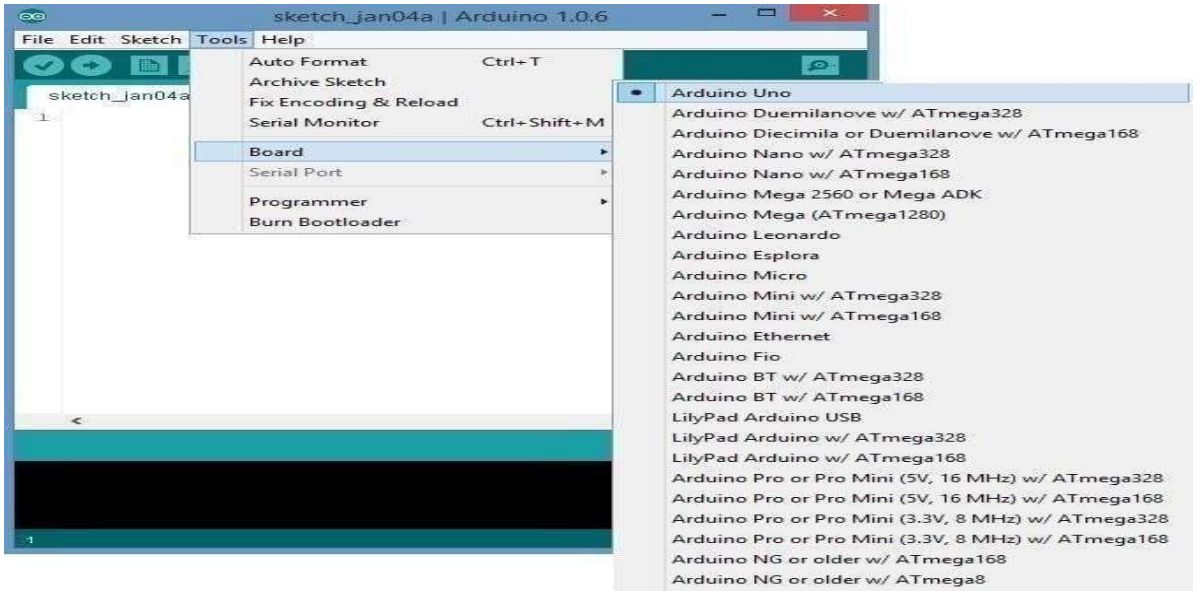
8. Now find the Arduino shortcut on your Desktop and click on it. The IDE will open up and you’ll seethecode editor:





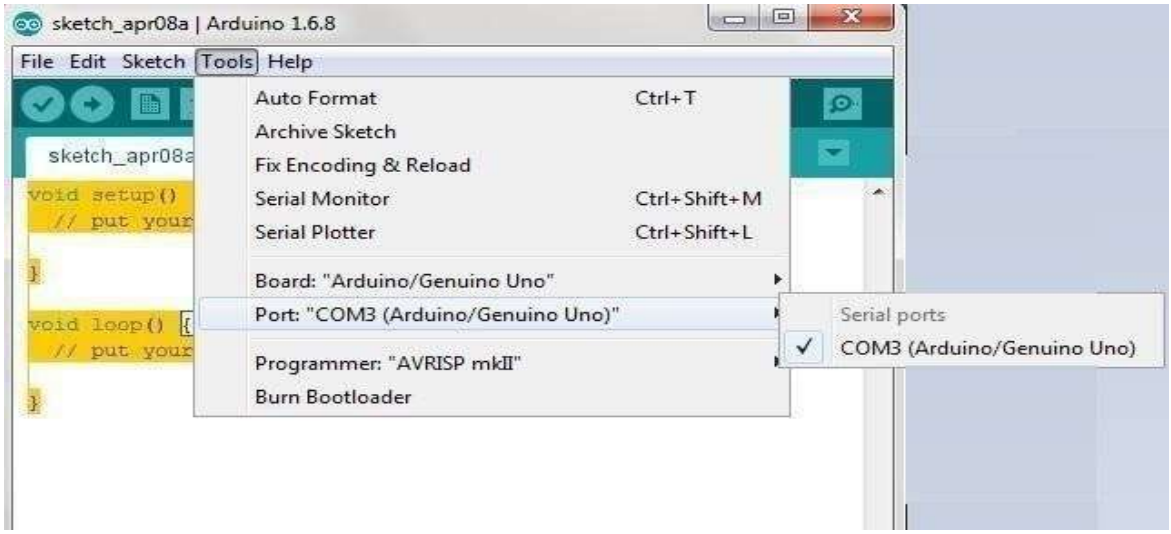
CONFIGURING THE ARDUINO IDE

The next thing to do is to make sure the software is set up for your particular Arduino board. Go to the "Tools" drop-down menu, and find "Board". Another menu will appear where you can select from a list of Arduino models. I have the Arduino Uno R3, so I chose "Arduino Uno".



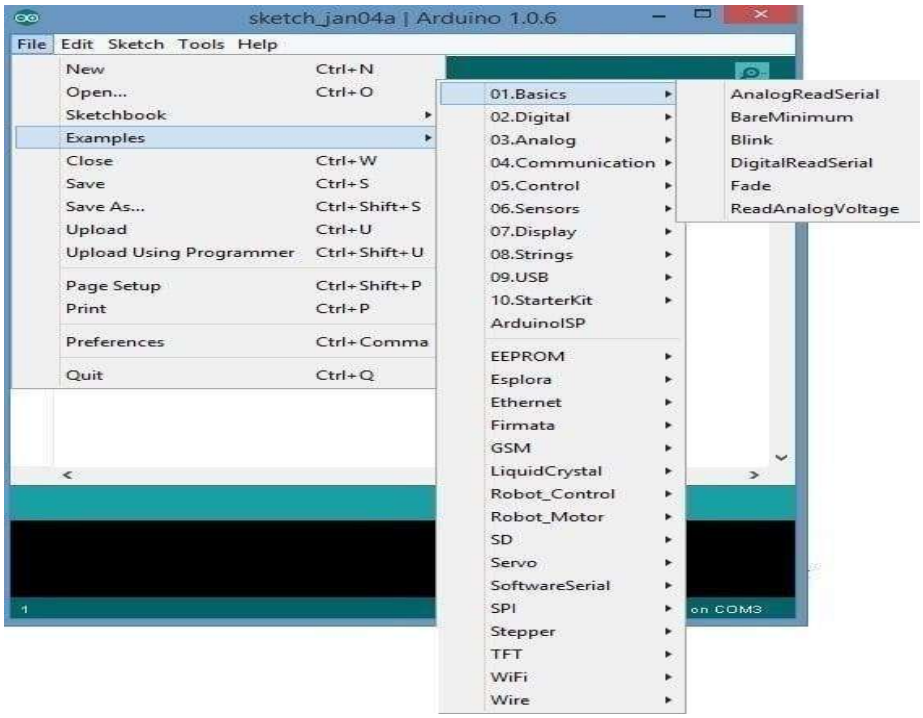
Selecting arduino board

Selecting arduino port



EXPLORING THE ARDUINO IDE

If you want, take a minute to browse through the different menus in the IDE. There is a good variety of example programs that come with the IDE in the "Examples" menu. These will help you get started with your Arduino right away without having to do lots of research:



Running the Arduino IDE Software

This is a display of the Arduino IDE Software. The application is ready to be used to create amazing projects.

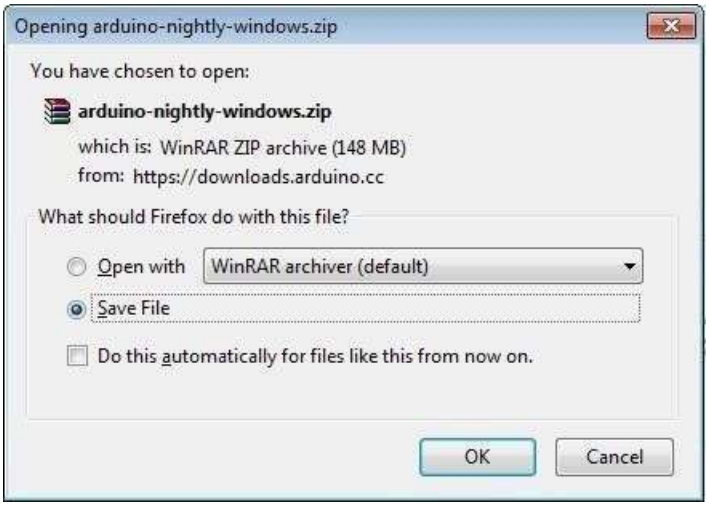
Steps to connect Arduino board:

Step 1 − First you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, you will needa standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image.

In case you use Arduino Nano, you will need an A to Mini-B cable instead as shown in the following image.

Step 2 − 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 3 − Power up your board.

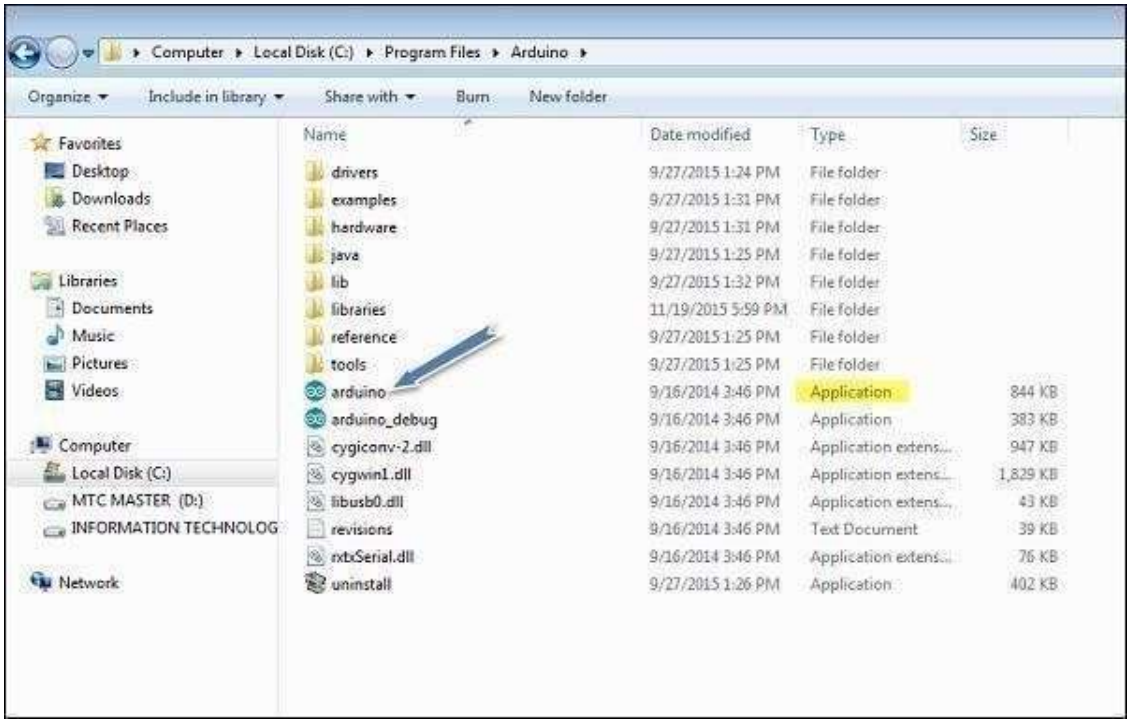
The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an Arduino Diecimila, you have tomake sure that the board is configured to draw power from the USB connection. The power source

is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port.

Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

Step 4 − Launch Arduino IDE.

After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.

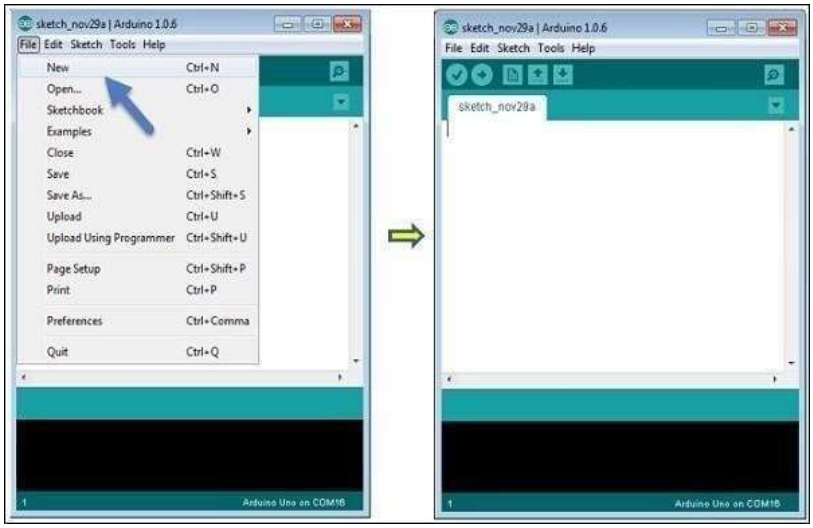


Step 5 − Open your first project.

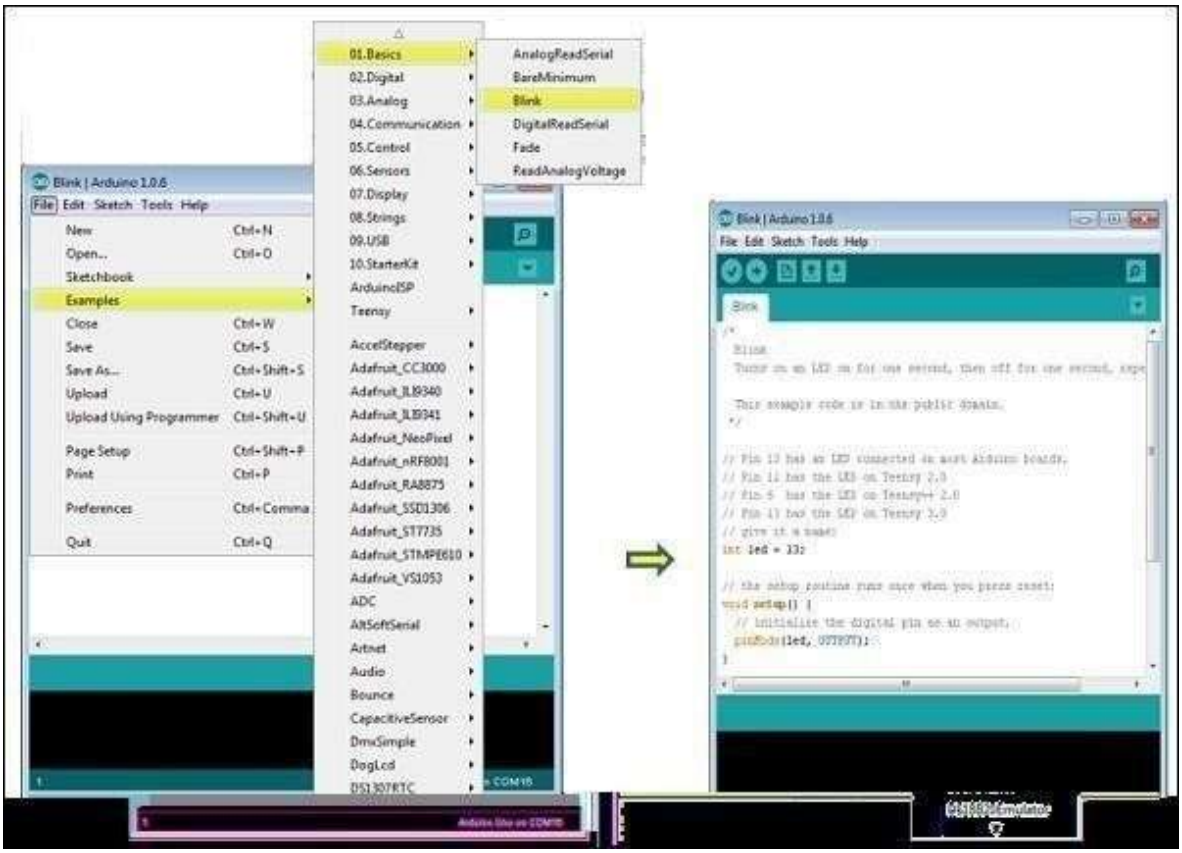
Once the software starts, you have two options −

- 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.

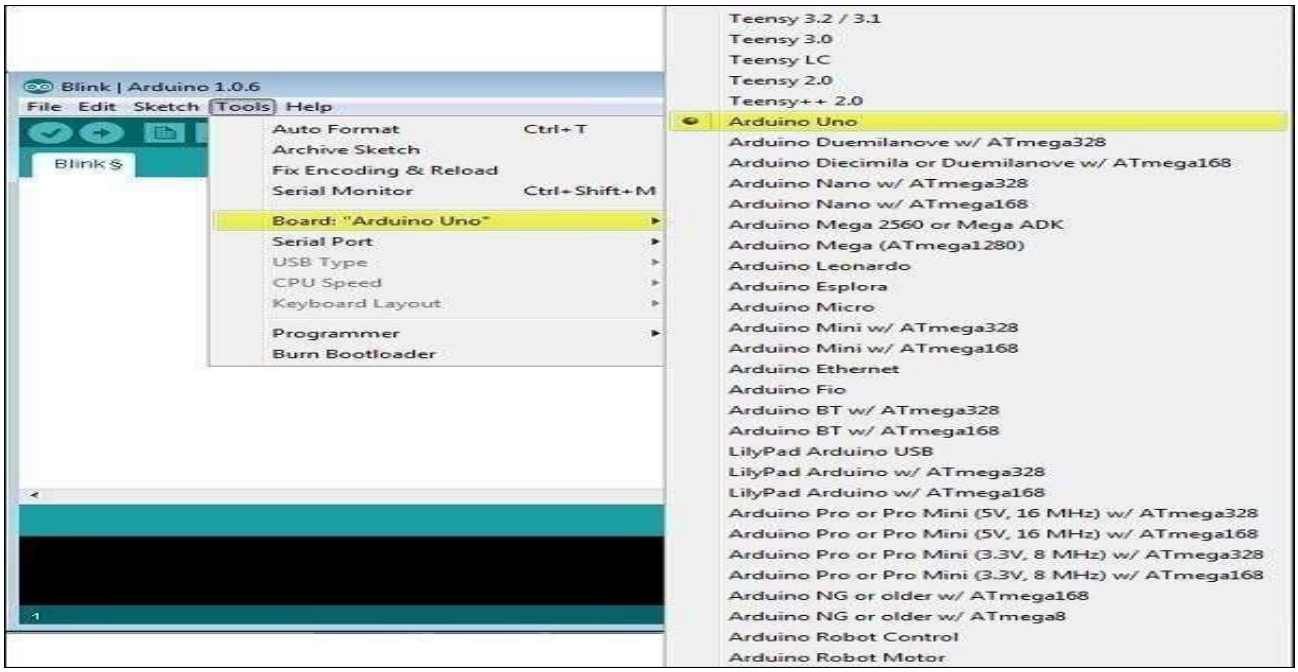


Here, we are selecting just one of the examples with the name Blink. It turns the LED on and off with some time delay. You can select any other example from the list.

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.

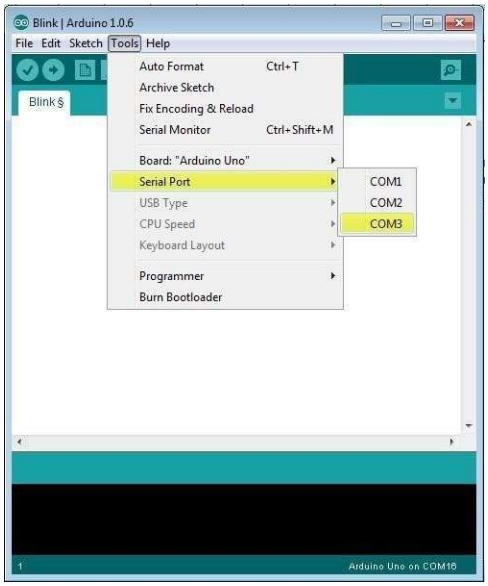
Go to Tools → Board and select your board.



Here, we have selected Arduino Uno board according to our tutorial, but you must select the name matchingthe board that you are using.

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.

A − Used to check if there is any compilation error.

B − Used to upload a program to the Arduino board.

C − Shortcut used to create a new sketch.

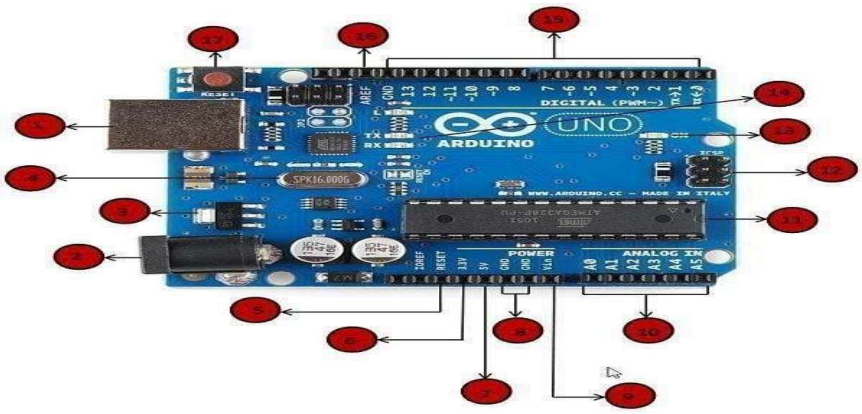
D − Used to directly open one of the example sketch.

E − Used to save your sketch.

F − Serial monitor used to receive serial data from the board and send the serial data to the board.

Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TXLEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

Arduino uno Board



1. Power USB

Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection (1).

2. Power (Barrel Jack)

Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2).

3. Voltage Regulator

The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements.

4. Crystal Oscillator

The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. Ittells us that the frequency is 16,000,000 Hertz or 16 MHz.

5, 17- Arduino Reset

You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO boardin two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5).

6,7,8,9: Pins (3.3, 5, GND, Vin)

- 3.3V (6) − Supply 3.3 output volt
  5V (7) − Supply 5 output volt
- Most of the components used with Arduino board works fine with 3.3 volt and 5 volt.
- GND (8)(Ground) − There are several GND pins on the Arduino, any of which can be usedto ground your circuit.
- Vin (9) − This pin also can be used to power the Arduino board from an external power source, like AC mains power supply.

10. Analog pins

The Arduino UNO board has six analog input pins A0 through A5. These pins can read the signal from ananalog sensor like the humidity sensor or temperature sensor and convert it into a digital value that

can beread by the microprocessor.

Main microcontroller

Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers areusually 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. For more details about the IC construction and functions, you can refer to the data sheet.

11. ICSP pin

Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK,RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus.

12. Power LED indicator

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, then there is something wrong with the connection.

13. TX and RX LEDs

On your board, you will find two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication. Second, the TX and RX led (13). The TX led flashes with different speed while sending theserial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.

14. Digital I/O

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 labeled "~" can be used to generate PWM.

15. AREF

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.

Sketch − The first new terminology is the Arduino program called "**sketch**".

Structure

Arduino programs can be divided in three main parts: **Structure, Values** (variables and constants), and **Functions**. In this tutorial, we will learn about the Arduino software program, step by step, and how we can write the program without any syntax or compilation error.

Let us start with the **Structure**. Software structure consists of two main functions −

- Setup( ) function
- Loop( ) function

```
Void setup ( ) {


}
```

- **PURPOSE** − The **setup()** function is called when a sketch starts. Use it to initialize the variables, pinmodes, start using libraries, etc. The setup function will only run once, after each power up or resetof the Arduino board.
- **INPUT** − -
- **OUTPUT** − -
- **RETURN** − -

```
Void Loop ( ) {
                    }
```

- **PURPOSE** − After creating a **setup()** function, which initializes and sets the initial values, the **loop()** function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.
- **INPUT** − -
- **OUTPUT** − -
- **RETURN** − -

Result:

Thus, the Arduino IDE has been successfully installed.

EXP:2 **Raspberry pi Platform and Python Programming.**

AIM:

Introduction to Raspberry Pi Pico W:

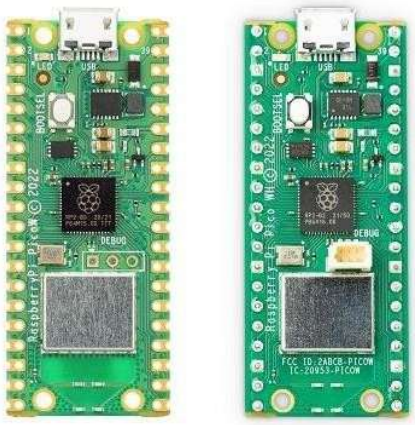
The Raspberry Pi Pico W is a compact and affordable microcontroller board developed by the Raspberry Pi Foundation. Building upon the success of the Raspberry Pi Pico, the Pico W variant brings wireless connectivity to the table, making it an even more versatile platform for embedded projects. In this article, we will provide a comprehensive overview of the Raspberry Pi Pico W, highlighting its key features and capabilities.

Features:

- RP2040 microcontroller with 2MB of flash memory
- On-board single-band 2.4GHz wireless interfaces (802.11n)
- Micro USB B port for power and data (and for reprogramming the flash)
- 40 pins 21mmx51mm 'DIP' style 1mm thick PCB with 0.1″ through-hole pins also with edge castellations
- Exposes 26 multi-function 3.3V general purpose I/O (GPIO)
- 23 GPIO are digital-only, with three also being ADC-capable
- Can be surface mounted as a module
- 3-pin ARM serial wire debug (SWD) port
- Simple yet highly flexible power supply architecture
- Various options for easily powering the unit from micro-USB, external supplies, or batteries
- High quality, low cost, high availability
- Comprehensive SDK, software examples, and documentation
- Dual-core Cortex M0+ at up to 133MHz
- On-chip PLL allows variable core frequency
- 264kByte multi-bank high-performance SRAM

Raspberry Pi Pico W:

The Raspberry Pi Pico W is based on the RP2040 microcontroller, which was designed by Raspberry Pi in-house. It combines a powerful ARM Cortex-M0+ processor with built-in Wi-Fi connectivity, opening up a range of possibilities for IoT projects, remote monitoring, and wireless communication. The Pico W retains the same form factor as the original Pico, making it compatible with existing Pico accessories and add-ons.



RP2040 Microcontroller:

At the core of the Raspberry Pi Pico W is the RP2040 microcontroller. It features a dual-core ARM Cortex-M0+ processor running at 133MHz, providing ample processing power for a wide range of applications. The microcontroller also includes 264KB of SRAM, which is essential for storing and manipulating data during runtime. Additionally, the RP2040 incorporates 2MB of onboard flash memory for program storage, ensuring sufficient space for your code and firmware.

Wireless Connectivity:

The standout feature of the Raspberry Pi Pico W is its built-in wireless connectivity. It includes an onboard Cypress CYW43455 Wi-Fi chip, which supports dual-band (2.4GHz and 5GHz) Wi-Fi 802.11b/g/n/ac. This allows the Pico W to seamlessly connect to wireless networks, communicate with other devices, and access online services. The wireless capability opens up new avenues for IoT projects, remote monitoring and control, and real-time data exchange.

GPIO and Peripherals:

Similar to the Raspberry Pi Pico, the Pico W offers a generous number of GPIO pins, providing flexibility for interfacing with external components and peripherals. It features 26 GPIO pins, of which 3 are analog inputs, and supports various protocols such as UART, SPI, I2C, and PWM. The Pico W also includes onboard LED indicators and a micro-USB port for power and data connectivity.

MicroPython and C/C++ Programming:

The Raspberry Pi Pico W can be programmed using MicroPython, a beginner-friendly programming language that allows for rapid prototyping and development. MicroPython provides a simplified syntax and high-level abstractions, making it easy for newcomers to get started. Additionally, the

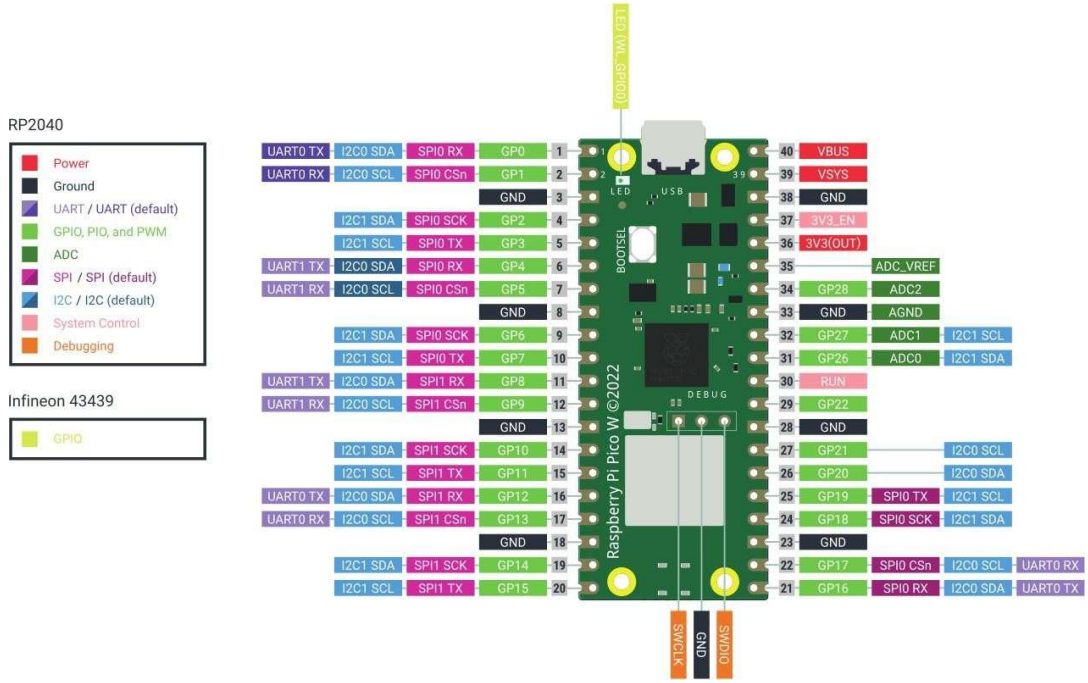
Pico W is compatible with C/C++ programming, allowing experienced developers to leverage the rich ecosystem of libraries and frameworks available.

Programmable Input/Output (PIO) State Machines:

One of the unique features of the RP2040 microcontroller is the inclusion of Programmable Input/Output (PIO) state machines. These state machines provide additional processing power and flexibility for handling real-time data and timing-critical applications. The PIO state machines can be programmed to interface with custom protocols, generate precise waveforms, and offload tasks from the main processor, enhancing the overall performance of the system.

Open-Source and Community Support

As with all Raspberry Pi products, the Pico W benefits from the vibrant and supportive Raspberry Pi community. Raspberry Pi provides extensive documentation, including datasheets, pinout diagrams, and programming guides, to assist developers in understanding the board's capabilities. The community offers forums, online tutorials, and project repositories, allowing users to seek help, share knowledge, and collaborate on innovative projects.



The Raspberry Pi Pico W brings wireless connectivity to the popular Raspberry Pi Pico microcontroller board. With its powerful RP2040 microcontroller, built-in Wi-Fi chip, extensive GPIO capabilities, and compatibility with MicroPython and C/C++ programming, the Pico W offers a versatile and affordable platform for a wide range of embedded projects. Whether you are a beginner or an experienced developer, the Raspberry Pi Pico W provides a user-friendly and flexible platform to bring your ideas to life and explore the exciting world of wireless IoT applications.

RESULT:

Thus the study about the Introduction to Raspberry PI platform has been successfully executed.

EXP : 3 **Turn ON and OFF the LEDs.**

AIM:

To write a program to switch light on when the input is 1 and switch the light off when the input is 0 using Arduino.

COMPONENTS REQUIRED:

1. Arduino
2. Breadboard
3. Jumper wires
4. Resistor
5. LED

ALGORITHM:

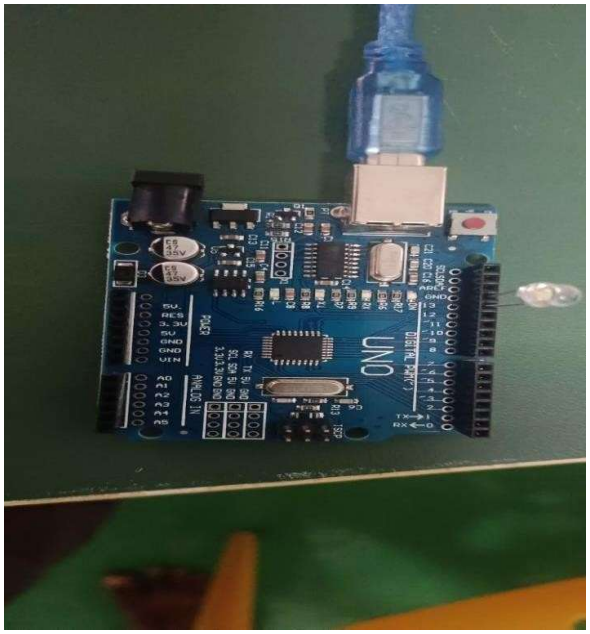
- Start the process.
- Connect micro USB power input to Arduino.
- Connect HDMI to the system to act as monitor for Arduino.
- Connect USB port 2.0 to mouse and keyboard.
- Enter the coding in the terminal for installing python and GPTO.
- Open new sketch →enter coding →save.
- Save file location → open terminal → paste file location in terminal → press enter.
- In the terminal window to get output enter 0 or 1, to switch light ON when the input is 1 and switch light OFF when the input is0 in breadboard using Arduino.
- Stop the process.

CODING:

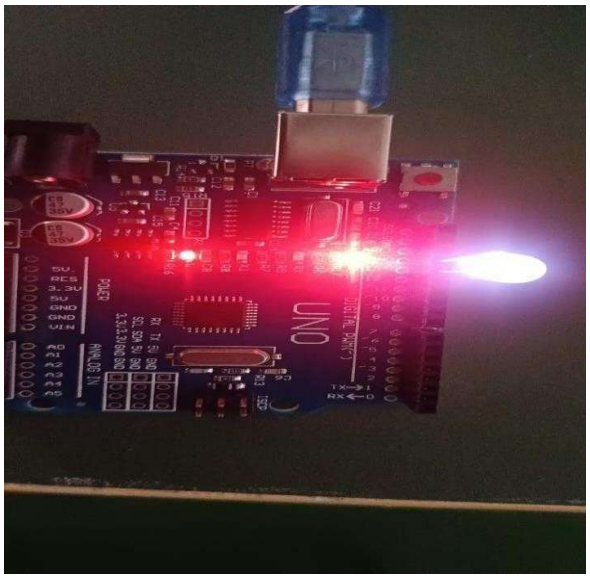
```
void setup() {
// put your setup code here, to run once:
pinMode(13,OUTPUT);
}
void loop() {
// put your main code here, to run repeatedly:
digitalWrite(13,HIGH);
delay(10000);
digitalWrite(13,LOW);
delay(10000);
}
```

OUTPUT:

Input



Output



RESULT:

Thus the output to switch light ON/OFF using Arduino has been successfully executed.

EXP : 4(A) **Identify the objects using IR sensor**

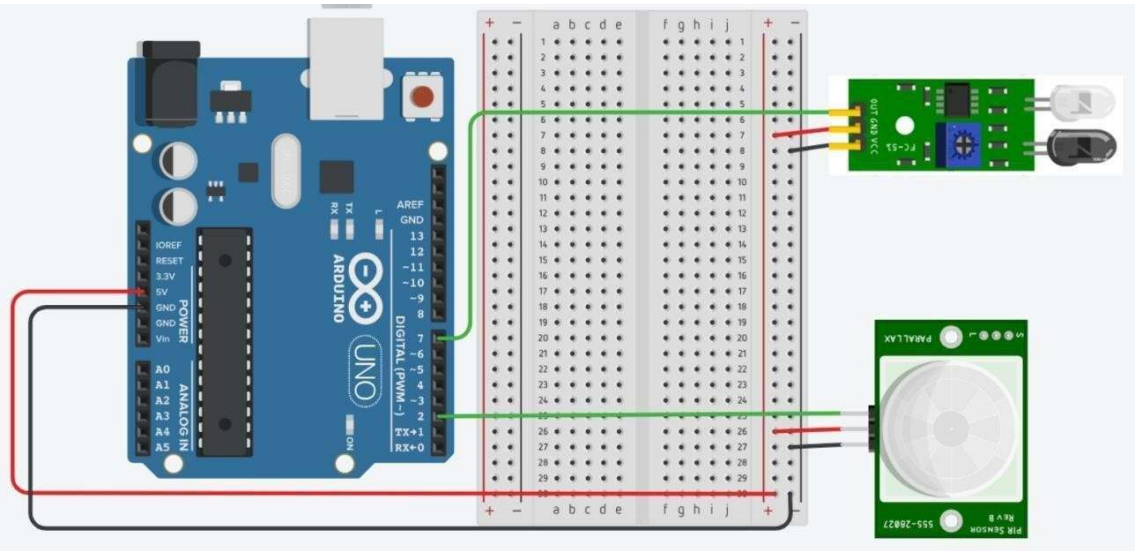
AIM:

The aim of this experiment is to identify objects using IR sensor and Arduino UNO.

COMPONENTS REQUIRED:

- Arduino Uno
- IR transmitter
- IR Receiver
- IR sensor
- Jumper wires
- Bread Board

Circuit Diagram:



ALGORITHM:

IR SENSOR:

- Connect the IR sensor to the breadboard using an IR interface cable with 3 -pin header. Connect Ground to the – pin, Power to the + pin of the PIR sensor.
- Connect your power supply to the breadboard. The recommended power is within the range of +4.5 - +5.5 Volts.
- Fix the sensor to a table or wall so that it has 180 degrees of detection range.

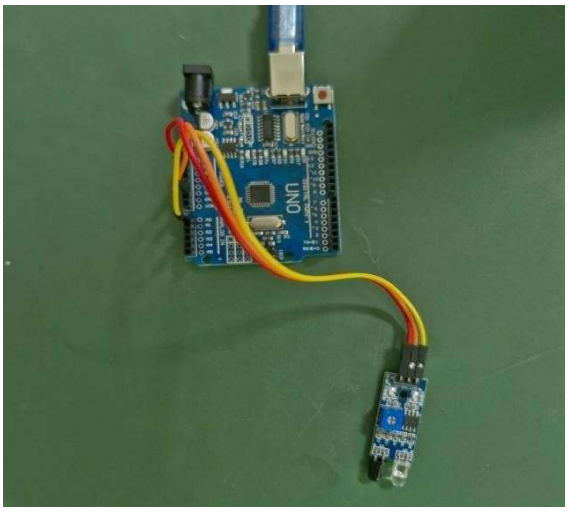
- Check your circuit, and turn on your power supply to test the IR distance sensor.
- Wait for approximately 44 milliseconds for the IR to start up.
- Place an object within the sensor's detection range.

Program:

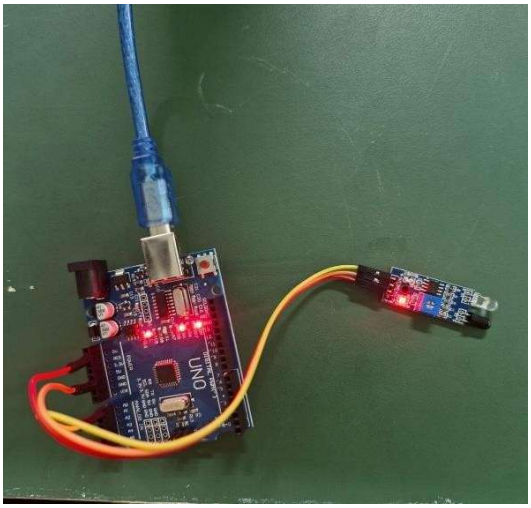
```
 int IR_sensor=8;
int value=0;
void setup() (
// put your setup code here, to run once:
Serial.begin(9600);
pinMode(IR_sensor,INPUT);
}
void loop() {
// put your main code here, to run repeatedly:
value=digitalRead(IR_sensor);
if(value==0)
{
Serial.println("OBJECT  DETECT");
}
else
{
Serial.println("OBJECT  CLEAR");
}
}
```

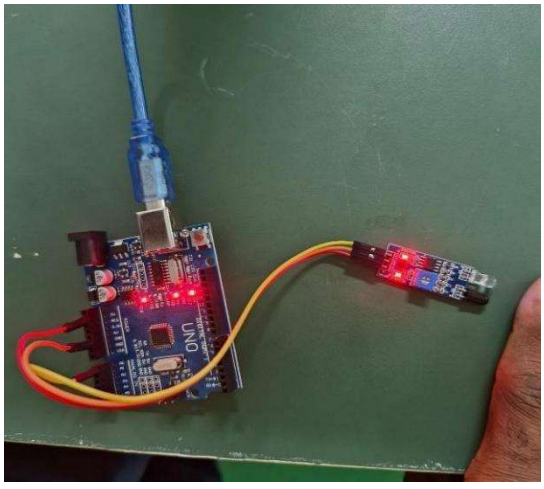
Output:

Input :



Output :





Result:

Thus, the objects have been identified successfully using IR sensors.

EXP : 4(B) **Identify the objects using PIR sensor**

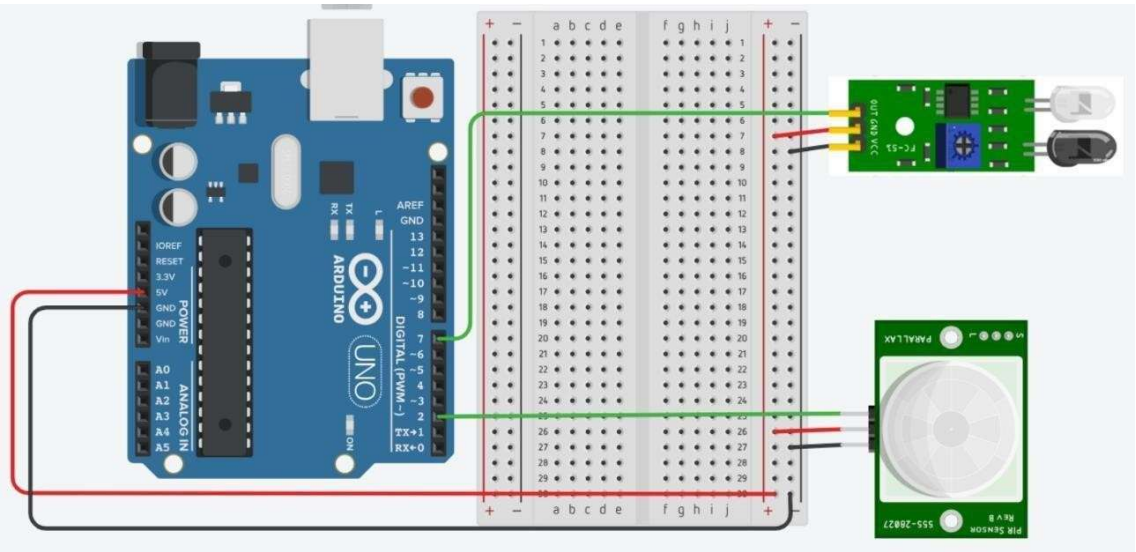
AIM:

The aim of this experiment is to identify objects using PIR sensors and Arduino UNO.

COMPONENTS REQUIRED:

- Arduino Uno
- IR transmitter
- IR Receiver
- PIR sensor
- Jumper wires
- Bread Board

Circuit Diagram:



ALGORITHM:

PIR SENSOR:

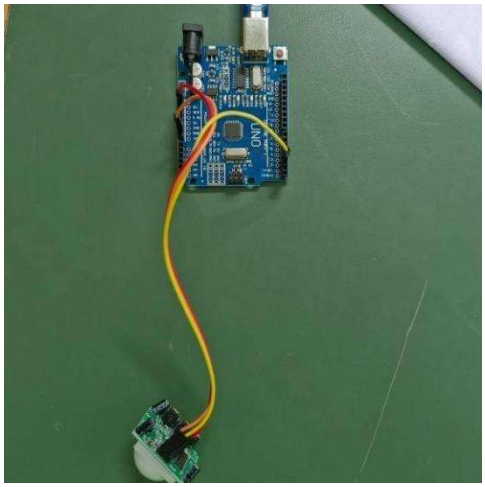
- Wire the PIR sensor to the breadboard making sure that Ground goes to the – pin, Power to the + pin of the PIR sensor.
- Connect your power supply to the breadboard. The recommended power is within the range of +3.3 - +5.0 Volts.
- Secure the sensor to a table or wall so that it is facing parallel to the scanning surface.
- Check your circuit, and turn on your power supply to test the PIR motion sensor.
- Wait for 10 to 60 seconds for the PIR sensor to calibrate itself.

Program:

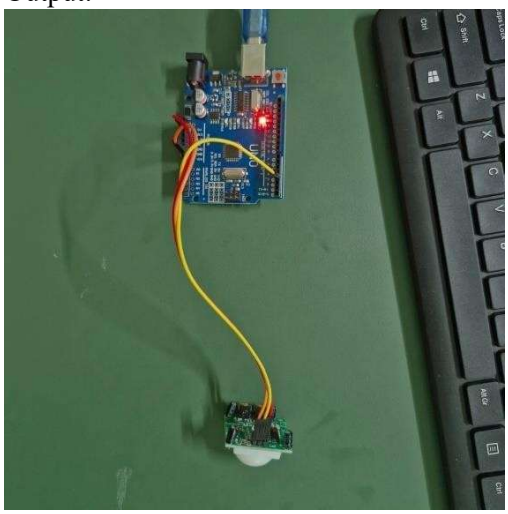
```
void setup() {
// put your setup code here, to run once:
Serial.begin(9600);
pinMode(13,OUTPUT );
}
void loop() {
// put your main code here, to run repeatedly:
  int v=analogRead(A0);
Serial.println(v);
delay(1000);
if(v>600)
{
digitalWrite(13, HIGH);
Serial.println("motion detected.");
}
else
{
digitalWrite(13, LOW) ;
}
}
```

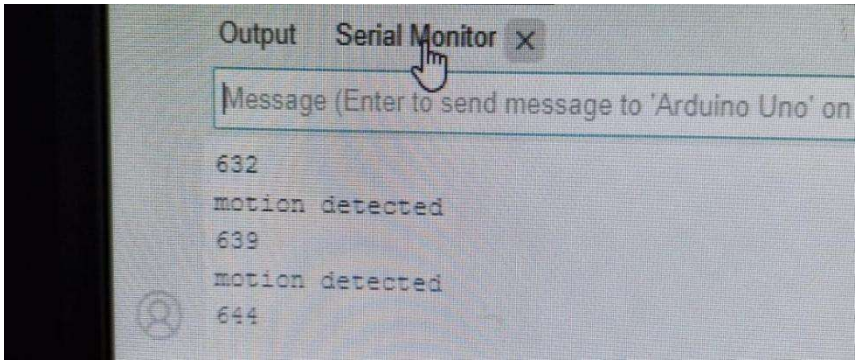
Output:

Input :



Output:





Result:

Thus, the objects have been identified successfully using PIR sensor.

EXP: 5 **Measure the moisture level of soil using a soil moisture sensor**

AIM:

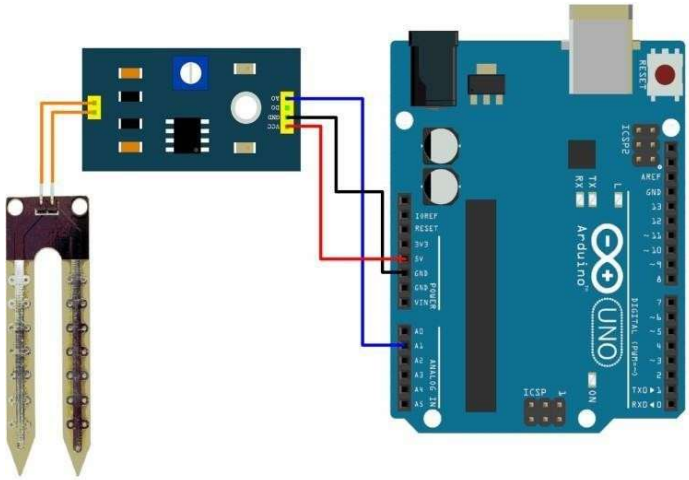
The aim of this experiment is to develop a system that can measure the moisture level of soil using a soil moisture sensor and Arduino Uno. The system control the sensor and to display the moisture level.

Algorithm:

The algorithm for the system is as follows:

1. The soil moisture sensor will be used to measure the moisture level of the soil.
2. The program will read the data from the sensor and display it on the screen.
3. The user will be able to see the moisture level of the soil and take action accordingly.
4. Connect the soil moisture sensor to the Arduino.
5. Write the program.
6. Run the program.

Circuit Diagram:

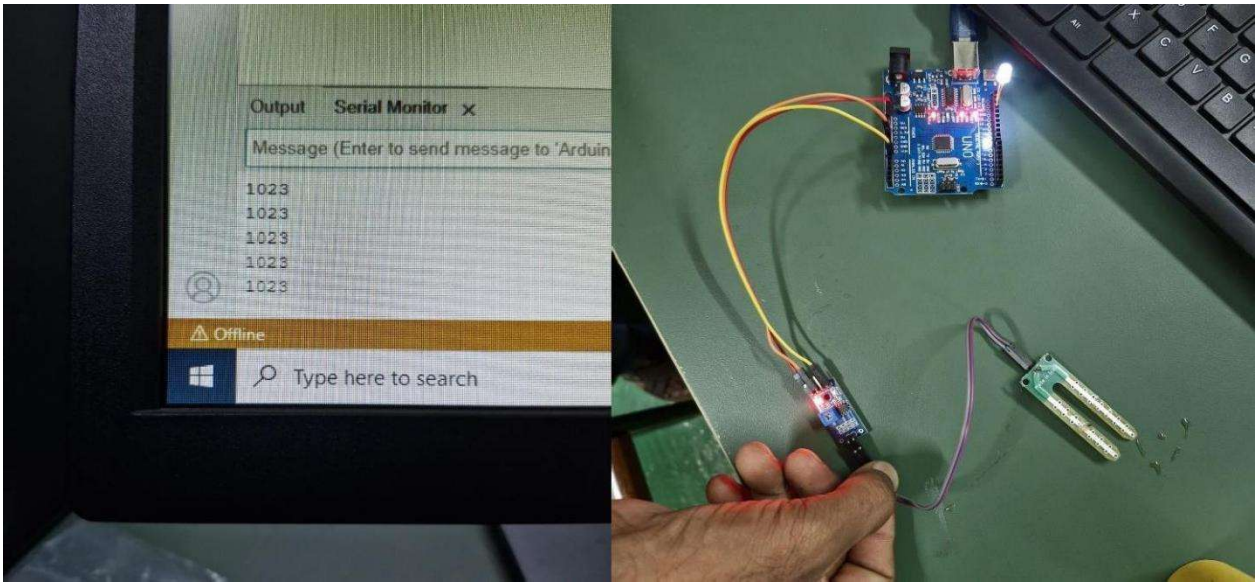


Program:

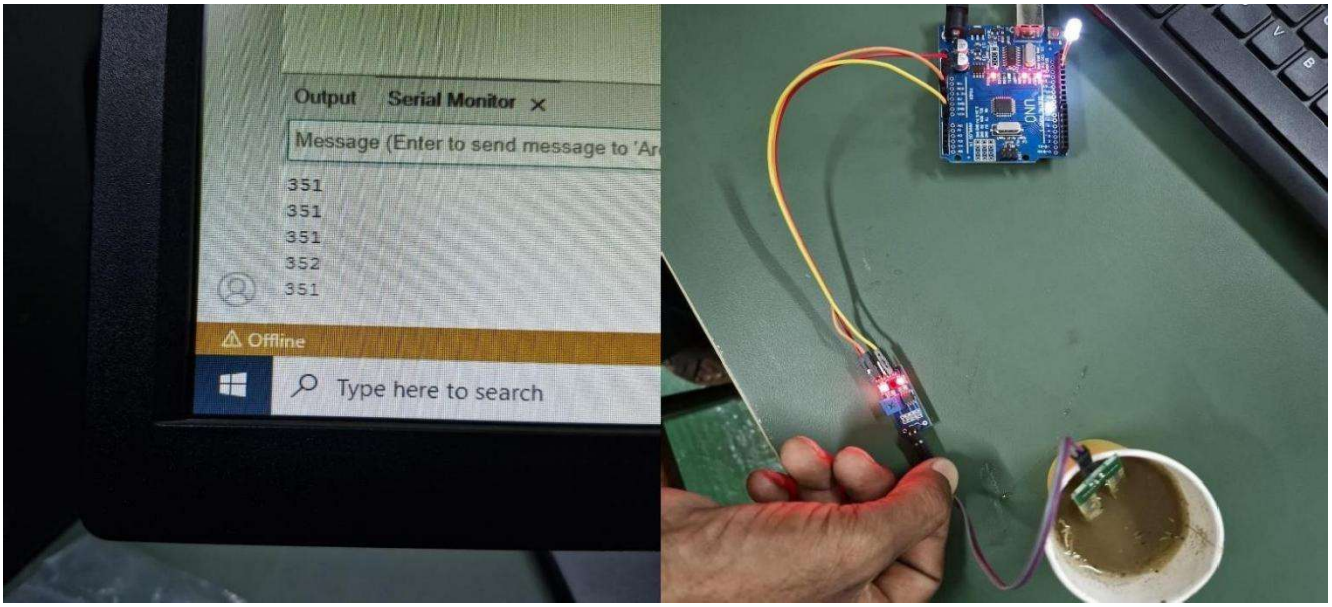
```
int sensor = A0;
void setup() {
  // put your setup code here, to run once:
pinMode(sensor,INPUT);
Serial.begin(9400);
}

void loop() {
  // put your main code here , to run repeatedly:
int x = analogRead(sensor);
Serial.println(x);
}
```

Output:

Input connection:



Output connection:



Result:

Thus, the moisture level of soil has been successfully measured using soil moisture sensor.

EXP : 6 **Measure the distance between an ultrasonic sensor and an obstacle**

AIM:

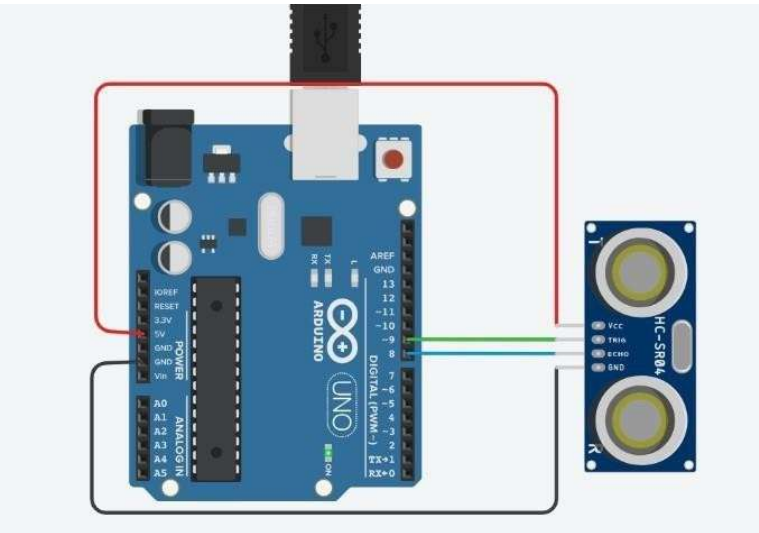
The aim of this experiment is to develop a system that can measure the distance between an ultrasonic sensor and an obstacle using Arduino Uno. The system control the sensor and to display the distance.

Algorithm:

The algorithm for the system is as follows:

1. The ultrasonic sensor will be used to send a sound wave to the obstacle.
2. The sound wave will bounce back to the sensor.
3. The program will measure the time it takes for the sound wave to travelto the obstacle and back.
4. The distance to the obstacle can be calculated using the following formula:
5. distance = (speed of sound * time) / 2
6. Connect the ultrasonic sensor to the Arduino.
7. Write the program.
8. Run the program.

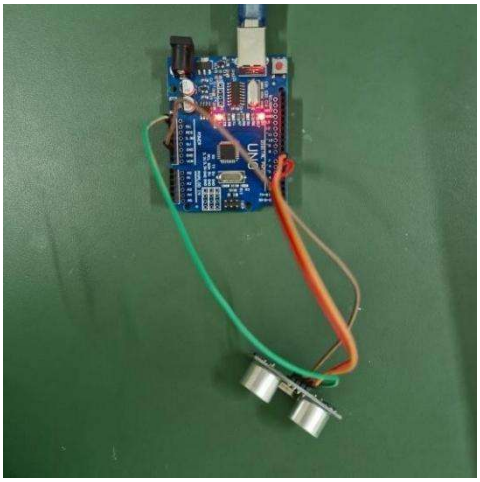
Circuit Diagram:

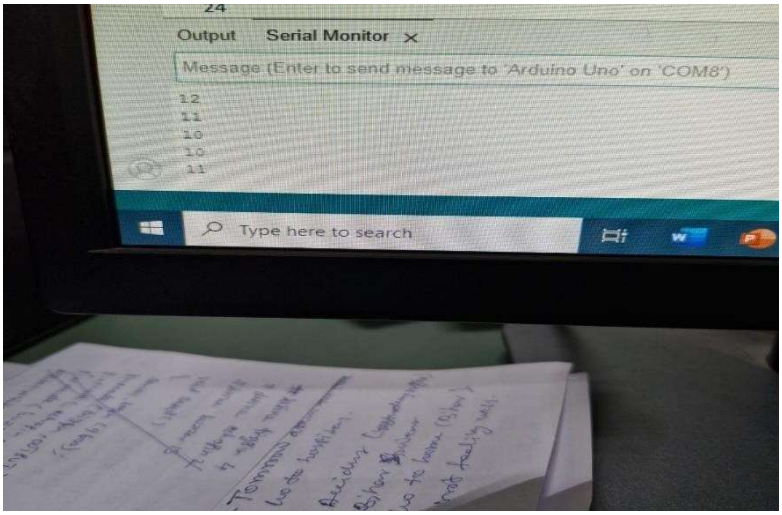


Program:

```
int trig = 7;
int echo = 6;
void setup() {
// put your setup code here, to run once:
Serial.begin(9600);
pinMode(7, 0UTPUT);
PinMode(6, INPUT);
}

void loop() {
// put your main code here, to run repeatedly:
digitalWrite(trig, LOW);
delayMicroseconds(2);
digitalWrite(trig, HIGH);
delayMicroseconds(100);
digitalWrite(trig, LOW);
timeInMicro = pulseIn(echo, HIGH) ;
distanceInCm = timeInMicro/29/2;
Serial.println(distanceInCm);
delay(1000);

}
```

Output:





Result:

Thus, the distance has been successfully measured using ultra sonic sensor.

EXP: 7 Identify the leakage of gas/smoke in the environment

AIM:

The aim of this experiment is to develop a system that can identify the leakage of gas/smoke in the environment using Arduino Uno. The system control the sensor and to display an alert if gas/smoke is detected.

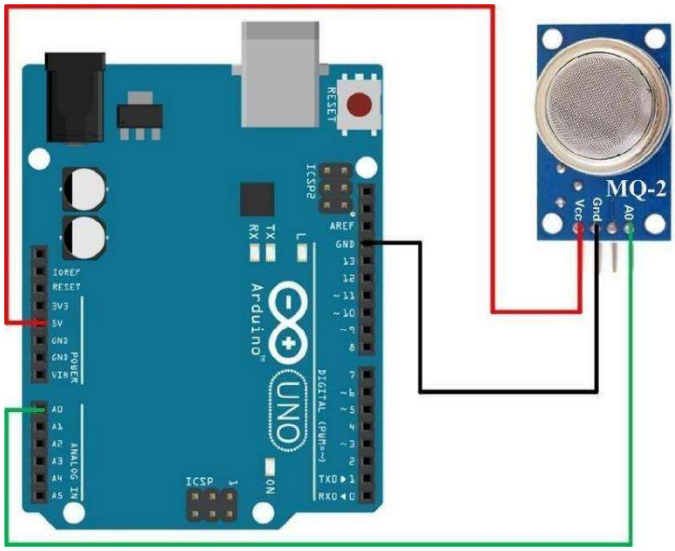
Algorithm:

The algorithm for the system is as follows:

1. The gas/smoke sensor will be used to detect the presence of gas/smoke in the environment.
2. The program will read the data from the sensor and compare it to a threshold value.
3. If the gas/smoke level is above the threshold value, an alert will be displayed.

4. Connect the gas/smoke sensor to the Arduino.
5. Write the program.
6. Run the program.

Circuit Diagram

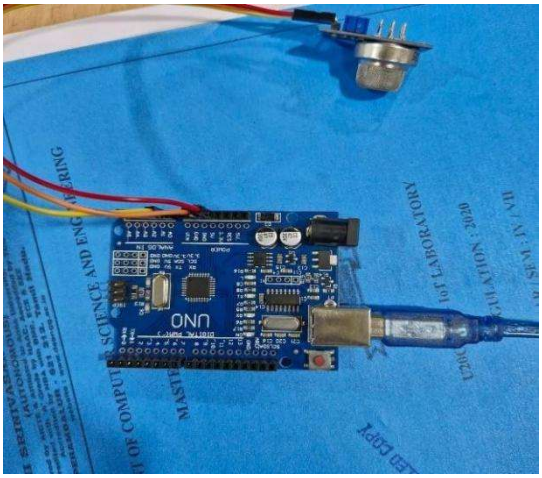


Program:

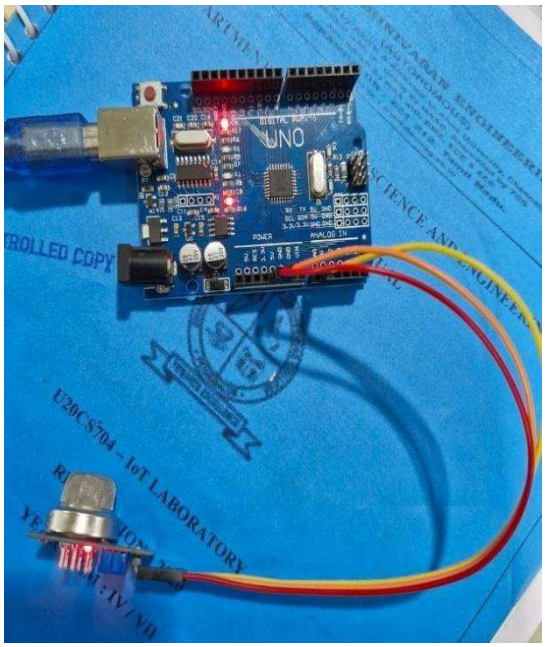
```
void setup() {
// put your setup code here, to run once:
Serial.begin(9600);
}


void loop() {
// put your main code here, to run repeatedly:
int sensorValue=analogRead(A0);
Serial.println(sensorValue);
delay(1000);

}
```

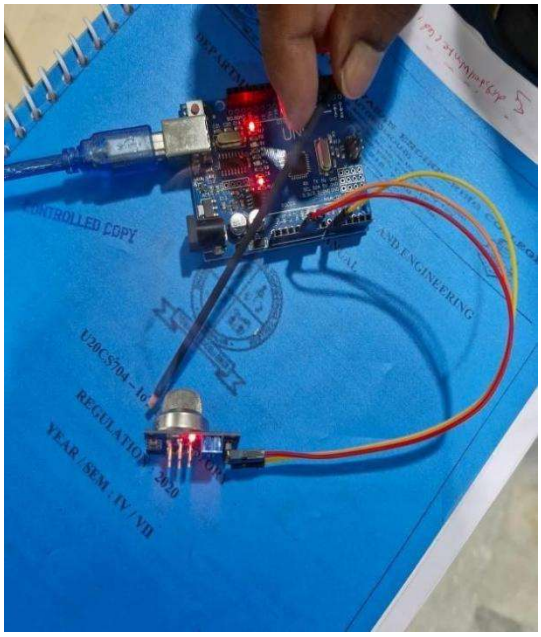
Output:

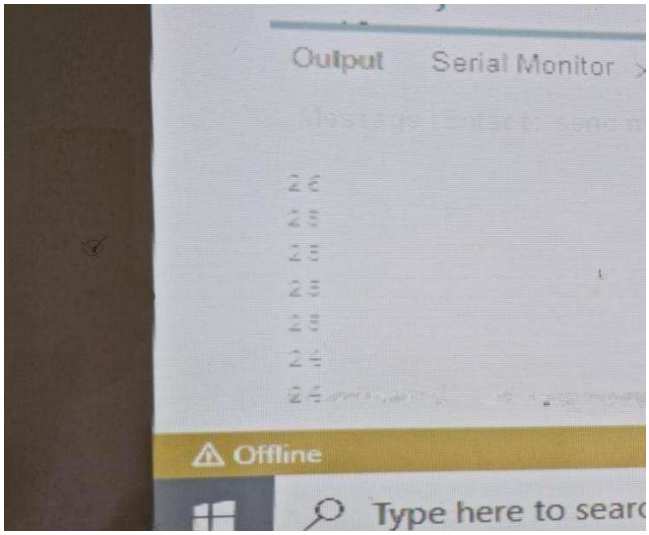
Input connection:



Output connection:







Result:

Thus, the gas / smoke sensor has been successfully implemented.

EXP: 8 Measure the humidity and moisture value of the environment

AIM:

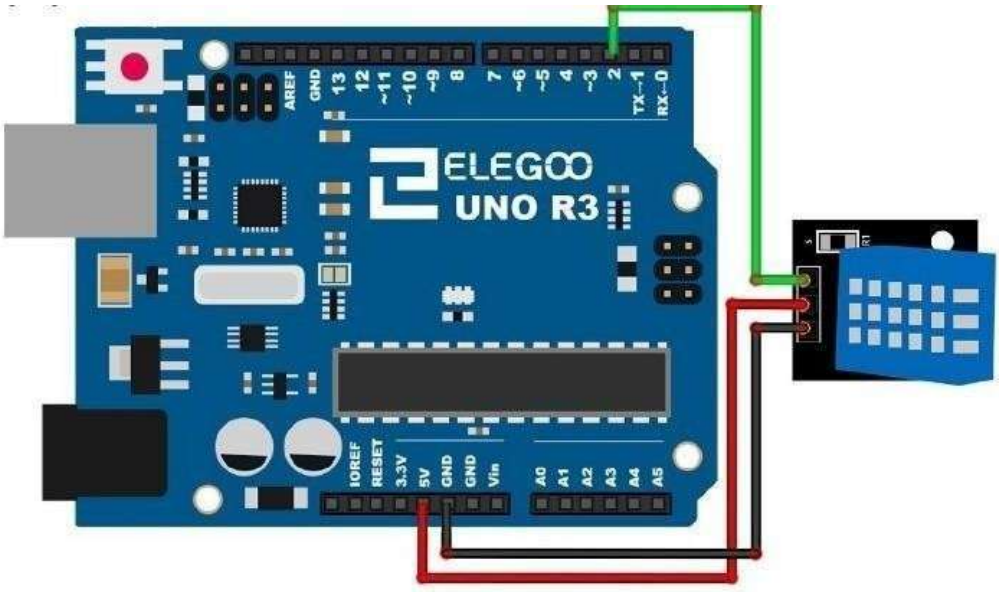
The aim of this experiment is to develop a system that can measure the humidity and moisture value of the environment using Arduino Uno. The system control the sensor and to display the humidity and moisture values.

Algorithm:

The algorithm for the system is as follows:

1. The humidity and moisture sensor will be used to measure the humidity and moisture levels in the environment.
2. The program will read the data from the sensor and display it on the screen.
3. The user will be able to see the humidity and moisture levels and take action accordingly.
4. Connect the humidity and moisture sensor to the Arduino.
5. Write the program.
6. Run the program.

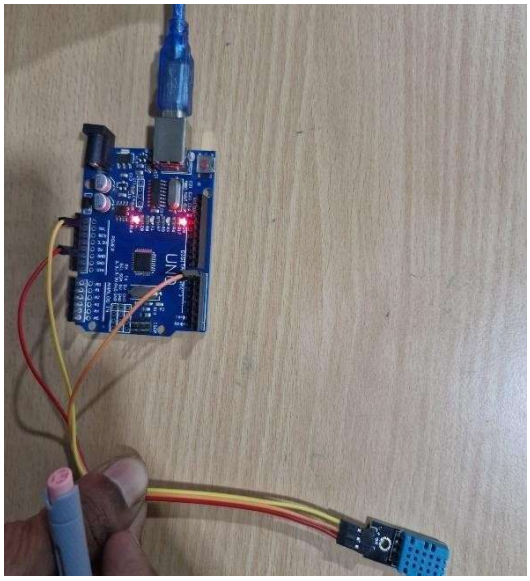
Circuit Diagram:

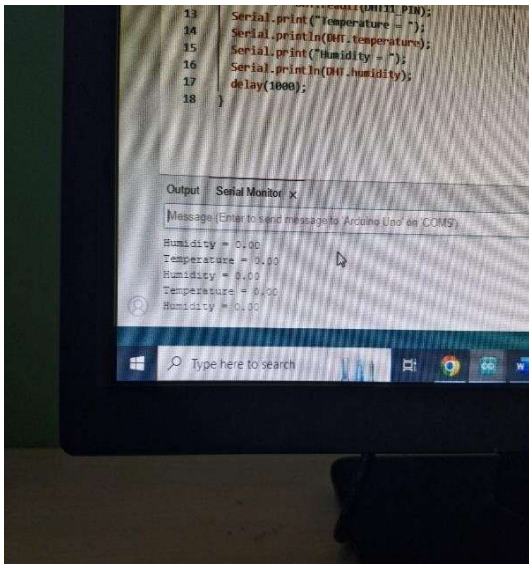


Program:

```
#include <dht.h>
dht DHT;
#define DHT11_PIN 7

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

void loop() {
int chk = DHT.read11(DHT11_PIN);
Serial.print("Temperature = ");
Serial.println(DHT.temperature);
Serial.print("Humidity = ");
Serial.println(DHT.humidity);
delay(1000);
}
```

Output:





Result:

Thus, the humidity and moisture level has been successfully measured.

EXP : 9 Control a LED using a relay switch

AIM:

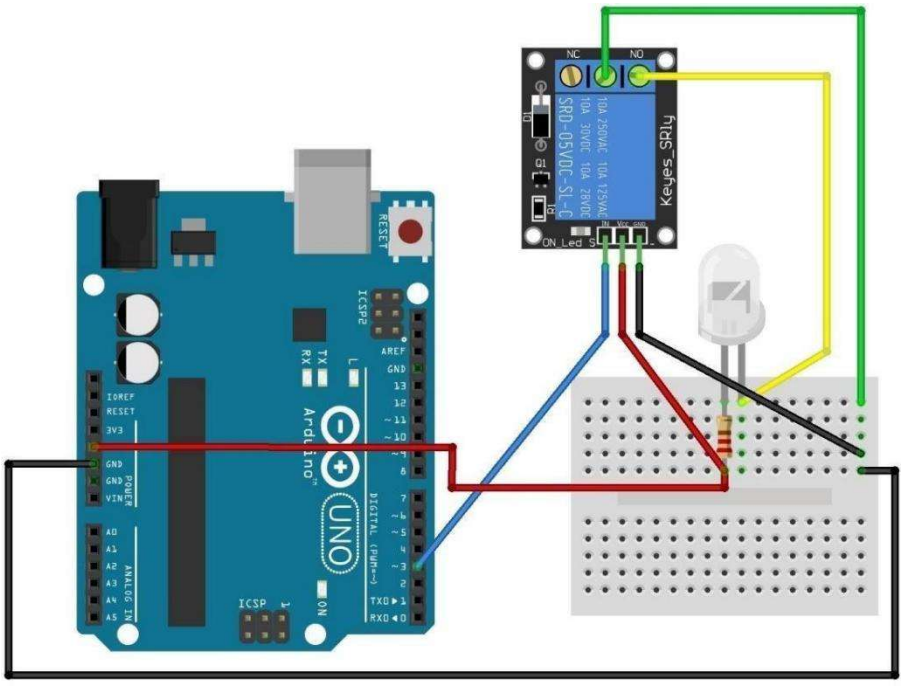
The aim of this experiment is to develop a system that can control a LED using a relay switch using Arduino Uno. The system controls the relay switch and to turn the LED on and off.

Algorithm:

The algorithm for the system is as follows:

1. The relay switch will be used to control the flow of current to the LED.
2. The program will send a signal to the relay switch to turn the LED on or off.
3. The LED will turn on or off depending on the signal from the program.
4. Connect the relay switch to the Arduino.
5. Write the program.
6. Run the program.

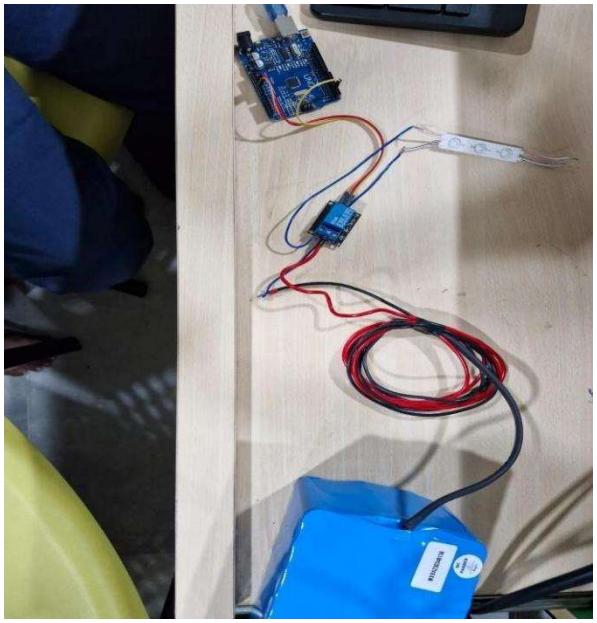
Circuit Diagram

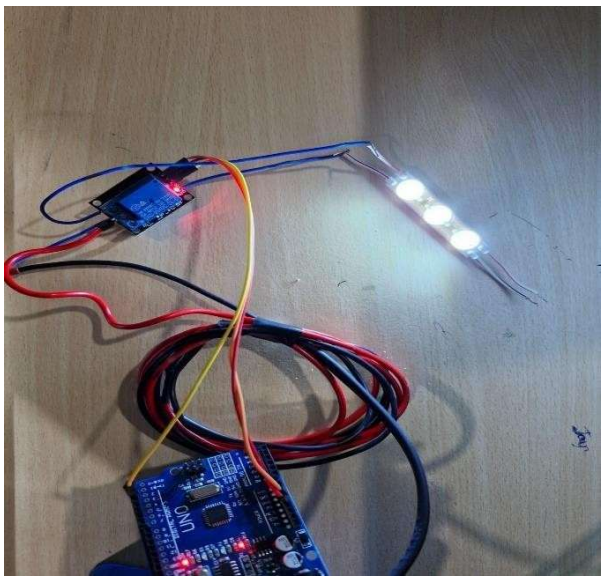


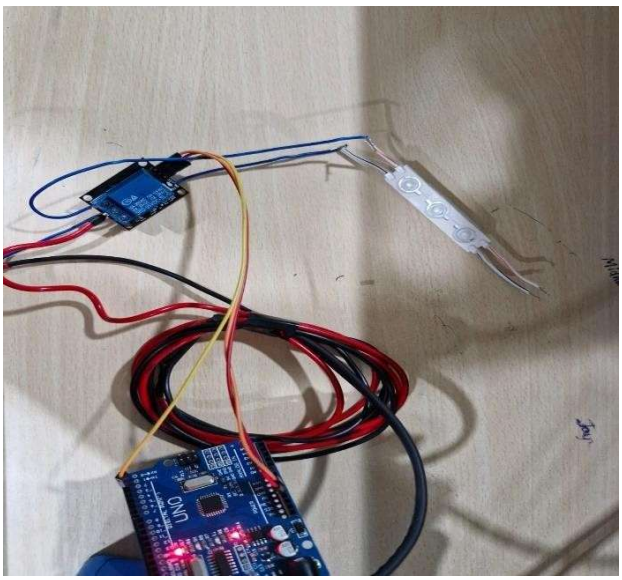
Program:

```
#define RELAY_PIN 2 // Relay control pin connected to digital pin 2

void setup() {

pinMode(RELAY_PIN, OUTPUT);
}


void  loop() {

digitalWrite(RELAY_PIN, HIGH);
delay(10000);
digitalWrite(RELAY_PIN, LOW);
delay(10000);
}
```

Output:







Result:

Thus, the LED has been successfully controlled using relay switch.

EXP: 10 Identify the Rain using the Rain Sensor

AIM:

The aim of this experiment is to develop a system that can identify the rain fall using Arduino Uno. The system control the sensor and to display the rain fall information.

Algorithm:

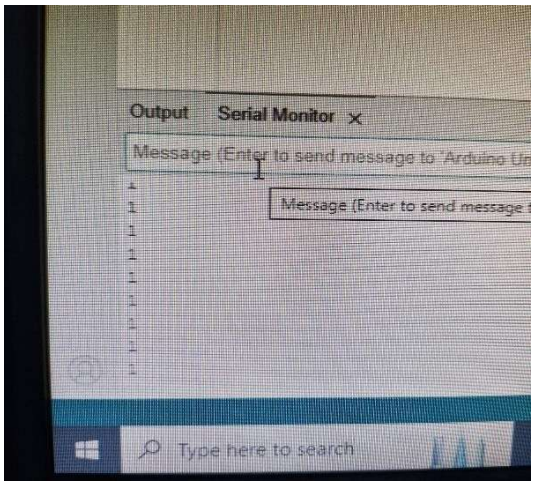
The algorithm for the system is as follows:

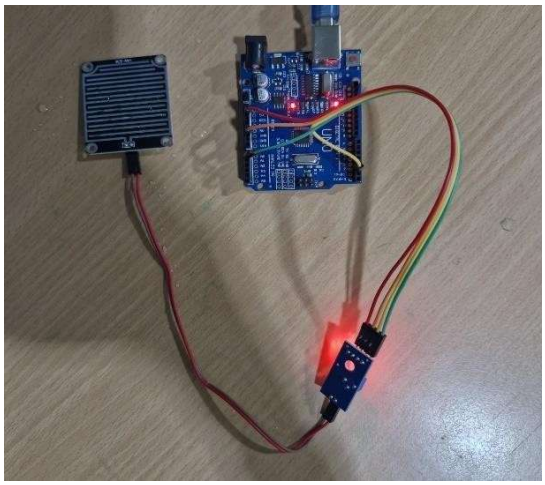
1. The rain sensor will be used to identify the rainfall in the environment.
2. The program will read the data from the sensor and display it on the screen.
3. The user will be able to see the rainfall and take action accordingly.
4. Connect the rain sensor to the Arduino.
5. Write the program.
6. Run the program.

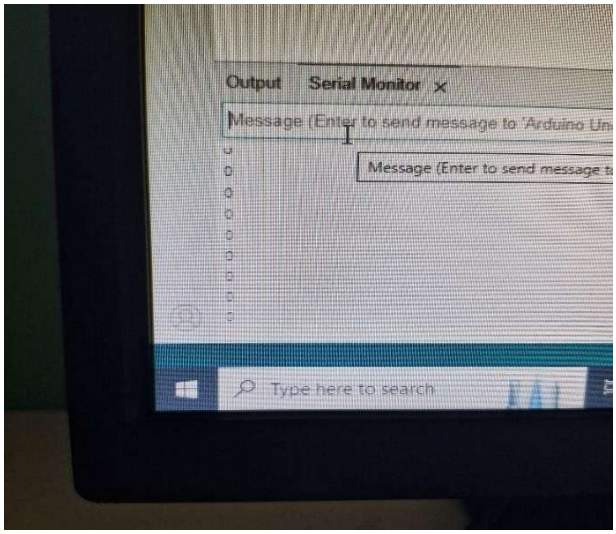
Program:

```
 Int Sensorpin = 2;
void setup() {
// put your setup code here, to run once:
pinMode(Sensorpin, INPUT);
Serial.begin(9600);
}

void loop() {
// put your main code here, to run repeatedly:
int sensorData = digitalRead(Sensorpin);
delay(1000);
}
```

Output:







Result:

Thus, the rainfall has been successfully identified.

EXP : 10 **Line Follower Robot**

AIM:

The aim of this experiment is to develop a line following robot is a basic robot designed to follow a predetermined line or path. The system will use IR transmitters and receivers (photodiodes). They are used to send and receive the lights. When IR rays fall on a white surface, it is reflected towards IR receiver, generating some voltage changes.

Algorithm:

The algorithm for the system is as follows:

STEP 1: If only the right sensor detects a black line, then the robot must steer right to follow the line.

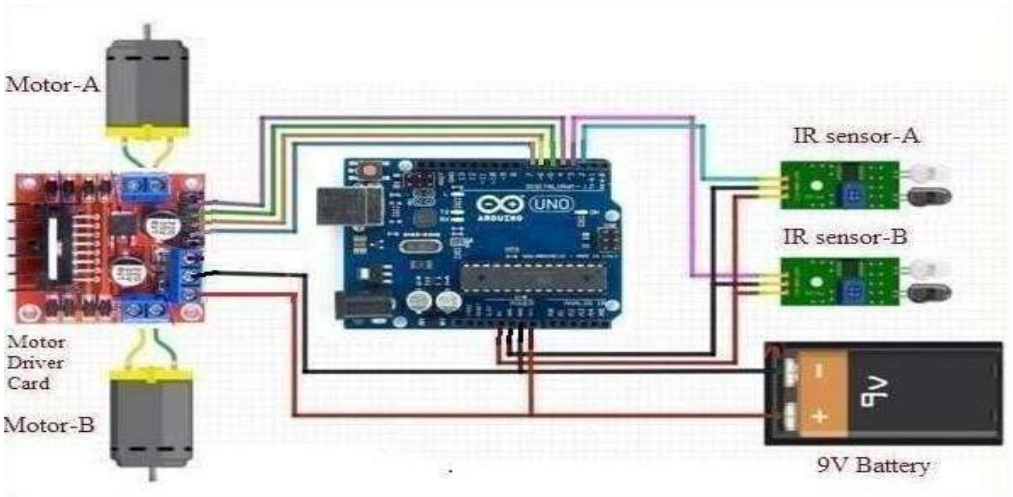
STEP 2: In this case, both the sensors do not detect a black line, so the robot should keep moving forward in the same direction and maintain its current journey path.

STEP 3: This case is a bit tricky. Depending on your game scenario, both the sensors detecting a black line could mean either it is a finish line or it is just a loop which is part of a bigger line-following problem.

STEP 4: To connect the battery with the circuit and place the battery on chassis. I have connected everything with jumper wires. To make a permanent setup, you can directly solder everything together.

STEP 5: That first place robot on a black surface (both sensors should be on top of the black surface) then adjust the variable resistor of IR Module until the output led of IR module become off.

Circuit Diagram:

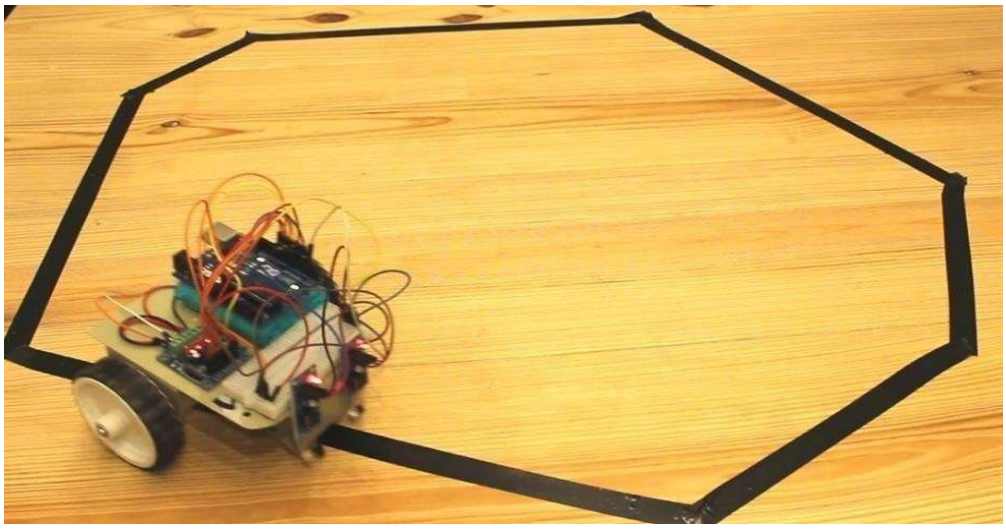


Program:

```
// Define pins for sensors and motors
#define leftSensor 2  // Left IR sensor
#define rightSensor 3 // Right IR sensor
#define leftMotor 9  // Left motor
#define rightMotor 10 // Right motor

void setup() {
  // Set the sensor pins as input
  pinMode(leftSensor, INPUT);
  pinMode(rightSensor, INPUT);

  // Set the motor pins as output
  pinMode(leftMotor, OUTPUT);
  pinMode(rightMotor, OUTPUT);
}

void loop() {
  // Read the sensor values
  int leftValue = digitalRead(leftSensor);
  int rightValue = digitalRead(rightSensor);

  // If both sensors detect the line, move forward
  if (leftValue == LOW && rightValue == LOW) {
    moveForward();
  }
  // If left sensor detects the line, turn left
  else if (leftValue == LOW) {
  turnLeft();
  }
  // If right sensor detects the line, turn right
  else if (rightValue == LOW) {
  turnRight();
  }
  // If no sensors detect the line, stop
  else {
    stopMoving();
  }
}

// Move forward
void moveForward() {
  digitalWrite(leftMotor, HIGH);
  digitalWrite(rightMotor, HIGH);
}

// Turn left
void turnLeft() {
  digitalWrite(leftMotor, LOW);  // Stop left motor
  digitalWrite(rightMotor, HIGH); // Move right motor
}
// Turn right
```

```
void turnRight() {
  digitalWrite(leftMotor, HIGH);  // Move left motor
  digitalWrite(rightMotor, LOW);  // Stop right motor
}

// Stop moving
void stopMoving() {
  digitalWrite(leftMotor, LOW);
  digitalWrite(rightMotor, LOW);
}
```

Output:



Result:

The result of the experiment is that the system was able to a line following robot is a basic robot designed to follow a predetermined line or path using Arduino and Python. The Python program was able to system will use IR transmitters and receivers (photodiodes). They are used to send and receive the lights.

EXP : 11 **SMART PARKING SYSTEM**

AIM:

Smart parking technologies ensure to reduce the number of circling around the streets for finding a parking spot. This ultimately smoothens the traffic flow and minimize the search traffic on streets as much as possible.

Processor:

- Proposing a Customized Automated Vehicle Parking System.
- Suitable for On street, Off street and Multi-level parking Lots.
- Public can know the availability of parking vacancies in Real-time through mobile application.
- Governance bodies can monitor the system from their Remote command and Control center.
- Parking fees will be charged as per parking time.

Algorithm:

The algorithm for the system is as follows:

Step1: Extract file

Step2: Copy the main project folder

Step3: Paste in xampp/htdocs/

Step4: Open a browser and go to URL http://localhost/phpmyadmin/

Step5: Then, click on the databases tab

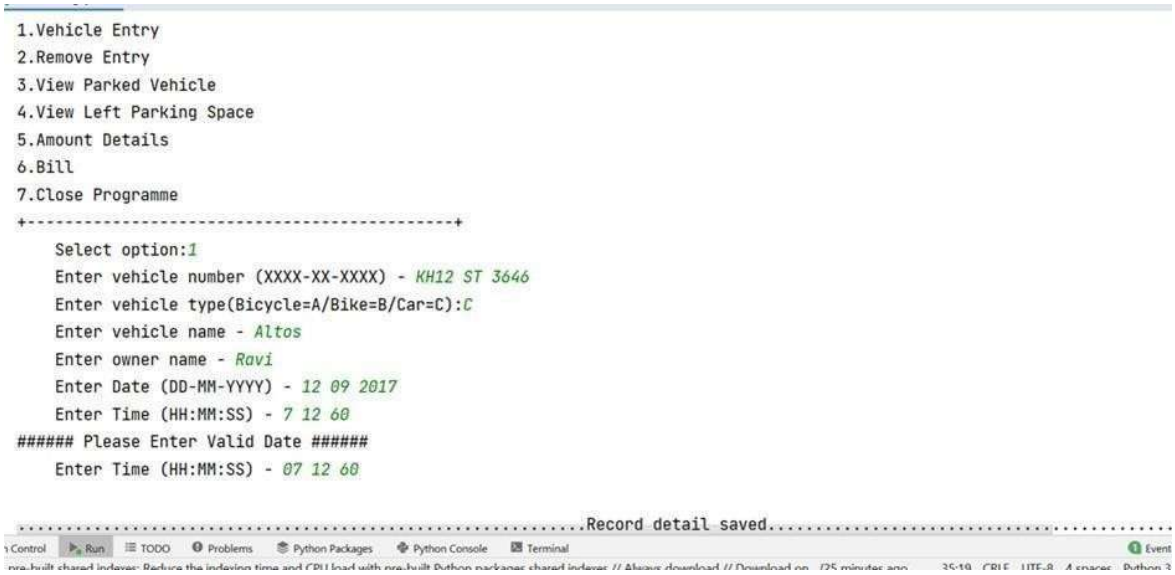
Step6: Create a database naming “smart_users” and then click on the import tab.

Program:

```
// Define pins for ultrasonic sensors and LEDs
#define trigPin1 9  // Trig pin for Slot 1
#define echoPin1 2  // Echo pin for Slot 1
#define ledPin1 12  // LED pin for Slot 1

#define trigPin2 10  // Trig pin for Slot 2
#define echoPin2 3  // Echo pin for Slot 2
#define ledPin2 13  // LED pin for Slot 2

// Variables to store measured distance
long duration1, duration2;
int distance1, distance2;

void setup() {
  // Set up serial communication for monitoring
  Serial.begin(9600);

  // Set pins for ultrasonic sensors and LEDs
  pinMode(trigPin1, OUTPUT);
  pinMode(echoPin1, INPUT);
  pinMode(ledPin1, OUTPUT);

  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);
  pinMode(ledPin2, OUTPUT);
}

void loop() {
  // Check Slot 1
  distance1 = getDistance(trigPin1, echoPin1);
  if (distance1 < 20) { // If a vehicle is detected (within 20 cm)
    digitalWrite(ledPin1, HIGH); // Turn on the LED (Slot 1 occupied)
  } else {
    digitalWrite(ledPin1, LOW);  // Turn off the LED (Slot 1 free)
  }

  // Check Slot 2
  distance2 = getDistance(trigPin2, echoPin2);
  if (distance2 < 20) { // If a vehicle is detected (within 20 cm)
    digitalWrite(ledPin2, HIGH); // Turn on the LED (Slot 2 occupied)
  } else {
    digitalWrite(ledPin2, LOW);  // Turn off the LED (Slot 2 free)
  }

  // Print the distances to the Serial Monitor
  Serial.print("Slot 1 distance: ");
  Serial.println(distance1);
  Serial.print("Slot 2 distance: ");
  Serial.println(distance2);
```

```
  // Wait a bit before checking again
  delay(1000);
}

// Function to get distance from ultrasonic sensor
int getDistance(int trigPin, int echoPin) {
  long duration;
  int distance;

  // Clear the trigPin by setting it LOW for 2 microseconds
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  // Trigger the sensor by setting trigPin HIGH for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read the echoPin, which gives the time for the sound wave to return
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance (time / 2 * speed of sound: 34300 cm/s)
  distance = duration * 0.034 / 2;

  return distance;
}
```

Output:


1.Vehicle Entry
2.Remove Entry
3.View Parked Vehicle
4.View Left Parking Space
5.Amount Details
6.Bill
7.Close Programme
+---+
Select option:1
Enter vehicle number (XXXX-XX-XXXX) - KH12 ST 3646
Enter vehicle type(Bicycle=A/Bike=B/Car=C):C
Enter vehicle name - Altos
Enter owner name - Ravi
Enter Date (DD-MM-YYYY) - 12 09 2017
Enter Time (HH:MM:SS) - 7 12 60
Please Enter Valid Date
Enter Time (HH:MM:SS) - 07 12 60
Record detail saved
Control
Run
TODO
Problems
Python Packages
Python Console
Terminal
Event

Result:

The driver can view available parking slots directly from their smartphone with such a solution.

EXP : 12 **SMART WASTE MANAGEMENT SYSTEM**

AIM:

The aim of this experiment is to develop a system that can be a smart waste management system. The system will be use a python program to control the sensor and to display the smart waste management system.

Algorithm:

The algorithm for the system is as follows:

- The proposed system would be able to automate the solid waste monitoring process and management of the overall collection process using IOT (Internet of Things).
- The Proposed system consists of main subsystems namely Smart Trash System (STS) and Smart Monitoring and Controlling Hut (SMCH).
- In the proposed system, whenever the waste bin gets filled this is acknowledged by placing the circuit at the waste bin, which transmits it to the receiver at the desired place in the area or spot.
- In the proposed system, the received signal indicates the waste bin status at the monitoring and controlling system.

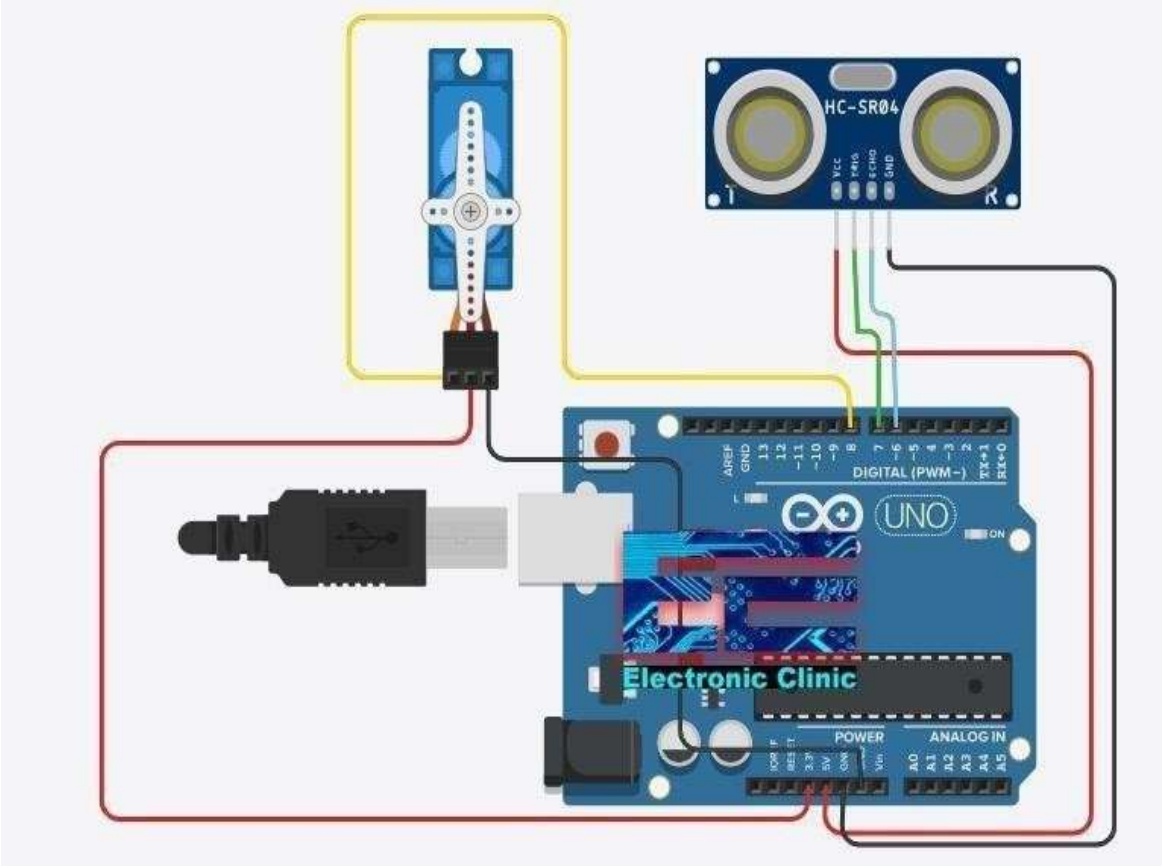
Hardware Requirements:

- Arduino UNO
- Ultrasonic sensor
- IR sensor
- Moisture sensor
- DC motor

SOFTWARE REQUIREMENTS:

- Arduino IDE

Circuit Diagram:



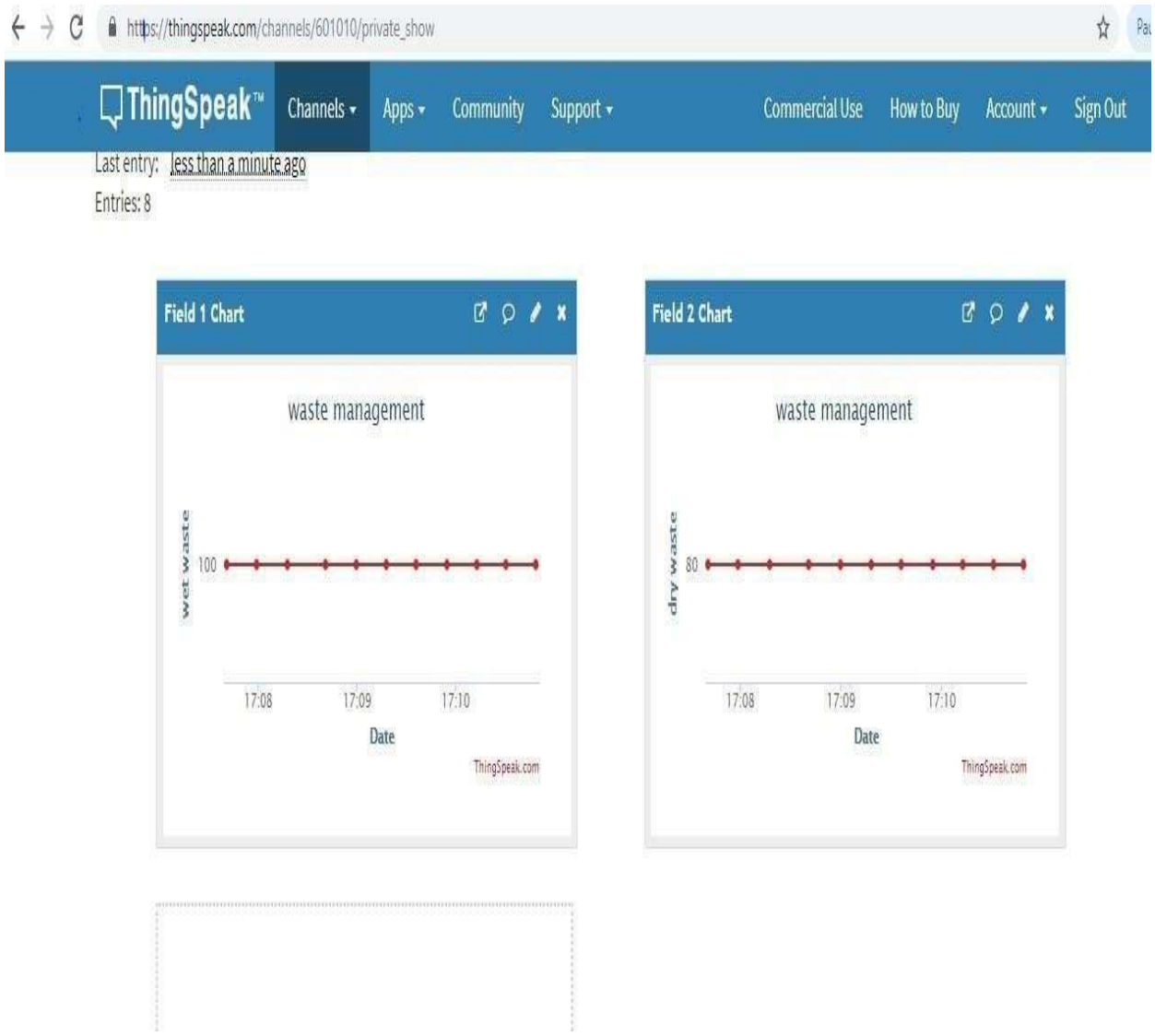
- Ultrasonic sensor Sensors measure distances by using ultrasonic waves. The sensor emits an ultrasonic wave and receives the reflected wave back from the target.

Program:

```
// Define pins for ultrasonic sensor and LED/Buzzer
#define trigPin 9
#define echoPin 10
#define ledPin 13  // LED or buzzer pin for full-bin alert

// Define the maximum capacity of the bin (in cm, for example)
#define MAX_CAPACITY 20  // Max level when bin is considered full (e.g., 20 cm from sensor)

// Variables to store measured distance
long duration;
int distance;

void setup() {
  // Initialize serial communication (optional, for monitoring)
  Serial.begin(9600);
```

```
  // Set pins for ultrasonic sensor and LED/Buzzer
  pinMode(trigPin, OUTPUT);

        pinMode(echoPin, INPUT);
        pinMode(ledPin, OUTPUT);
}

void loop() {
  // Get the current bin waste level (measured distance from sensor)
  distance = getDistance();

  // Display the current distance (optional)
  Serial.print("Distance: ");
  Serial.println(distance);

  // Check if the bin is full
  if (distance <= MAX_CAPACITY) {
    // If distance is less than or equal to the max capacity, bin is full
    alertFull();
  } else {
    // Otherwise, turn off the alert (LED/Buzzer)
    digitalWrite(ledPin, LOW);
  }

  // Wait before the next measurement
  delay(1000);
}

// Function to get the distance from the ultrasonic sensor
int getDistance() {
  // Clear the trigPin by setting it LOW for 2 microseconds
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  // Trigger the sensor by setting the trigPin HIGH for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read the echoPin, which gives the time for the sound wave to return
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance (time / 2 * speed of sound: 34300 cm/s)
  int distance = duration * 0.034 / 2;

  return distance;
}

// Function to trigger the full-bin alert (LED/Buzzer)
void alertFull() {
  Serial.println("Bin is full. Needs emptying!");
  digitalWrite(ledPin, HIGH);  // Turn on LED/Buzzer to indicate the bin is full
}
```

Output:


https://thingspeak.com/channels/601010/private_show
ThingSpeak™
Channels
Apps
Community
Support
Commercial Use
How to Buy
Account
Sign Out
Last entry: less than a minute ago
Entries: 8
Field 1 Chart
waste management
wet waste
100
17:08
17:09
17:10
Date
ThingSpeak.com
Field 2 Chart
waste management
dry waste
80
17:08
17:09
17:10
Date
ThingSpeak.com

Result:

The result of the experiment is that the system was able to measure the smart waste management system using Arduino and Python. The python program was able to measure the thing speak cloud.

EXP : 13 SMART WEATHER MONITORING SYSTEM USING ARDUINO

AIM:

To write a program to get smart weather monitoring system using Arduino.

Components Required:

- Nodemcu ESP8266 board x 1
- Rain sensor x 1
- DHT11 sensor x 1
- LDR sensor x 1
- LCD x 1
- I2C module x

Algorithm:

The algorithm for the system is as follows:

STEP 1: Start the process.

STEP 2: Start Arduino 1.8.8

STEP 3: Include the DHT library to the Arduino software.

STEP 4: Then enter the coding in Arduino software.

STEP 5: Complete the coding in Arduino.

STEP 6: In Arduino board connect VCC to the power supply 5V and connect SIG to digital signal DT and connect SND to ground GND using jumper wires.

STEP 7: Connect the arduino board with USB cable to the system.

STEP 8: Select tools Selected.

STEP 9: Upload the coding to arduino board. Then the output will be displayed in the serial monitor.

STEP 10: Stop the process.

Program:

```
#include <LiquidCrystal_I2C.h>
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <DHT.h>

LiquidCrystal_I2C lcd(0x27, 16, 2); // Initialize I2C LCD (address 0x27, 16x2 LCD)
DHT dht(D3, DHT11); // DHT11 sensor at pin D3
BlynkTimer timer; // Blynk Timer for periodic updates

char auth[] = "YourAuthToken"; // Blynk authentication token
char ssid[] = "YourWiFiSSID"; // WiFi SSID
char pass[] = "YourWiFiPassword"; // WiFi Password

void weather() {
 // Read temperature and humidity from DHT11 sensor
 float h = dht.readHumidity();
 float t = dht.readTemperature();

 // Read rainfall from analog input A0
 int r = analogRead(A0);
 r = map(r, 0, 1023, 100, 0); // Map rainfall reading to percentage (0-100)

 // Light level detection (for simplicity, using D4 as a placeholder)
 bool l = digitalRead(D4);

 // Check if the DHT sensor failed to read
 if (isnan(h) || isnan(t)) {
  Serial.println("Failed to read from DHT sensor!");
  return;
 }

 // Send data to Blynk virtual pins
 Blynk.virtualWrite(V0, t); // Send temperature to Blynk
 Blynk.virtualWrite(V1, h); // Send humidity to Blynk
 Blynk.virtualWrite(V2, r); // Send rainfall status to Blynk

 // Control an LED on the Blynk app (V3) based on light level
 WidgetLED led1(V3);
 if (l == 0) {
  led1.on(); // Light level is high
  lcd.setCursor(9, 1);
  lcd.print("L: High ");
 } else {
  led1.off(); // Light level is low
  lcd.setCursor(9, 1);
  lcd.print("L: Low  ");
 }
```

```
  // Display data on the LCD
  lcd.setCursor(0, 0);
  lcd.print("T: ");
  lcd.print(t);  // Temperature
  lcd.print("C");

  lcd.setCursor(0, 1);
  lcd.print("H: ");
  lcd.print(h);  // Humidity
  lcd.print("%");

  lcd.setCursor(9, 0);
  lcd.print("R: ");
  lcd.print(r);  // Rain percentage
  lcd.print("%  ");
}

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

  // Initialize LCD and backlight
  lcd.init();
  lcd.backlight();

  // Initialize DHT11 sensor
  dht.begin();

  // Initialize Blynk
  Blynk.begin(auth, ssid, pass);

  // Setup weather function to run every 10 seconds
  timer.setInterval(10000L, weather);  // Update weather every 10 seconds
}

void loop() {
  // Run Blynk and the timer
  Blynk.run();
  timer.run();
}
```

Output:


7:15 pm
Wether monitoring
TEMPERATURE
V0
25.8
HUMIDITY
V1
82
RAINFALL
V2
LIGHT

Result:

Smart weather monitoring system using Arduino was implemented successfully.