

DHANALAKSHMI SRINIVASAN ENGINEERING COLLEGE

(AUTONOMOUS)

(Approved by AICTE & Affiliated to Anna University, Chennai)

Accredited with 'A' Grade by NAAC, Accredited by TCS

Accredited by NBA with BME, ECE & EEE

PERAMBALUR - 621 212. Tamil Nadu.

website : www.dsengg.ac.in



U20CS704

INTERNET OF THINGS

LABORATORY

B.E/B.Tech Degree Practical Examination

Name

Branch

Semester

Reg.No.

Academic Year

DHANALAKSHMI SRINIVASAN ENGINEERING COLLEGE
(AUTONOMOUS)



(Approved by AICTE & Affiliated to Anna University, Chennai)
Accredited with 'A' Grade by NAAC, Accredited by TCS
Accredited by NBA with BME, ECE & EEE
PERAMBALUR - 621 212. Tamil Nadu.
website : www.dsengg.ac.in



Name

Reg.No.

Sem. /Branch

Sub.Code/Subject

Certified that this is the Bonafide record of the Practical done for the above subject in the
Laboratory during the period

.....

Staff in Charge

Head of the Department

Submitted for the University Practical Examination held at
DHANALAKSHMI SRINIVASAN ENGINEERING COLLEGE
(AUTONOMOUS), PERAMBALUR on.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

TABLE OF CONTENTS

[illegible]

DHANALAKSHMI SRINIVASAN ENGINEERING COLLEGE



(AUTONOMOUS)
(Approved by AICTE & Affiliated to Anna University, Chennai)
Accredited with 'A' Grade by NAAC, Accredited by TCS
Accredited by NBA with BME, ECE & EEE
PERAMBALUR - 621 212. Tamil Nadu.
website : www.dsengg.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Vision and Mission of the Department:

Vision

To produce globally competent, socially responsible professionals in the field of Computer Science and Engineering.

Mission

- M1: Impart high quality experiential learning to get expertise in modern software tools
- M2: Inculcate industry exposure and build inter disciplinary research skills.
- M3: Mould the students to become Software Professionals, Researchers and Entrepreneurs by providing advanced laboratories.
- M4: Acquire Innovative skills and promote lifelong learning with a sense of societal and ethical responsibilities

State the Program Educational Objectives (PEOs)

PEO 1: Graduates of the programme will develop proficiency in identifying, formulating, and resolving complex computing problems.

PEO 2: Graduates of the programme will achieve successful careers in the field of computer science and engineering, pursue advanced degrees, or demonstrate entrepreneurial success.

PEO 3: Graduates of the programme will cultivate effective communication skills, teamwork abilities, ethical values, and leadership qualities for professional engagement in industry and research organizations.

EXP : 1

Date:

Arduino Installation

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 should also 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 the download zipped file. Copy the folder to a suitable place such as your desktop.



INSTALLING THE ARDUINO IDE ON Windows 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":

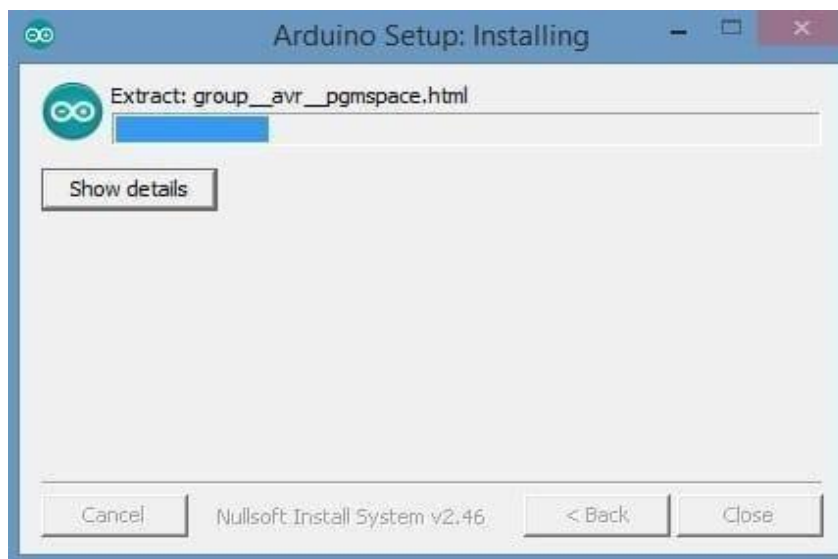


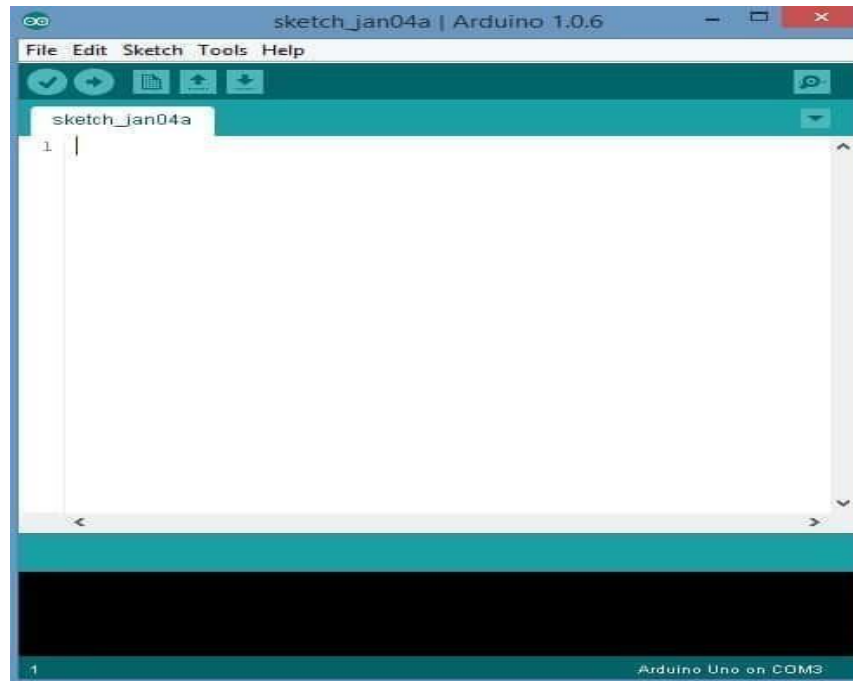
6. Select which folder to install the program to, then click “Install”:



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

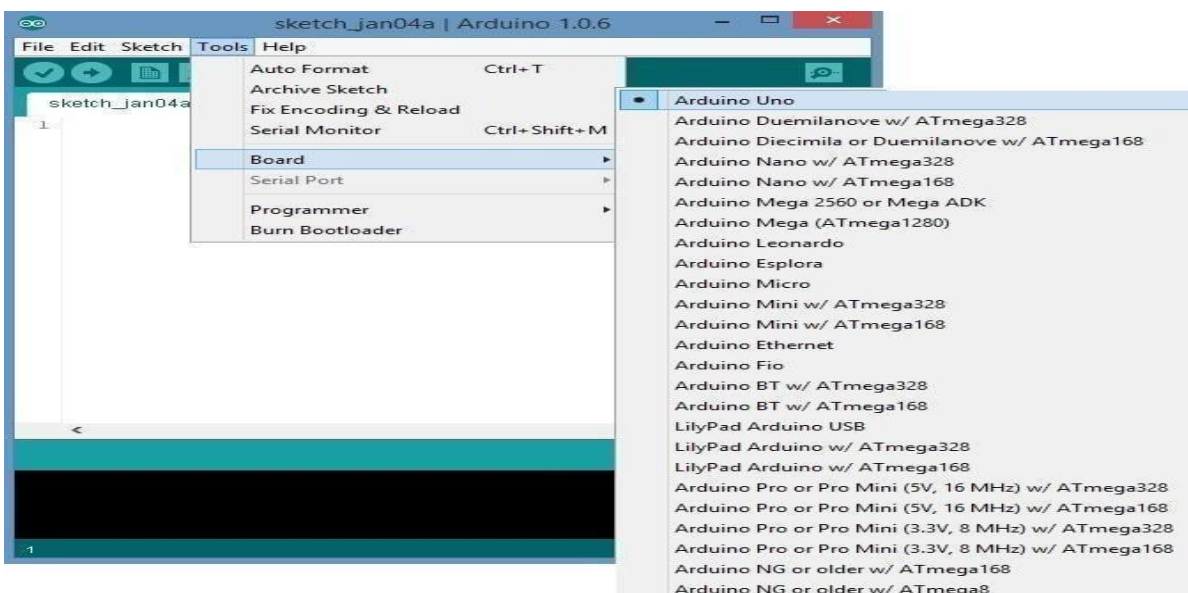
8. Now find the Arduino shortcut on your Desktop and click on it. The IDE will open up and you'll see the code editor:





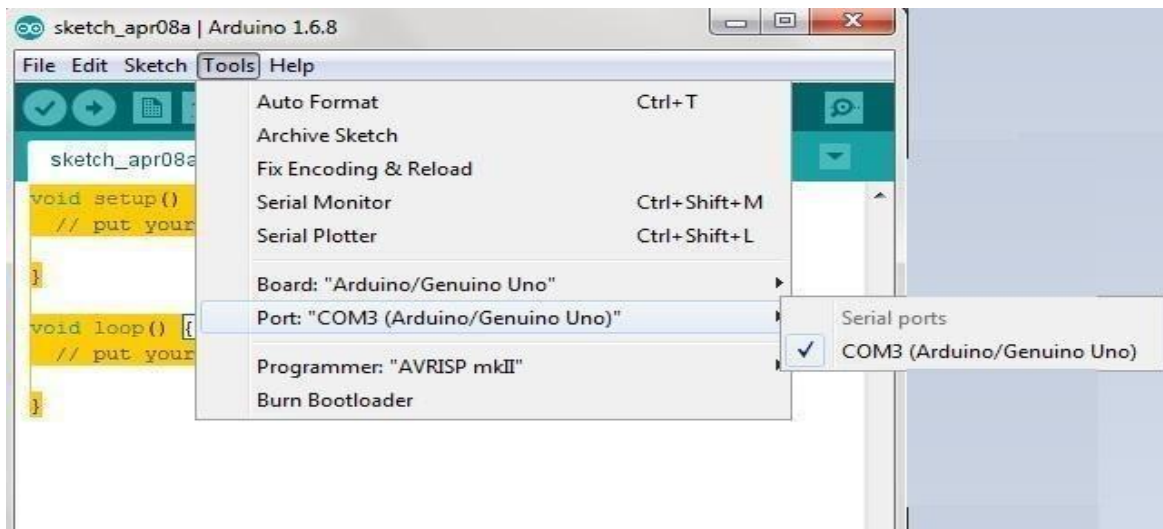
CONFIGURING THE ARDUINO IDE

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”.



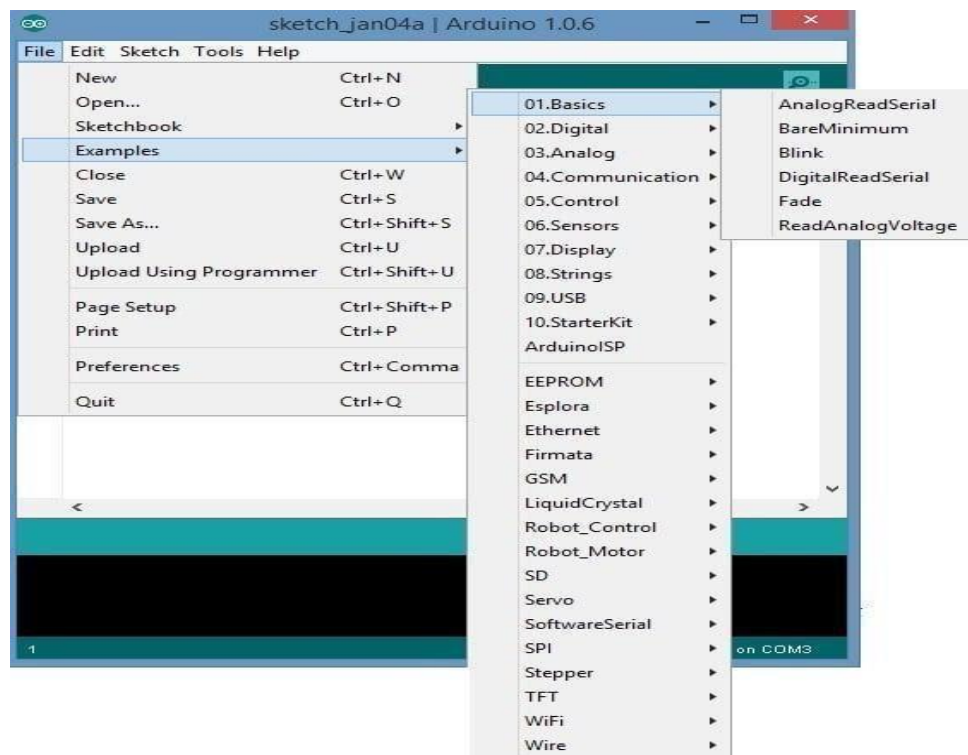
Selecting arduino board

Selecting arduino port



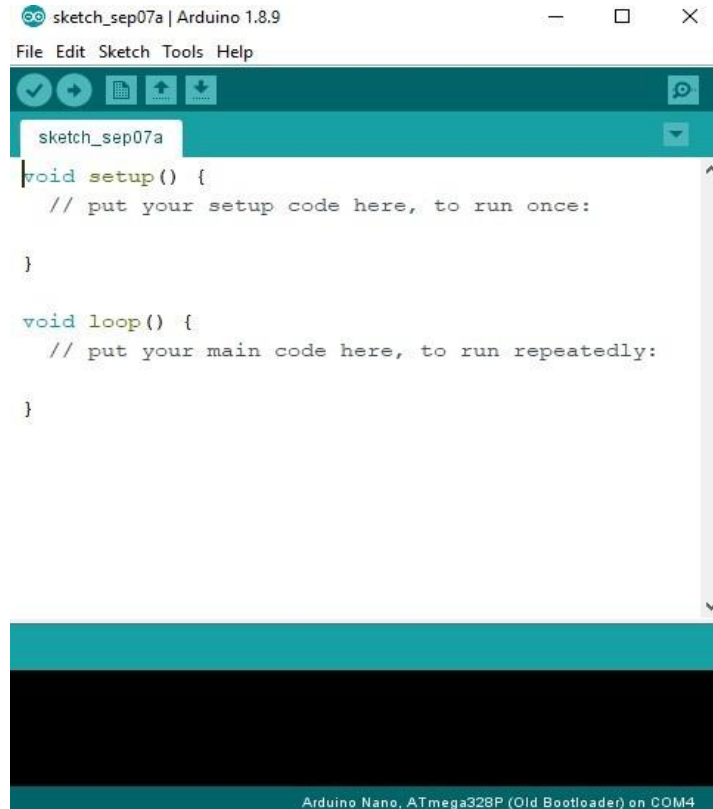
EXPLORING THE ARDUINO IDE

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 need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image.

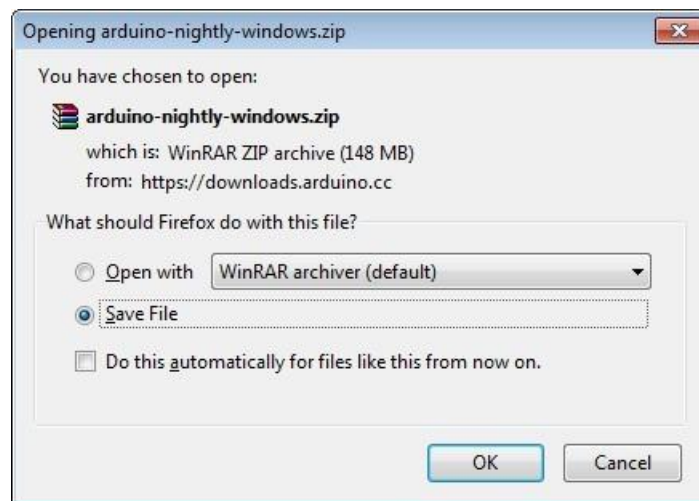


In case you use Arduino Nano, you will need an A to Mini-B cable instead as shown in the following image.



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.

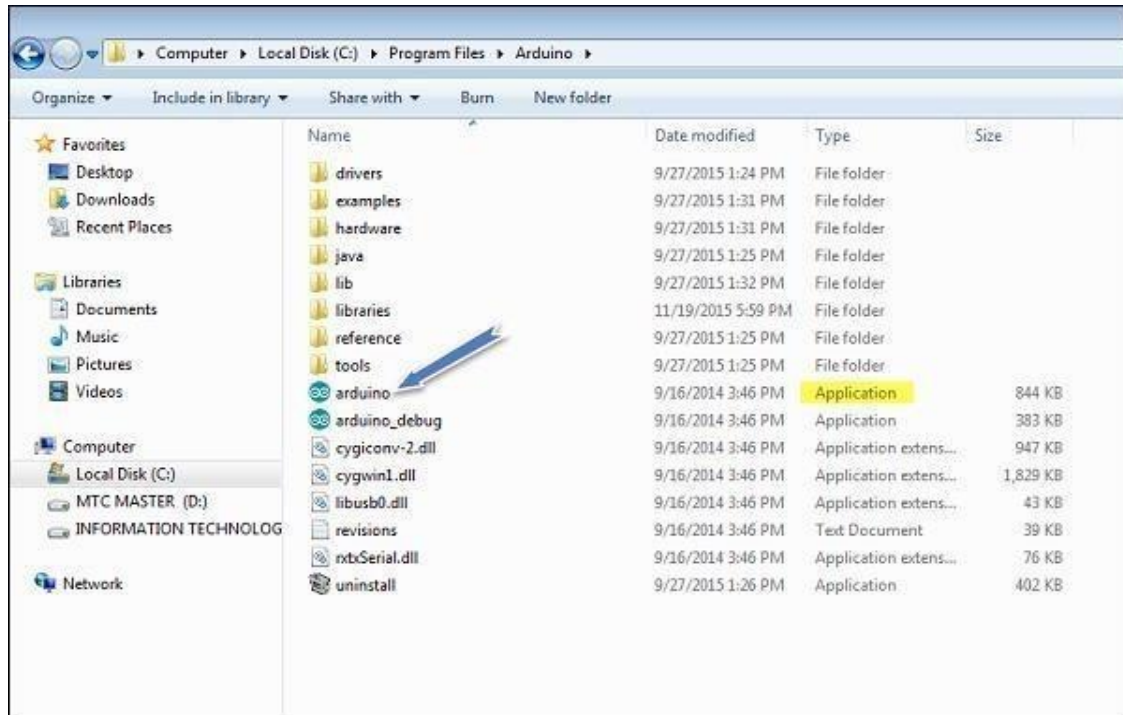
The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an Arduino Diecimila, you have to make sure that the board is configured to draw power from the USB connection. The power source

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.

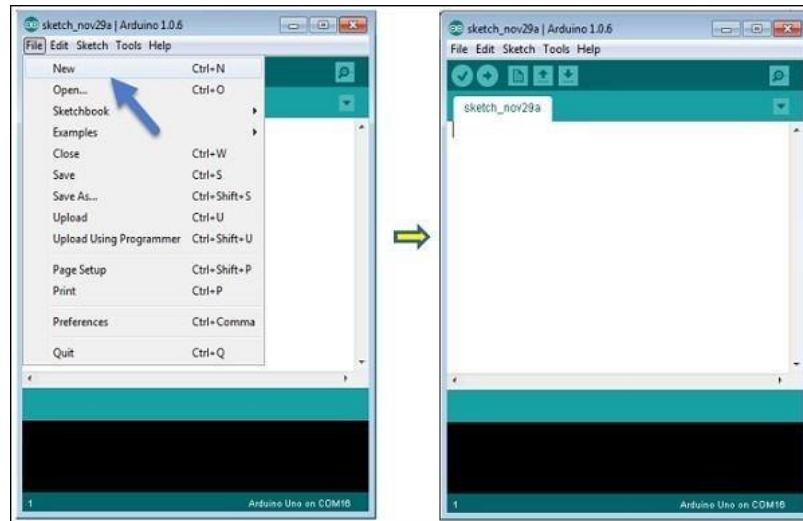


Step 5 – Open your first project.

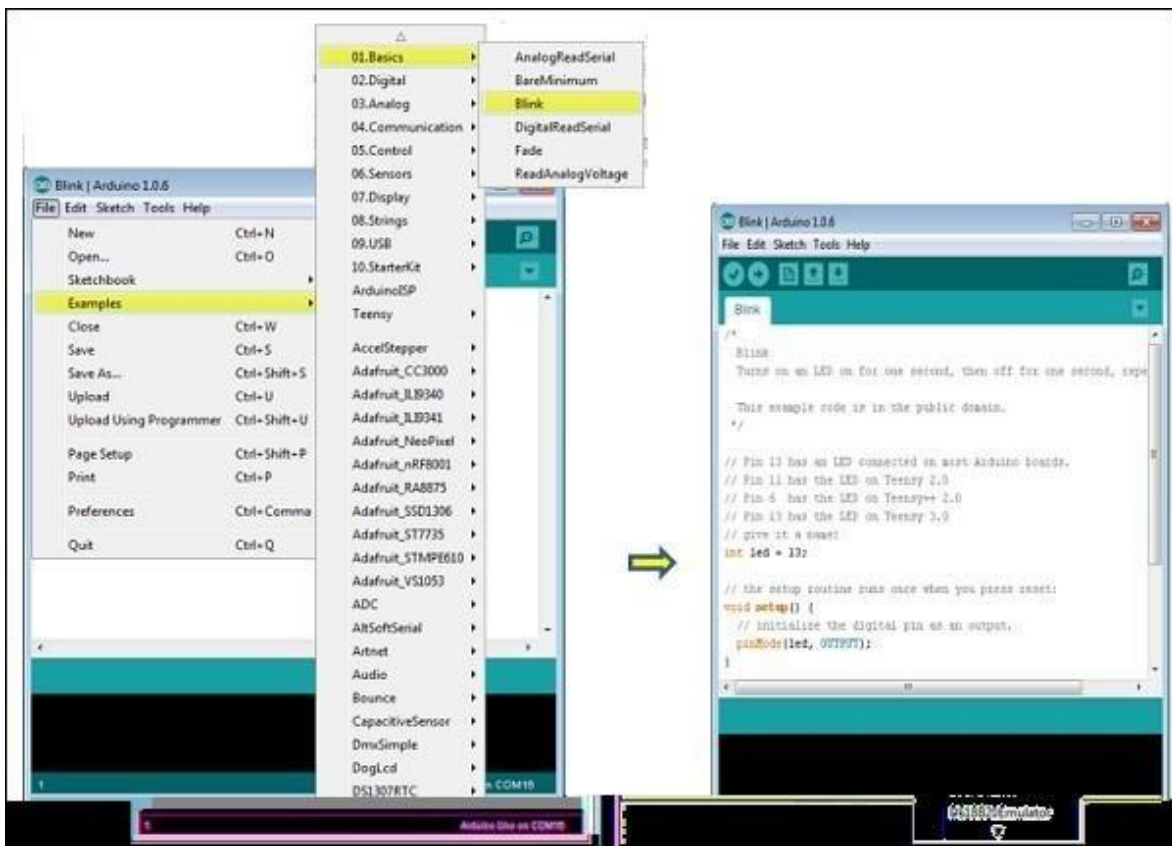
Once the software starts, you have two options –

- 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.

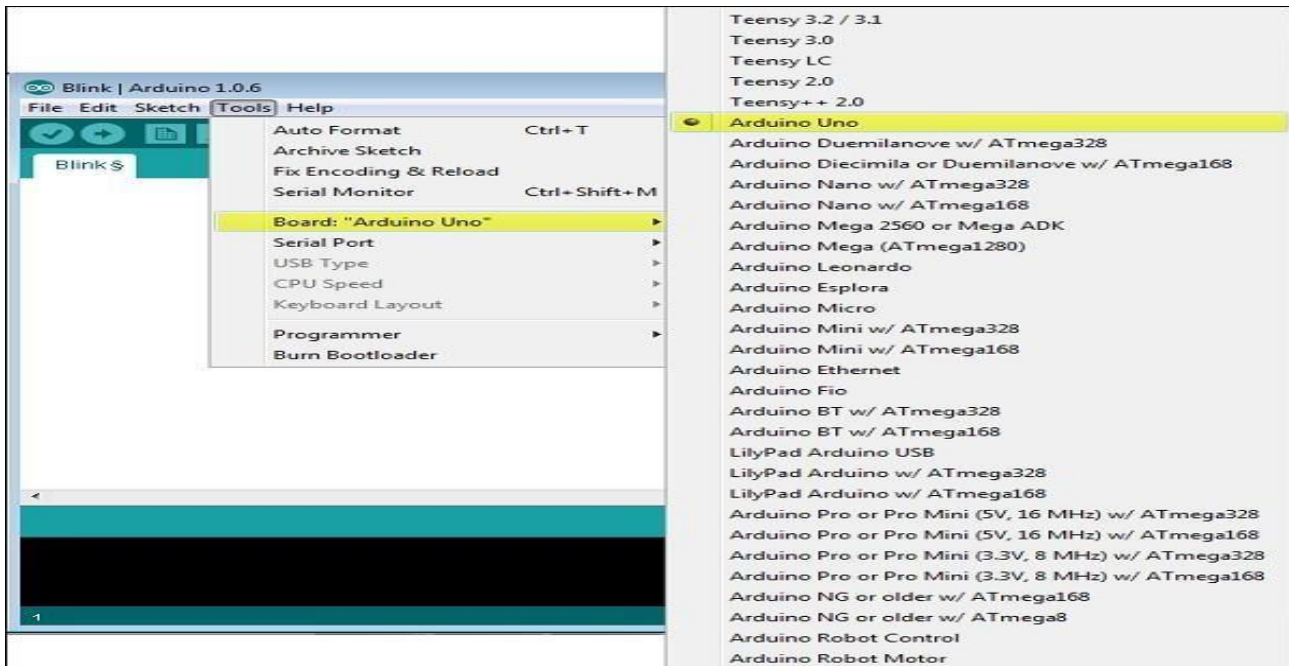


Here, we are selecting just one of the examples with the name Blink. It turns the LED on and off with some time delay. You can select any other example from the list.

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.

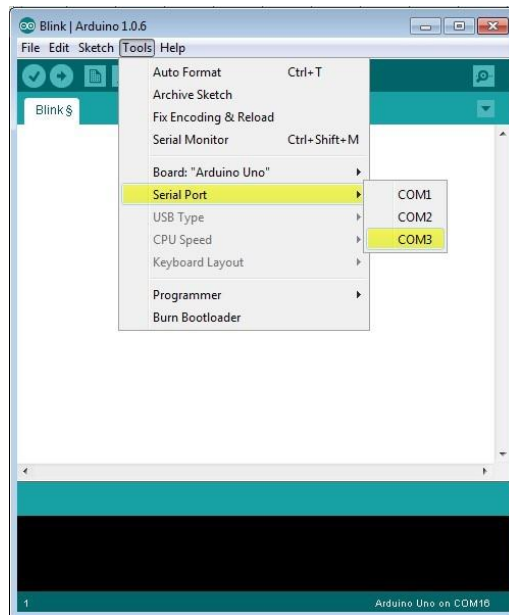
Go to Tools → Board and select your board.



Here, we have selected Arduino Uno board according to our tutorial, but you must select the name matching the 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.

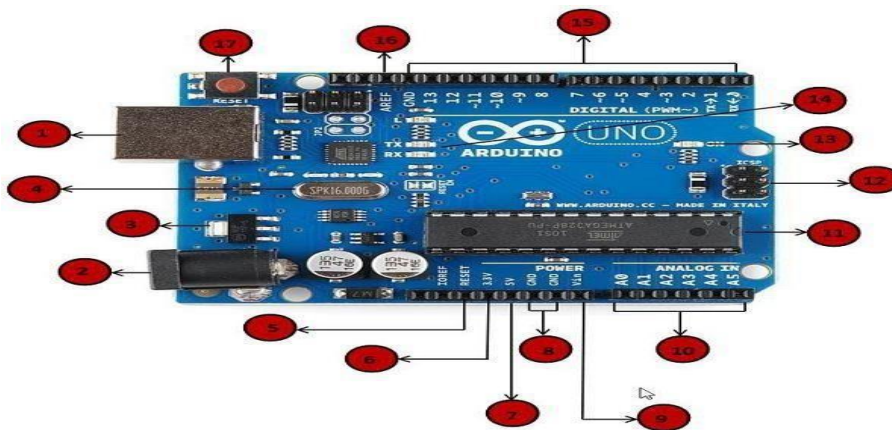
D – Used to directly open one of the example sketch.

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 TX LEDs 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. It tells 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 board in 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 used to 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 an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that

can be read by the microprocessor.

11. 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 are usually of the ATMEGA 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.

12. 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.

13. 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.

14. 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 the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.

15. 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.

16. 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 reset of 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

Date:

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 Raspberry pi.

COMPONENTS REQUIRED:

1. Raspberry pi
2. Breadboard
3. Jumper wires
4. Resistor
5. LED

ALGORITHM:

- Start the process.
- Connect micro USB power input to Raspberry pi
- Connect HDMI to the system to act as monitor for Raspberry pi.
- Connect USB port 2.0 to mouse and keyboard.
- Enter the coding in the terminal for installing python and GPTO.
- Open notepad → enter coding → save as → file extension python or py.
- Copy 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 is 0 in breadboard using Raspberry pi.
- Stop the process.

CODING:

```
sudo apt-get install python-pip
sudo apt-get install python-dev
sudo pip install RPi.GPIO
sudo -i
#python
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(18,GPIO.OUT)
ip=int(input("enter the value: "))
```

```
ifip==1:
print "LED on"
GPIO.output(18,GPIO.HIGH)
time.sleep(1)
elifip==0:
print "LED off"
GPIO.output(18,GPIO.LOW)
time.sleep(1)
```

OUTPUT:

```
pi@raspberrypi:~ $ cd Desktop/
pi@raspberrypi:~/Desktop $ sudo apt-get install python-rpi.gpio
Reading package lists... Done
Building dependency tree
Reading state information... Done
python-rpi.gpio is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
pi@raspberrypi:~/Desktop $ nano ledexample.py
pi@raspberrypi:~/Desktop $ python ledexample.py
```



RESULT:

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

EXP : 3

Date:

Identify the objects using IR and PIR sensor

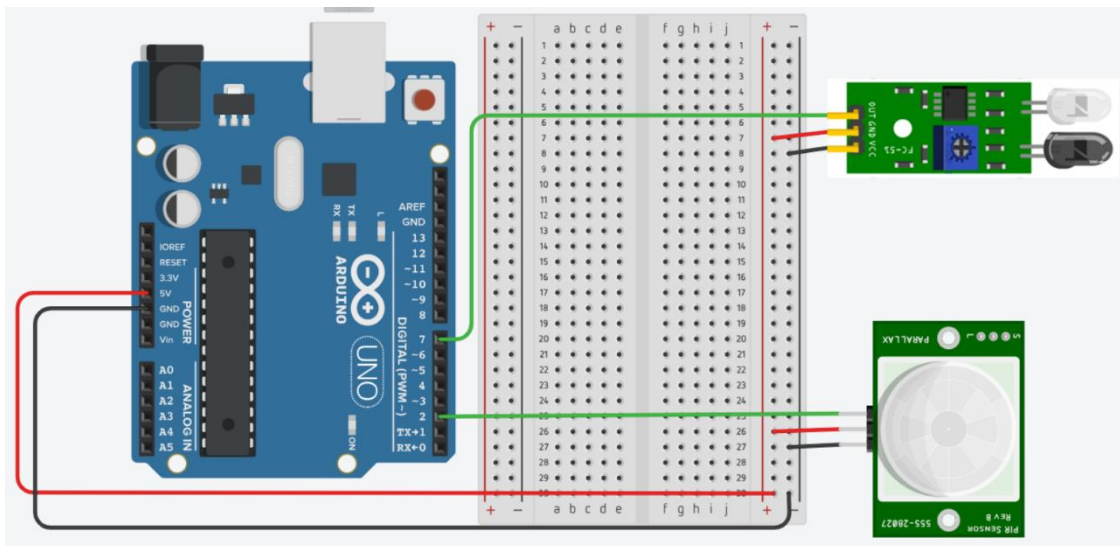
AIM:

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

COMPONENTS REQUIRED:

- Arduino Uno
- IR transmitter
- IR Receiver
- PIR sensor
- 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.

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:

```
const int pirSensorPin = 2; // PIR sensor output pin
const int irSensorPin = 7; // IR sensor output pin
int pirState = LOW;      // we start, assuming no motion detected

void setup() {
  pinMode(irSensorPin, INPUT);
  pinMode(pirSensorPin, INPUT);
  Serial.begin(9600);
}

void loop() {
  int pirSensorValue = digitalRead(pirSensorPin); // read input value
  int irSensorValue = digitalRead(irSensorPin);

  if(irSensorValue == 1){
    Serial.println("Object detected by IR sensor!");
  }


  if (pirSensorValue == HIGH) {      // check if the input is HIGH
    if (pirState == LOW) {
      Serial.println("Motion detected!");
      // We only want to print on the output change, not state
      pirState = HIGH;
    }
  } else {
    if (pirState == HIGH){
      Serial.println("Motion ended!");
      // We only want to print on the output change, not state
      pirState = LOW;
    }
  }

  delay(1000); // Delay for 1 second before checking sensors again
}
```

Output:

Serial Monitor

Object detected by IR sensor!
Object detected by IR sensor!
Object detected by IR sensor!
Object detected by IR sensor!
Object detected by IR sensor!
Object detected by IR sensor!
Motion detected!
Motion ended!
Motion detected!
Motion ended!
Object detected by IR sensor!
Motion detected!
Motion ended!

SendClear

Result:

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

EXP: 4

Date: **Measure the moisture level of soil using a soil moisture sensor**

AIM:

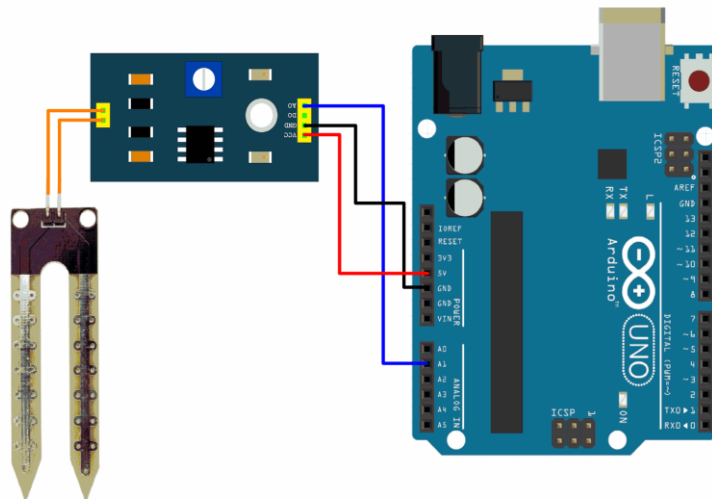
The aim of this experiment is to develop a system that can measure the moisture level of soil using a soil moisture sensor and Arduino Uno. The system controls the sensor and displays the moisture level.

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:

```
const int soilMoisturePin = A1; // Analog pin A1 for soil moisture sensor
```

```
void setup() {  
  pinMode(soilMoisturePin, INPUT);  
  Serial.begin(9600); //Set baud rate for serial communication  
}
```

```
void loop() {  
  int soilMoistureValue = analogRead(soilMoisturePin);  
  
  Serial.print("Soil Moisture: ");  
  Serial.print(soilMoistureValue);  
  Serial.println("");  
  if(soilMoistureValue > 1000){  
    Serial.println("Sensor is not in the soil or Sensor not connected\n");  
  }  
  else if(soilMoistureValue >= 600 && soilMoistureValue <= 1000) {  
    Serial.println("Dry");  
  }  
  else if(soilMoistureValue >= 370 && soilMoistureValue < 600) {  
    Serial.println("Humid");  
  }  
  else{  
    Serial.println("Wet");  
  }  
  delay(1000); // Delay for 1 second before reading the sensor again  
}
```

Output:

Serial Monitor

dry

Soil Moisture: 949

Dry

Soil Moisture: 768

Dry

Soil Moisture: 398

Humid

Soil Moisture: 41

Wet

Soil Moisture: 183

Wet

Soil Moisture: 323

Wet

Soil Moisture: 916

Dry

Soil Moisture: 386

Humid

Soil Moisture: 171

Wet

Soil Moisture: 64

Wet

Soil Moisture: 662

Dry

Send

Clear

Result:

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

EXP : 5

Date: **Measure the distance between an ultrasonic sensor and an obstacle**

AIM:

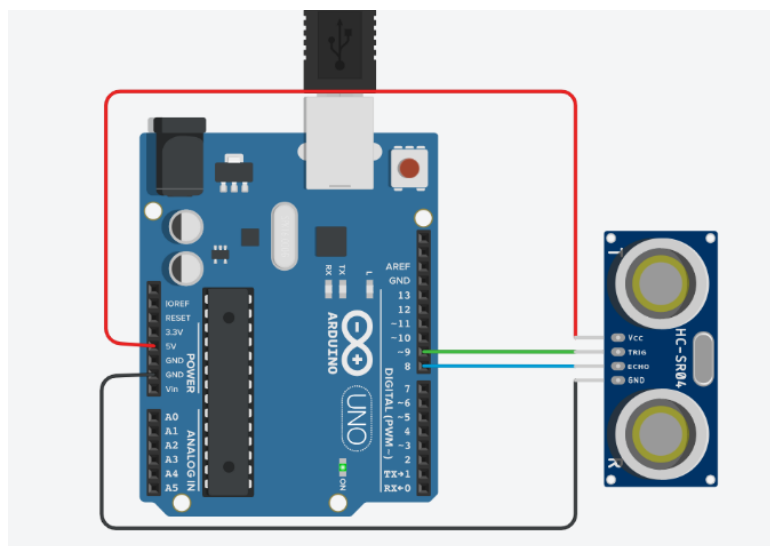
The aim of this experiment is to develop a system that can measure the distance between an ultrasonic sensor and an obstacle using Arduino Uno. The system control the sensor and to display the distance.

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 travel to the obstacle and back.
4. The distance to the obstacle can be calculated using the following formula:
5. $\text{distance} = (\text{speed of sound} * \text{time}) / 2$
6. Connect the ultrasonic sensor to the Arduino.
7. Write the program.
8. Run the program.

Circuit Diagram:



Program:

```
int trigPin = 9;  // TRIG pin
int echoPin = 8;  // ECHO pin
// define variables
long duration;
int distance;

void setup() {
  // Sets the trigPin as an Output
  pinMode(trigPin, OUTPUT);
  // Sets the echoPin as an Input
  pinMode(echoPin, INPUT);
  // Start Serial Communication
  Serial.begin (9600);
}

void loop() {
  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10-micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // measure duration of pulse from ECHO pin
  duration = pulseIn(echoPin, HIGH);
  // calculate the distance
  distance = duration * 0.034/2;
  // print the value to Serial Monitor
  Serial.print("distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  delay(500);
}
```

Output:

Serial Monitor

distance: 97 cm
distance: 97 cm
distance: 97 cm
distance: 97 cm
distance: 97 cm
distance: 97 cm
distance: 97 cm
distance: 97 cm
distance: 87 cm
distance: 26 cm
distance: 20 cm
distance: 189 cm
distance: 307 cm
distance: 332 cm
distance: 332 cm
distance: 332 cm
distance: 261 cm
distance: 183 cm
distance: 152 cm
distance: 122 cm
distance: 99 cm
distance: 99 cm
distance: 99 cm
distance: 99 cm

Send

Clear

Result

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

EXP : 6

Date: Identify the leakage of gas/smoke in the environment

AIM:

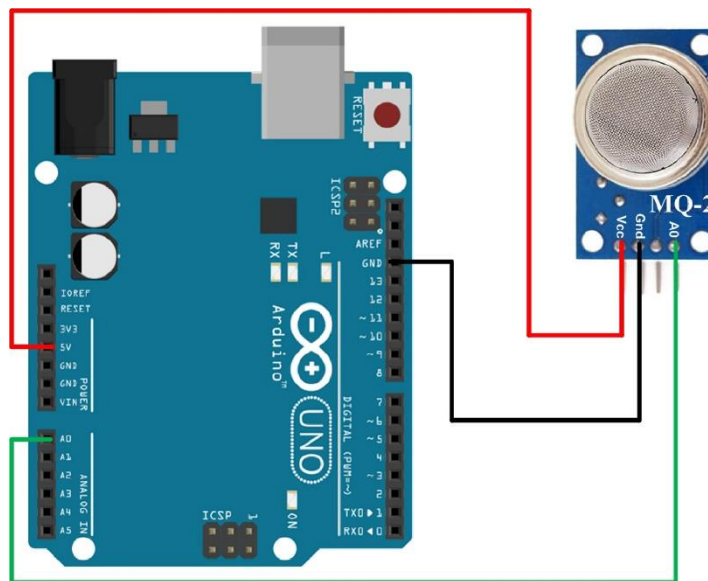
The aim of this experiment is to develop a system that can identify the leakage of gas/smoke in the environment using Arduino Uno. The system controls the sensor and displays an alert if gas/smoke is detected.

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:

```
#define Threshold 400
#define MQ2pin 0

float sensorValue;

void setup() {
  Serial.begin(9600); // Sets the serial port to 9600
  Serial.println("MQ2 warming up!");
  delay(20000); // allow the MQ2 to warm up
}

void loop() {
  sensorValue = analogRead(MQ2pin); // read analog input pin 0
  Serial.print("Sensor Value: ");
  Serial.print(sensorValue);

  if(sensorValue > Threshold)
  {
    Serial.print(" | Smoke detected!");
  }
  Serial.println("");
  delay(2000); // wait 2s for next reading
}
```

Output:

Serial Monitor

MQ2 warming up!

Sensor Value: 209.00

Sensor Value: 255.00

Sensor Value: 854.00 | Smoke detected!

Sensor Value: 63.00

Sensor Value: 792.00 | Smoke detected!

Sensor Value: 485.00 | Smoke detected!

Sensor Value: 910.00 | Smoke detected!

Sensor Value: 229.00

Sensor Value: 903.00 | Smoke detected!

Sensor Value: 572.00 | Smoke detected!

Sensor Value: 676.00 | Smoke detected!

Sensor Value: 755.00 | Smoke detected!

Sensor Value: 359.00

Sensor Value: 573.00 | Smoke detected!

Sensor Value: 140.00

Sensor Value: 691.00 | Smoke detected!

Sensor Value: 727.00 | Smoke detected!

Sensor Value: 413.00 | Smoke detected!

Sensor Value: 1006.00 | Smoke detected!

Sensor Value: 976.00 | Smoke detected!

Sensor Value: 60.00

Send

Clear

Result:

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

EXP: 7

Date: **Measure the humidity and moisture value of the environment**

AIM:

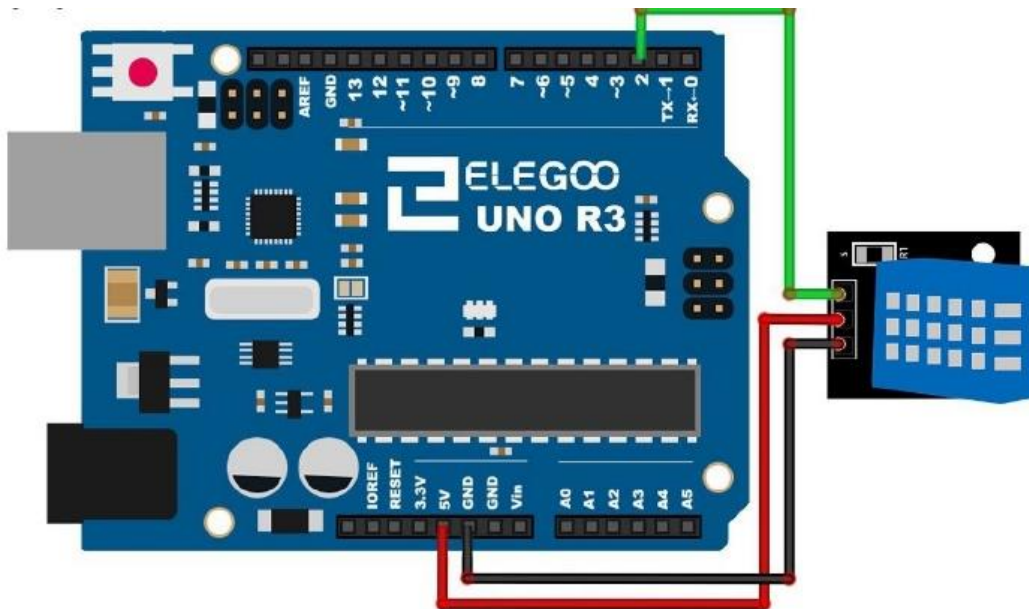
The aim of this experiment is to develop a system that can measure the humidity and moisture value of the environment using Arduino Uno. The system control the sensor and to display the humidity and moisture values.

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"

#define DHTPIN 2    // what pin we're connected to

#define DHTTYPE DHT11  // DHT 11

// Initialize DHT sensor for normal 16mhz Arduino
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  Serial.println("DHTxx test!");

  dht.begin();
}

void loop() {
  // Wait a few seconds between measurements.
  delay(2000);

  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  float h = dht.readHumidity();
  // Read temperature as Celsius
  float t = dht.readTemperature();
  // Read temperature as Fahrenheit
  float f = dht.readTemperature(true);

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
}
```

```
}

// Compute heat index
// Must send in temp in Fahrenheit!
float hi = dht.computeHeatIndex(f, h);

Serial.print("Humidity: ");
Serial.print(h);
Serial.print(" %\t");
Serial.print("Temperature: ");
Serial.print(t);
Serial.print(" *C ");
Serial.print(f);
Serial.print(" *F\t");
Serial.print("Heat index: ");
Serial.print(hi);
Serial.println(" *F");
}
```

Output:

Serial Monitor

DHTxx test!
Humidity: 75.00%
Humidity: 75.00%
Humidity: 75.00%
Humidity: 75.00%
Humidity: 76.00%
Humidity: 76.00%
Humidity: 76.00%
Humidity: 77.00%
Humidity: 75.00%
Humidity: 77.00%
Humidity: 75.00%
Humidity: 78.00%

Temperature: 18.00 *C 64.40 *F
Temperature: 18.00 *C 64.43 *F
Temperature: 19.00 *C 64.40 *F
Temperature: 19.00 *C 65.44 *F
Temperature: 18.00 *C 63.24 *F
Temperature: 17.00 *C 66.46 *F
Temperature: 18.00 *C 66.40 *F
Temperature: 20.00 *C 67.43 *F
Temperature: 18.00 *C 68.40 *F
Temperature: 19.00 *C 68.40 *F
Temperature: 18.00 *C 67.54 *F
Temperature: 19.00 *C 68.40 *F

Heat index: 72.76 *F
Heat index: 72.76 *F
Heat index: 72.76 *F
Heat index: 72.73 *F
Heat index: 72.76 *F
Heat index: 71.64 *F
Heat index: 72.74 *F
Heat index: 71.76 *F
Heat index: 71.76 *F
Heat index: 71.34 *F
Heat index: 72.76 *F
Heat index: 72.76 *F

Send

Clear

Result:

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

EXP : 8

Date: Control a LED using a relay switch

AIM:

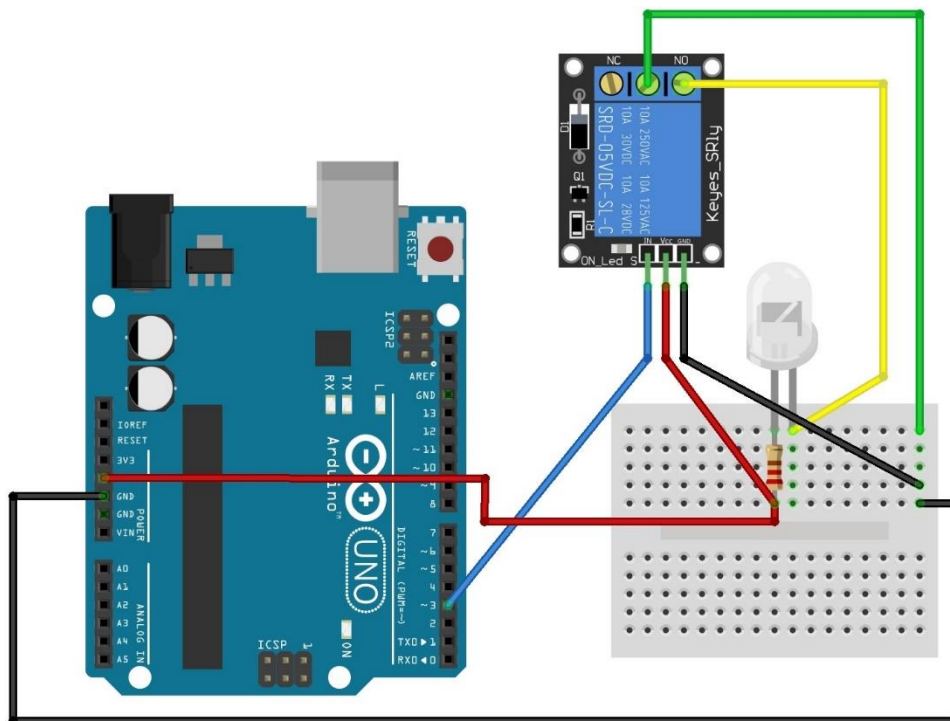
The aim of this experiment is to develop a system that can control a LED using a relay switch using Arduino Uno. The system controls the relay switch and to turn the LED on and off.

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:

```
int Relaypin= 3; // Define input pin for relay
void setup() {
    // put your setup code here, to run once:
    pinMode(Relaypin, OUTPUT); // Define the Relaypin as output pin
}
void loop() {
    // put your main code here, to run repeatedly:
    digitalWrite(Relaypin, HIGH); // Sends high signal
    delay(1000); // Waits for 1 second
    digitalWrite(Relaypin, LOW); // Makes the signal low
    delay(1000); // Waits for 1 second
}
```

Output:

Serial Monitor

LED on

LED On

LED off

LED On

LED off

LED On

LED off

LED On

LED off

LED On

LED off

LED On

LED off

LED On

LED off

LED On

LED off

LED On

LED off

LED On

LED off

LED On

LED off

LED On

LED off

Send

Clear

Result:

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

Mini Project

EXP : 9

Date:

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.

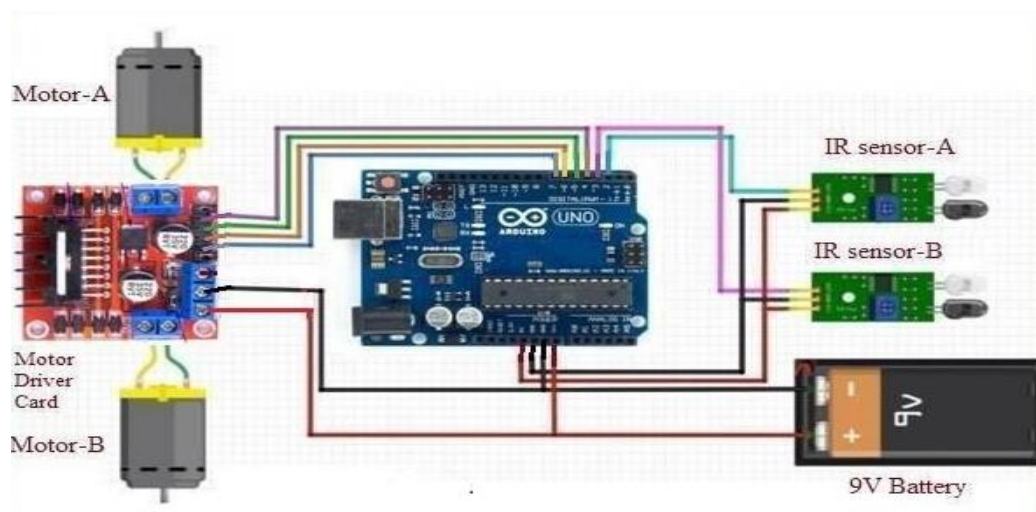
STEP 2: In this case, both the sensors do not detect a black line, so the robot should keep moving forward in the same direction and maintain its current journey path.

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:



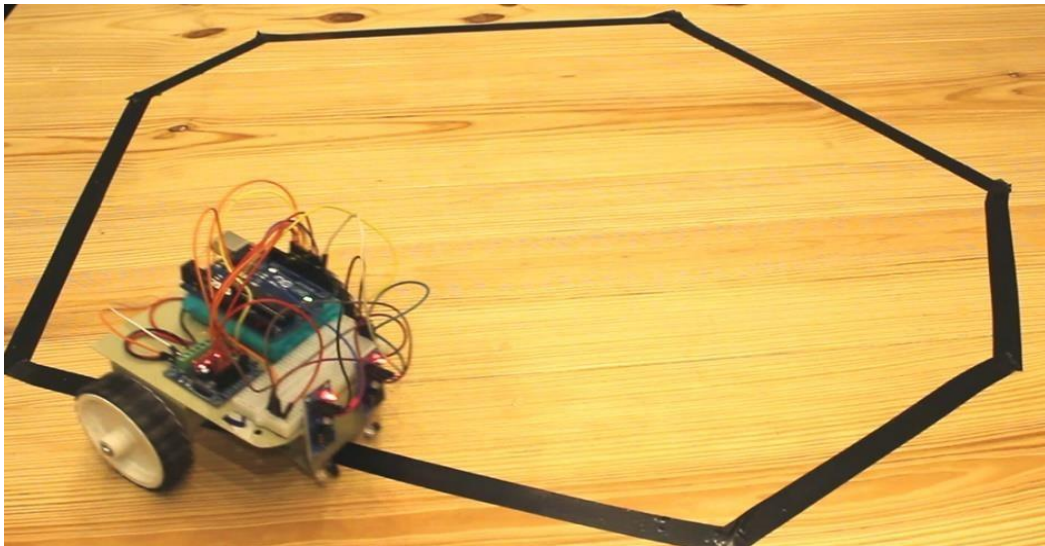
Program:

The Python program for the experiment is as follows:

```
/*----- Arduino Line Follower Code----- */  
  
/*-----defining Inputs-----*/  
  
#define LS 2    // left sensor  
#define RS 3    // right sensor  
  
/*-----defining Outputs-----*/  
  
#define LM1 4    // left motor  
#define LM2 5    // left motor  
#define RM1 6    // right motor  
#define RM2 7    // right motor  
  
void setup()  
{  
    pinMode(LS, INPUT);  
    pinMode(RS, INPUT);  
    pinMode(LM1, OUTPUT);  
    pinMode(LM2, OUTPUT);  
    pinMode(RM1, OUTPUT);  
    pinMode(RM2, OUTPUT);  
}  
  
void loop()  
{  
    if(digitalRead(LS) && digitalRead(RS))    // Move Forward  
    {  
        digitalWrite(LM1, HIGH);  
        digitalWrite(LM2, LOW);  
        digitalWrite(RM1, HIGH);  
        digitalWrite(RM2, LOW);  
    }  
}
```

```
if(!(digitalRead(LS)) && digitalRead(RS))    // Turn right
{
    digitalWrite(LM1, LOW);
    digitalWrite(LM2, LOW);
    digitalWrite(RM1, HIGH);
    digitalWrite(RM2, LOW);
}
if(digitalRead(LS) && !(digitalRead(RS)))    // turn left
{
    digitalWrite(LM1, HIGH);
    digitalWrite(LM2, LOW);
    digitalWrite(RM1, LOW);
    digitalWrite(RM2, LOW);
}
if(!(digitalRead(LS)) && !(digitalRead(RS)))    // stop
{
    digitalWrite(LM1, LOW);
    digitalWrite(LM2, LOW);
    digitalWrite(RM1, LOW);
    digitalWrite(RM2, 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.

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:

```
import colorama
from termcolor import colored
options_message = """
Choose:
1. To park a vehicle
2. To remove a vehicle from parking
3. Show parking layout
4. Exit
"""
class Vehicle:
    def __init__(self, v_type, v_number):
        self.v_type = v_type
        self.v_number = v_number
        self.vehicle_types = {1: 'c', 2: 'b', 3: 't'}
    def __str__(self):
        return self.vehicle_types[self.v_type]
class Slot:
    def __init__(self):
        self.vehicle = None
    @property
    def is_empty(self):
        return self.vehicle is None
class Parking:
    def __init__(self, rows, columns):
        self.rows = rows
        self.columns = columns
        self.slots = self._get_slots(rows, columns)
    def start(self)
    while True:
        try:
            print(options_message)

            option = input("Enter your choice: ")
            if option == '1':
                self._park_vehicle()
            if option == '2':
                self._remove_vehicle()
            if option == '3':
                self.show_layout()
            if option == '4':
```



```

        break
except ValueError as e:
    print(colored(f"An error occurred: {e}. Try again.", "red"))

    print(colored("Thanks for using our parking assistance system", "green"))
def _park_vehicle(self):
    vehicle_type = self._get_safe_int("Available vehicle types: 1. Car\t2. Bike\t3. Truck.\nEnter your choice: ")
    if vehicle_type not in [1, 2, 3]:
        raise ValueError("Invalid vehicle type specified")
        vehicle_number = input("Enter vehicle name plate: ")
    if not vehicle_number:
        raise ValueError("Vehicle name plate cannot be empty.")
        vehicle = Vehicle(vehicle_type, vehicle_number)
        col = self._get_safe_int("Enter the column where you want to park the vehicle: ")
        if col <= 0 or col > self.columns:
            raise ValueError("Invalid row or column number specified")
            col = self._get_safe_int("Enter the column where you want to park the vehicle: ")
        if col <= 0 or col > self.columns:
            raise ValueError("Invalid row or column number specified")
            row = self._get_safe_int("Enter the row where you want to park the vehicle: ")
        if row <= 0 or row > self.rows:
            raise ValueError("Invalid row number specified")
            slot = self.slots[row-1][col-1]
        if not slot.is_empty:
            raise ValueError("Slot is not empty. Please choose an empty slot.")
            slot.vehicle = vehicle

def _remove_vehicle(self):
    vehicle_number = input("Enter the vehicle number that needs to be removed from parking slot: ")
    if not vehicle_number:
        raise ValueError("Vehicle number is required.")

    for row in self.slots:
        for slot in row:
            if slot.vehicle and slot.vehicle.v_number.lower() == vehicle_number.lower():
                vehicle: Vehicle = slot.vehicle
                slot.vehicle = None
                print(colored(f"Vehicle with number '{vehicle.v_number}' removed from parking",
"green"))

```

```

        return

    else:

        raise ValueError("Vehicle not found.")

def show_layout(self):

    col_info = [f'<{col}>' for col in range(1, self.columns + 1)]
    print(colored(f'|{" ".join(col_info)}|columns', "yellow"))
    self._print_border(text="rows")

    for i, row in enumerate(self.slots, 1):

        string_to_printed = "|"
        for j, col in enumerate(row, 1):
            string_to_printed += colored(f"[{col.vehicle if col.vehicle else ' ' }]", "red" if col.vehicle else
"green")

            string_to_printed += colored(f"|<{i}>", "cyan")
        print(string_to_printed)
        self._print_border()

def _print_border(self, text=""):

    print(colored(f'|{'-' * self.columns * 3}|{colored(text, 'cyan')}}', "blue"))

def _get_slot_count(self):

    count = 0

    for row in self.slots:

        for slot in row:

            if slot.is_empty:

                count += 1

        return count

    @staticmethod

def _get_slots(rows, columns):

    slots = []

    for row in range(0, rows):

        col_slot = []

        for col in range(0, columns):

            col_slot.append(Slot())

```

```

        slots.append(col_slot)
    return slots

    @staticmethod
    def _get_safe_int(message):
        try:
            val = int(input(message))

            return val

        except ValueError:

            raise ValueError("Value should be an integer only")

def main():
    try:
        print(colored("Welcome to the parking assistance system.", "green"))
        print(colored("First let's setup the parking system", "yellow"))
        rows = int(input("Enter the number of rows: "))
        columns = int(input("Enter the number of columns: "))
        print("Initializing parking")

        parking = Parking(rows, columns)
        parking.start()

    except ValueError:

        print("Rows and columns should be integers only.")
    except Exception as e:
        print(colored(f"An error occurred: {e}", "red"))

if __name__ == '__main__':
    colorama.init() # To enable color visible in command prompt

    main()

```

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 1 Event
vra-built charat infover Before the infover time and FBI load with vra-built Duthon narbanas charat infover // Please download // Download on 126 minutes ago 35.10 CBE 1ITE.B A snare Duthon 3
```

Result:

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

EXP : 11**Date:****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.

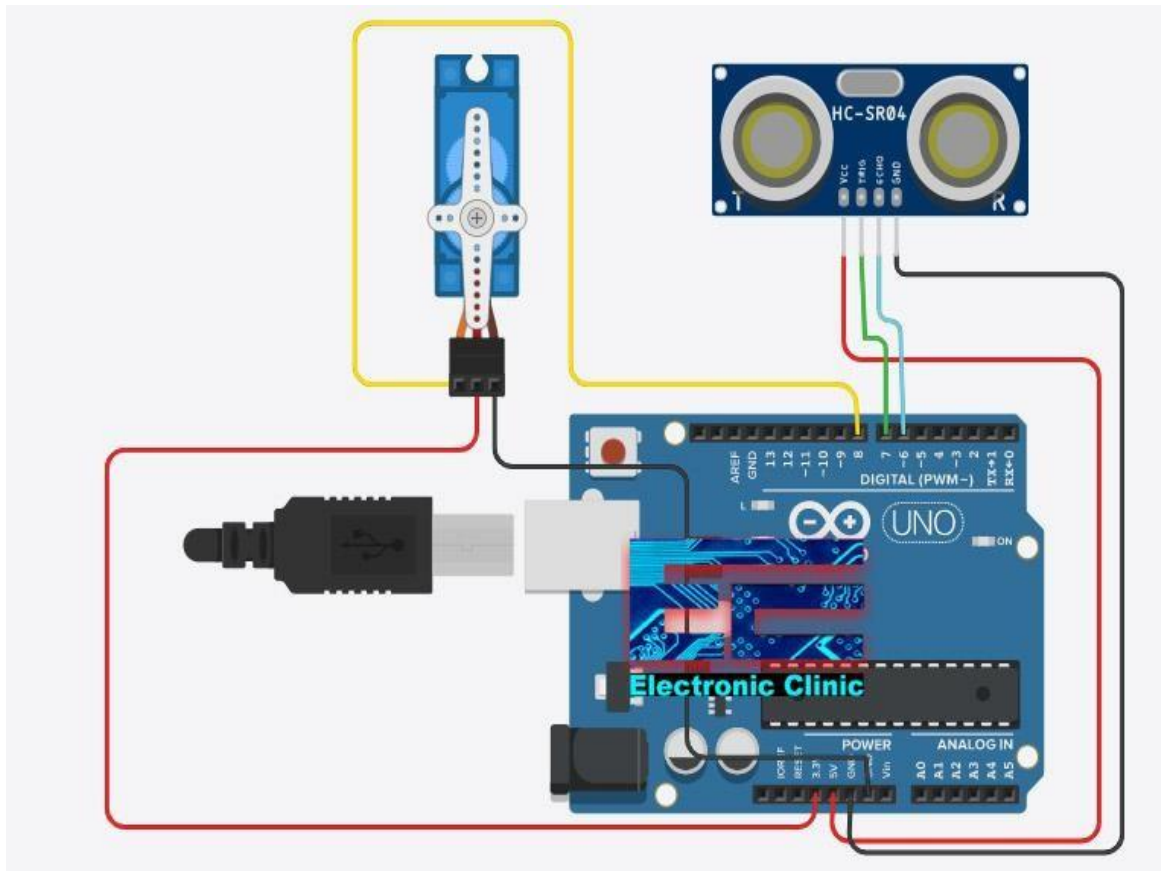
Hardware Requirements:

- 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:

The Python program for the experiment is as follows:

```
Import random
Class SmartWasteBin:
Def_init_(self,bin_id,max_capacity):
Self.bin_id=bin_id
Self.max_capacity=max_capacity
Self.current_level=0
Def add_waste(self,amount):
Self.current_level += amount
```

```
If self.current_level >= Self.max_capacity:
    Self.alert_full()
Def alert_full(self):
    Print(f"Bin {self.bin_id} is full.Needs emptying!")
Def main():
    Bins = [SmartWasteBin(bin_id,random.randint(50,200))
    for bin_id in range(1,6)]
    For_in range(20):
        For bin in bins:
            Waste_added =random.randint(0,30)
            Bin.add_waste(waste_added)
    If _name_=="_main_":
        Main()
```


Output:



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 : 12

Date: **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>
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <DHT.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);
DHT dht(D3, DHT11); //(sensor pin,sensor type)
BlynkTimer timer;

char auth[] = ""; //Enter the Auth code which was send by Blink
char ssid[] = ""; //Enter your WIFI Name
char pass[] = ""; //Enter your WIFI Password

void weather()
{
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  int r = analogRead(A0);
  bool l = digitalRead(D4);
  r = map(r, 0, 1023, 100, 0);
  if (isnan(h) || isnan(t))
  {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  Blynk.virtualWrite(V0, t); //V0 is for Temperature
  Blynk.virtualWrite(V1, h); //V1 is for Humidity  Blynk.virtualWrite(V2, r); //V2 is for Rainfall
  if (l == 0) {
    WidgetLED led1(V3);
    led1.on();
    lcd.setCursor(9, 1);
    lcd.print("L :");
    lcd.print("High");
    lcd.print(" ");
  }
}
```

```
else if (l == 1)
{
    WidgetLED led1(V3);
    led1.off();
    lcd.setCursor(9, 1);
    lcd.print("L :");
    lcd.print("Low");
    lcd.print(" ");
}
lcd.setCursor(0, 0);
lcd.print("T :");
lcd.print(t);
lcd.setCursor(0, 1);
lcd.print("H :");
lcd.print(h);
lcd.setCursor(9, 0);
lcd.print("R :");
lcd.print(r);
lcd.print(" ");
}

void setup() {
    Serial.begin(9600); // See the connection status in Serial Monitor
    lcd.init();
    lcd.backlight();
    Blynk.begin(auth, ssid, pass);
    dht.begin();
    // Setup a function to be called every second
    timer.setInterval(10L, weather);
}

void loop() {
    Blynk.run(); // Initiates Blynk
    timer.run(); // Initiates SimpleTimer
}
```

Output:



Result:

Smart weather monitoring system using Arduino was implemented successfully

