

AD22511-INTERNET OF THINGS AND ITS APPLICATIONS LABORATORY

EX.NO:1

DATE:

Install the IDE of the Arduino and study the programming in the development environment

STUDY OF ARDUINO IDE

The Arduino IDE is a software application where the Arduino programs are written in C or C++ and uploaded to the Arduino board

Installation:

It is an open-source software which can be downloaded for free from <https://www.arduino.cc/en/software>. Make sure to download the stable version and not the hourly builds or previous releases.

Configuration:

After successful installation, set the type to the model of Arduino board because this tells the IDE which microcontroller the Arduino board has so that it can compile the program accordingly

Head to the “Tools” menu in the navigator bar, choose the “Board” option and set the type of the Arduino board. Now the IDE knows about the board, now configure the IDE which USB port to use for sending data i.e. the compiled code to the Arduino board

Make sure to connect the Arduino board to the computer using a USB cable. After connecting, head to the “Tools” menu, choose the “Port” option and set the port to the USB port to the port which is connected the Arduino board

Writing programs:

Every program requires the following structure as they are the essential part of the program

```
void setup {  
}  
void loop {  
}
```

The “setup” function is executed only once and doesn’t change during

theruntime.Code suchas setting the pins for input and output goes here. Always remember to place this functionabove the “loop” function.

The “loop” function is executed post the execution of the “setup” function and is code inside this function is executed infinite no. of times. Most of the code that makes up the programs goes inside this function. The IDE provides many inbuilt functions that can be used here.

Uploading programs:

To send the programs to the Arduino board, click the right arrow button which is located just below the menu bar and to the left of the tick mark button. The tick mark button will compile the code and points out the errors in the code. By default, the upload button i.e. the right arrow button compiles the code before sending it to the Arduino board.

BASIC IN-BUILT FUNCTIONS

Arduino IDE provides us with many utility functions that helps us interpret and control theArduino boards during the run times. Some of the basic function it provides are as follows:

pinMode(pin, mode):

This function is used to configure the pin specified to behave as Input or Output. pin – numberrepresenting the pin number in the board, mode – INPUT or OUTPUT

digitalRead(pin):

Function which is used to read the digital signal from the specified pin. Returns either HIGHor LOW. pin – number representing the pin number in the board

digitalWrite(pin, value):

This function is used to write a HIGH or a LOW value to a digital pin. pin – number representing the pin number in the board, value – HIGH or LOW

analogRead(pin):

This function is used to read the analog signal from the specified analog pin. Returns an integervalue in the range 0 to 1023. pin – number representing the pin number in the board

analogWrite(pin, value):

This function is used to write an integer value to the analog pin. The integer value should be within the range 0 to 255. pin - number representing the pin number in the board, value - 0 to 255

tone(pin, value, duration):

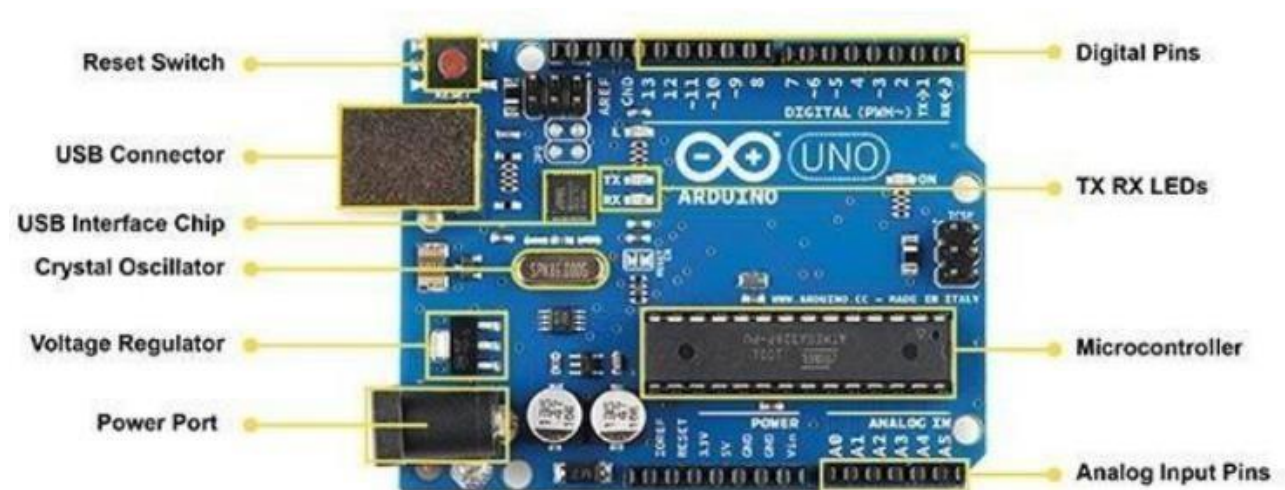
This function is used to generate a square wave for the specified “pin” with a frequency of “value” which last for “duration” milliseconds (defaults to infinity)

noTone(pin):

This function is used to stop the square wave being produced for the specified “pin”

delay(value):

Function that pauses the programs for the specified milliseconds i.e. value



Reset Switch - When this switch is clicked, it sends a logical pulse to the reset pin of the Microcontroller, and now runs the program again from the start

USB Connector - This is a printer USB port used to load a program from the Arduino IDE onto the Arduino board. The board can also be powered through this port

USB Interface Chip - It converts signals in the USB level to a level that an Arduino UNO board understands

Crystal Oscillator - This is a quartz crystal oscillator which ticks 16 million times a second. On each tick, the microcontroller performs one operation, for example, addition, subtraction, etc.

Voltage Regulator – Protects the board from burning out when more than 5 volts are given to the board and is located between the power port and USB connector

Power Port - The Arduino board can be powered through an AC-to-DC adapter or a battery. The power source can be connected by plugging in a 2.1mm center-positive plug into the powerjack of the board

Digital Pins - These pins can be used as either input or output pins. When used as output pins they act as power supply source and when used as input pins they read signals. Pins prepended with ~ (3, 5, 6, 9, 10, 11) are digital pins and can also act as analog pins

TX RX LEDs - TX stands for transmit, and RX for receive. These are indicator LEDs which blink whenever the UNO board is transmitting or receiving data

Microcontroller - The microcontroller used on the UNO board is Atmega328P which is pre-programmed with bootloader and has 32KB flash memory and 2KB RAM

Analog Input Pins – These pins read analog values and convert them to system understandable digital value. Due to high resistance they just measure voltage

AD22511-INTERNET OF THINGS AND ITS APPLICATIONS LABORATORY

EX.NO:2a

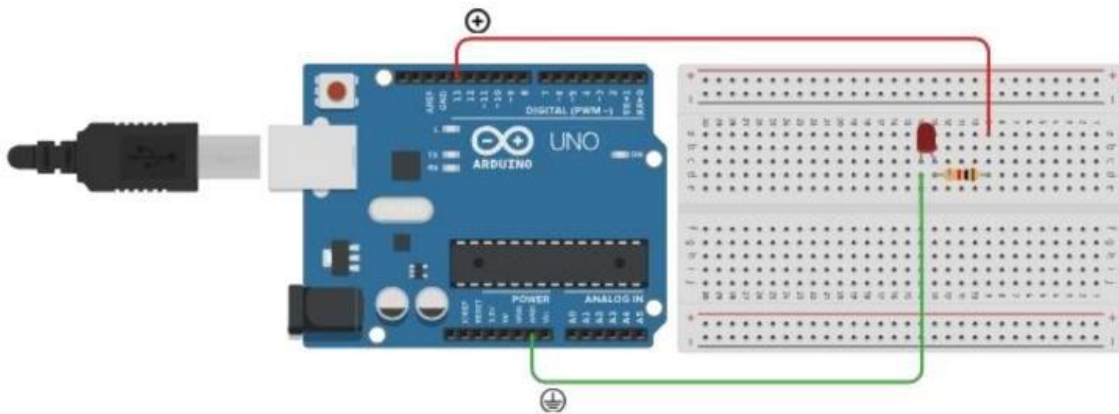
DATE:

Peripheral interfacing with IOT kit working with LED

AIM:

To understand the working of Arduino board and the IDE by working with the LED.

CIRCUIT DIAGRAM:



EQUIPMENT REQUIRED:

Arduino Uno R3, 1 k Ω Resistor, LED (any colour), bread board and jumper wires

DESCRIPTION:

Insert one side of the jumper wire into GND on the board. Lead the other side to the breadboard. Take out LEDs and resistors. Place one end of the resistor to the right of the jumper wire in the same row. Attach the resistor to any column in the same row. Now place the LED. The short end of the LED is connected to the bread board on the same column of the connected jumper wire. The long end of the LED is connected on the same column as that of the resistor. Finally, connect another jumper wire from the Pin 13 in Arduino to the bread board. The other end should be connected to the same column of the connected resistor.

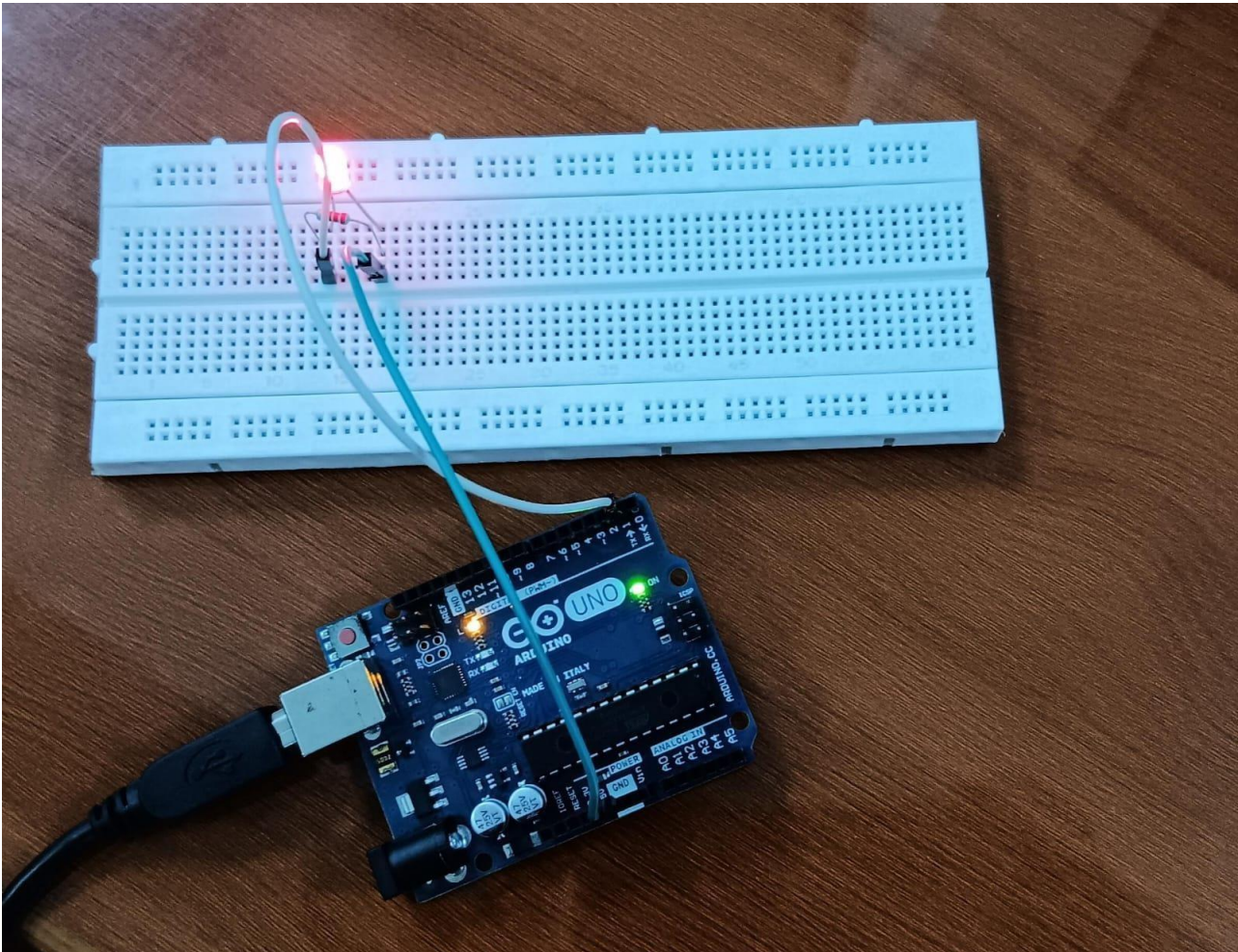
PROGRAM:

```
void setup()
```



```
{  
pinMode(13, OUTPUT);  
}  
void loop()  
{  
digitalWrite(13, HIGH);  
delay(1000);  
digitalWrite(13, LOW);  
delay(1000);  
}
```

SAMPLE INPUT AND OUTPUT:



RESULT:

Thusthe foundational knowledge on the Arduino board and the IDE is acquired byworking withthe LED light.

AD22511-INTERNET OF THINGS AND ITS APPLICATIONS LABORATORY

EX.NO:2b

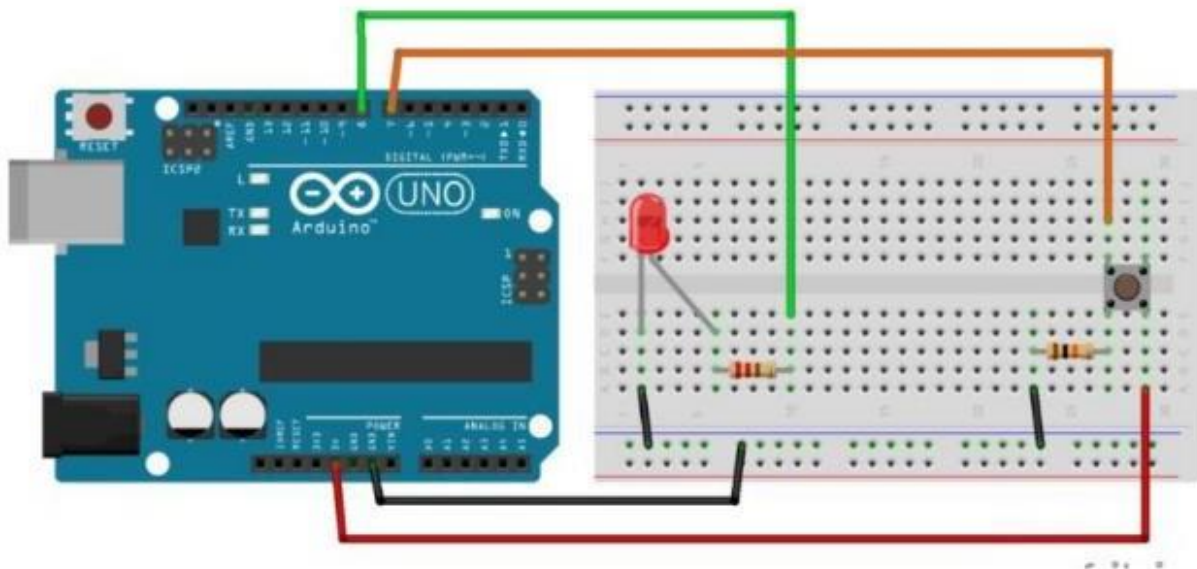
DATE:

Working with LED and button

AIM:

To understand the working of Arduino board and the IDE by working with the LED with button.

CIRCUIT DIAGRAM:



EQUIPMENT REQUIRED:

Arduino Uno R3, 1 k Ω Resistor, LED (any colour), BUTTON, bread board and jumper wires

DESCRIPTION:

- Insert one side of the jumper wire into GND on the board. Lead the other side to the breadboard. Take out LEDs and resistors. Place one end of the resistor to the right of the jumper wire in the same row.
- Attach the resistor to any column in the same row. Now place the LED. The short end of the LED is connected to the bread board on the same col-

umn of the connected jumper wire. The long end of the LED is connected on the same column as that of the resistor.

- Finally, connect another jumper wire from the Pin 13 in Arduino to the bread board. The other end should be connected the same column of the connected resistor.
- Connect the button in the breadboard. Using the Jumper Wires, connect one end to the pin 2 in Arduino board, one end to the resistor and one end to 5v in the Arduino board.

PROGRAM:

```
#define LED_PIN 13

#define BUTTON_PIN 4

byte lastButtonState = LOW;

byte ledState = LOW;

unsigned long debounceDuration = 50; // millis

unsigned long lastTimeButtonStateChanged = 0;

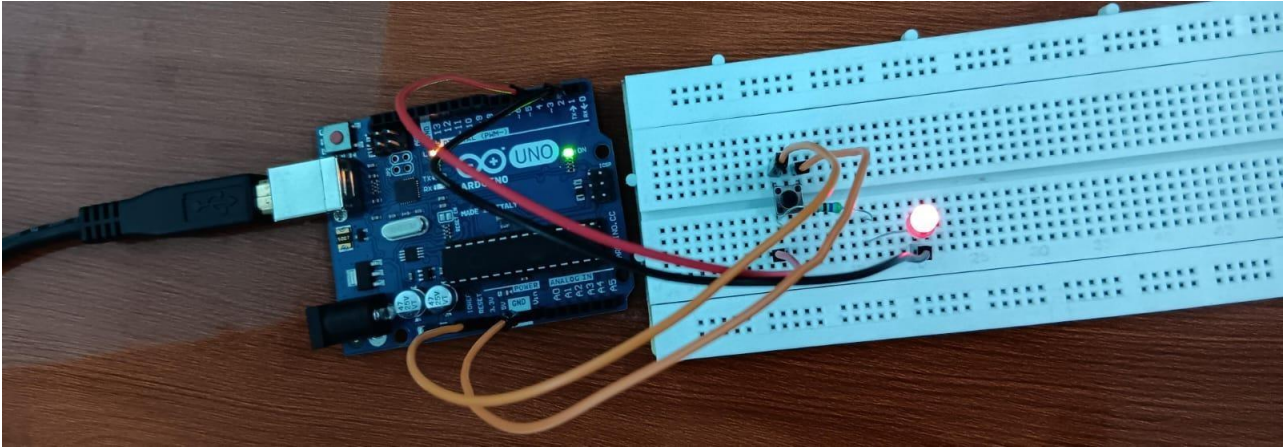
void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUTTON_PIN, INPUT);
}

void loop()
{
  if (millis() - lastTimeButtonStateChanged > debounceDuration) {
    byte buttonState = digitalRead(BUTTON_PIN);
    if (buttonState != lastButtonState) {
      lastTimeButtonStateChanged = millis();
      lastButtonState = buttonState;
      if (buttonState == LOW) {
        ledState = (ledState == HIGH) ? LOW : HIGH;
        digitalWrite(LED_PIN, ledState);
      }
    }
  }
}
```


}

}

SAMPLE INPUT AND OUTPUT:



RESULT:

Thus the C program to understand the working of Arduino board and the IDE by working with the LED with button was executed successfully.

AD22511-INTERNET OF THINGS AND ITS APPLICATIONS LABORATORY

EX.NO:2c

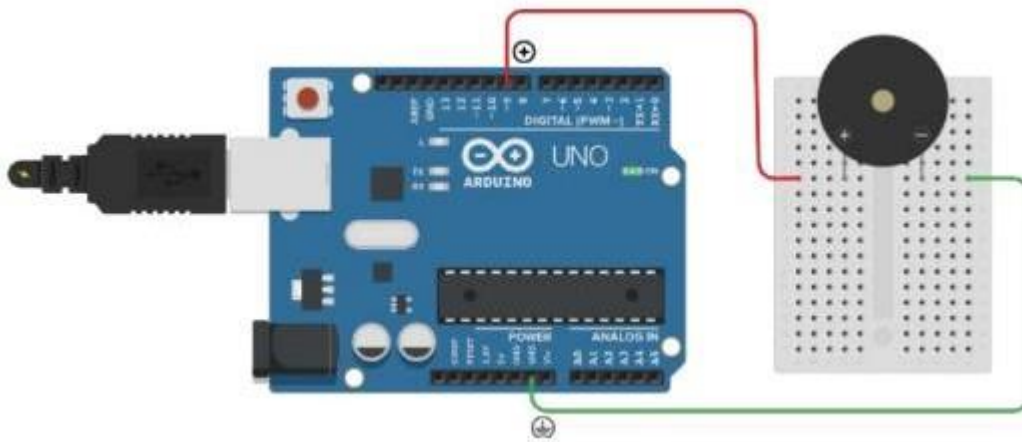
DATE:

Working with Buzzer

AIM:

To integrate Piezo (sound sensor) with Arduino Uno R3

CIRCUIT DIAGRAM:



EQUIPMENT REQUIRED:

Arduino Uno R3, Piezo (buzzer), bread board and jumper wires

DESCRIPTION:

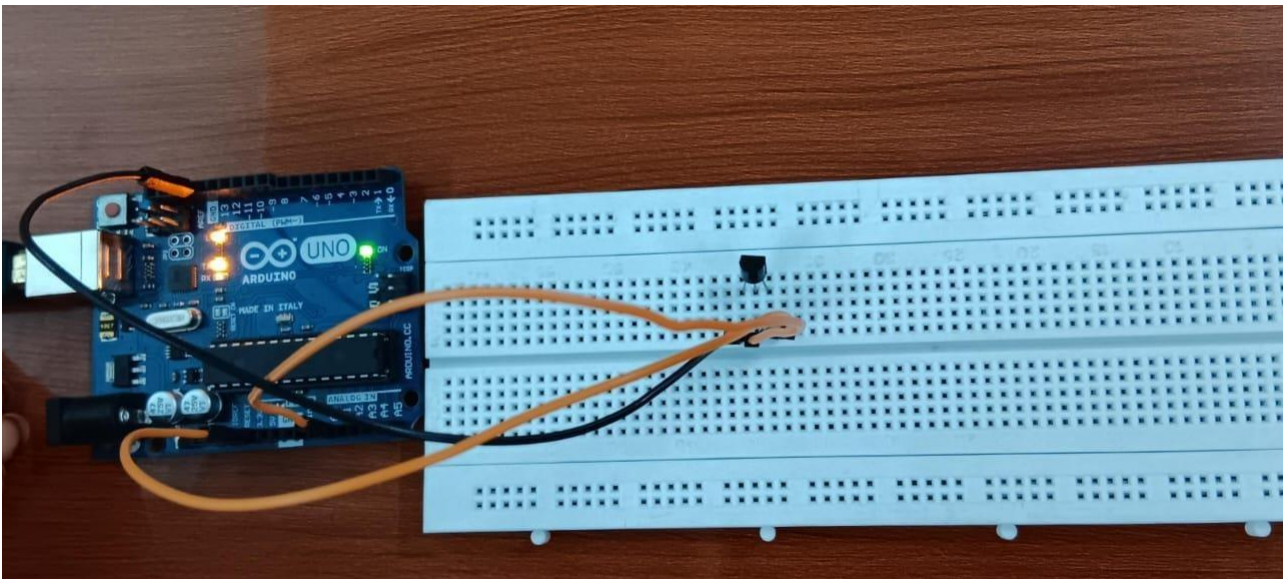
Insert one side of the jumper wire into GND on the board and the other side into the breadboard. Place the cathode of the Piezo on the same row where the wire from GND is placed. Finally, connect the jumper wire from Pin 9 of the Arduino board to the anode of Piezo in the breadboard

PROGRAM:

```
void setup(){  
  pinMode(9, OUTPUT);  
}  
void play (int cnt){
```

```
for (int i = 1; i <= 3; i += 1)
{
tone(9, 140);
delay(cnt);
noTone(9);
delay(cnt);
}
}
void loop()
{
play(500);
delay(300);
play(250);
delay(300);
}
```

SAMPLE INPUT AND OUTPUT:



RESULT:

Thus the Piezo was integrated with Arduino Uno using an appropriate program.

AD22511-INTERNET OF THINGS AND ITS APPLICATIONS LABORATORY

EX.NO:2c

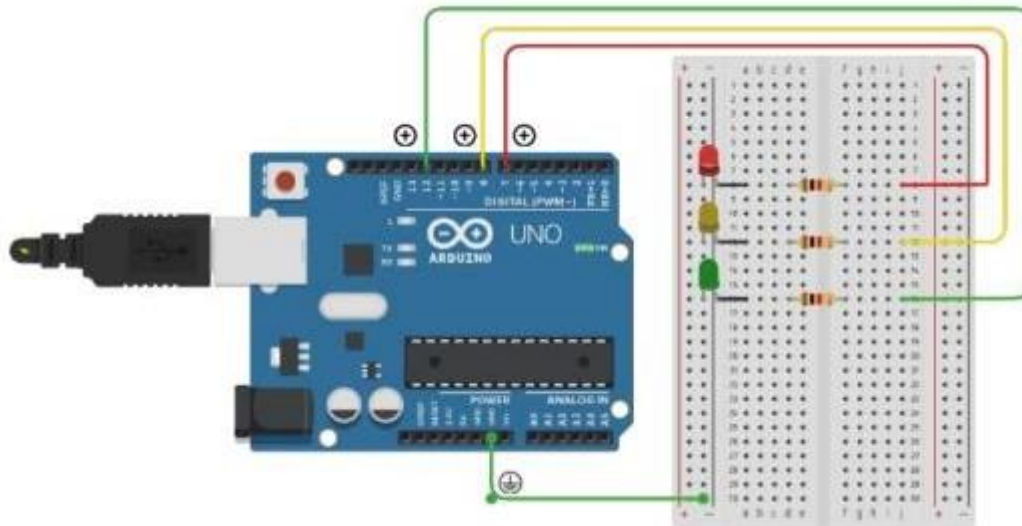
DATE:

Working with Traffic Light

AIM:

To stimulate traffic light behaviour using LEDs and Arduino Uno.

CIRCUIT DIAGRAM:



EQUIPMENT REQUIRED:

Arduino Uno R3, 1 k Ω Resistor, LED (red, green and yellow), bread board and jumper wires

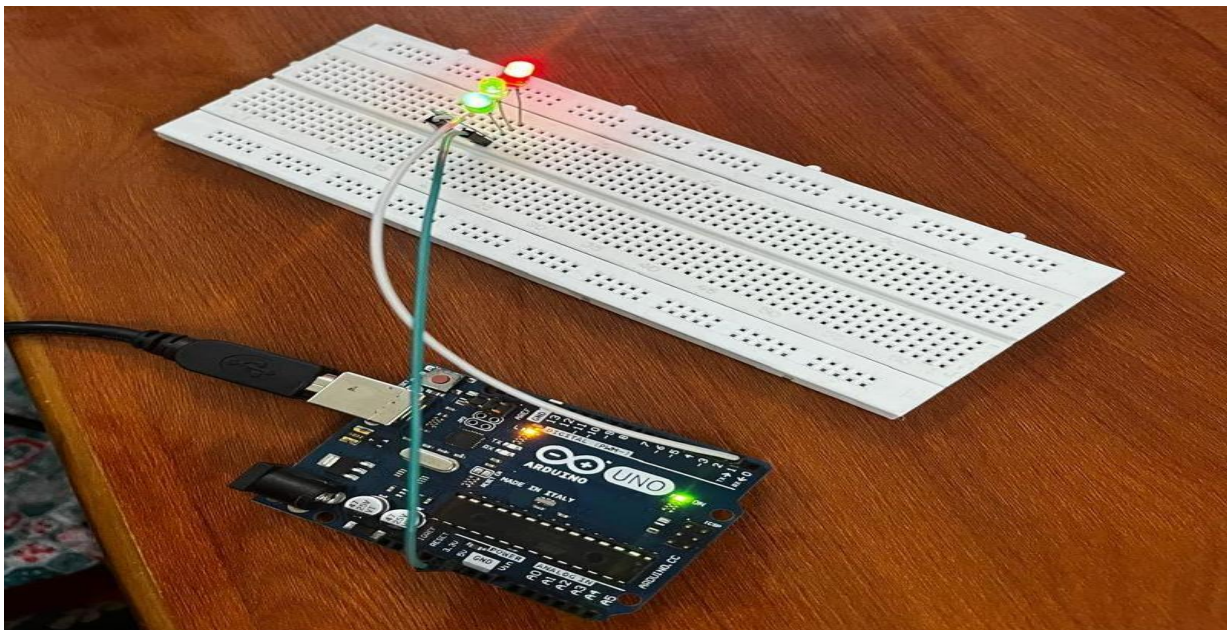
DESCRIPTION:

Insert one side of the jumper wire into GND on the board and the other side into the breadboard. Place the red, yellow and green LEDs cathodes (short end of led) parallel to each other on the same row where the wire from GND is placed. Similarly place the three resistors' one end to the anode (long end of led) of each of the LEDs. Finally, connect the jumper wires from Pin 12, Pin 8 and Pin 7 of the Arduino board to each of the resistors unconnected end in the breadboard

PROGRAM:


```
void setup(){
pinMode(7, OUTPUT);
pinMode(8, OUTPUT);
pinMode(12, OUTPUT);
digitalWrite(7, LOW);
digitalWrite(8, LOW);
digitalWrite(12, LOW);
}
void loop(){
digitalWrite(12, HIGH);
digitalWrite(8, LOW);
digitalWrite(7, LOW);
delay(1000);
digitalWrite(12, LOW);
digitalWrite(8, HIGH);
digitalWrite(7, LOW);
delay(1000);
digitalWrite(12, LOW);
digitalWrite(8, LOW);
digitalWrite(7, HIGH);
delay(1000);
}
```

SAMPLE INPUT AND OUTPUT:



RESULT:

Thus to stimulate traffic light behaviour using LEDs and Arduino Uno was executed successfully.

AD22511-INTERNET OF THINGS AND ITS APPLICATIONS LABORATORY

EX.NO:2d

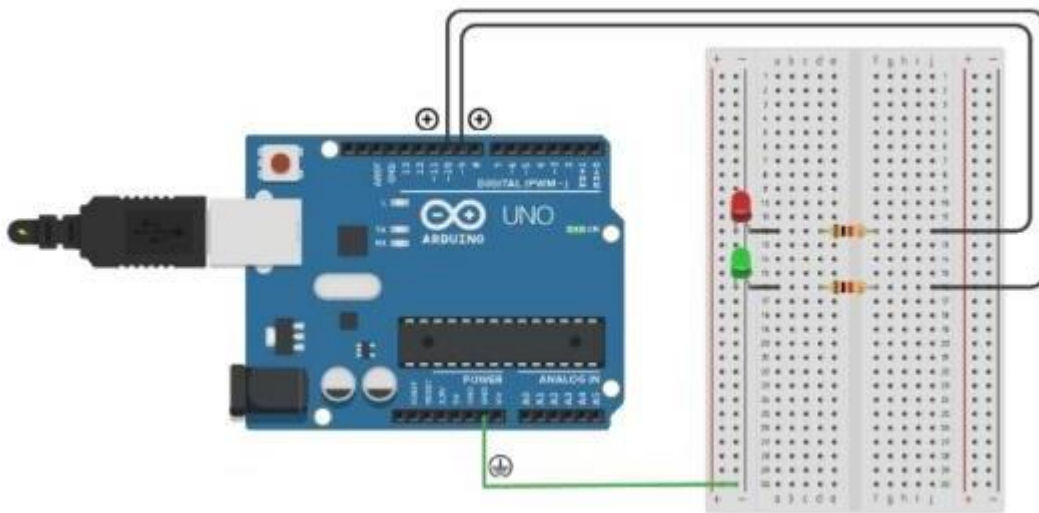
DATE:

Controlling LED intensity using PWM signal

AIM:

To control the intensity of the LED's using PWM signal with the help of Arduino Uno R3

CIRCUIT DIAGRAM:



EQUIPMENT REQUIRED:

Arduino Uno R3, 1 k Ω Resistor, LED (red and green), bread board and jumper wires

DESCRIPTION:

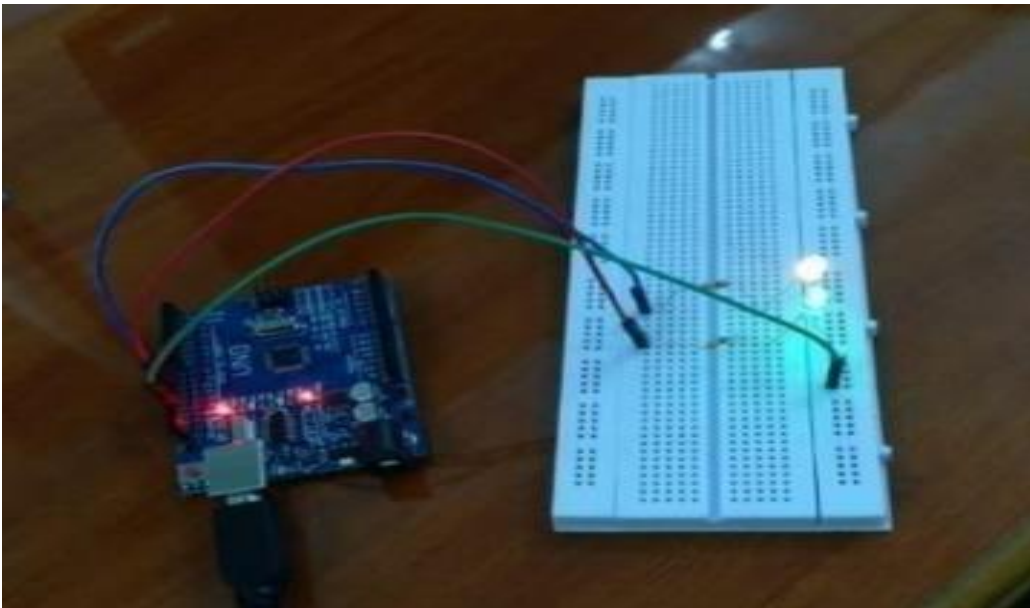
Insert one side of the jumper wire into GND on the board and the other side into the breadboard. Place the red and green LEDs cathodes (short end of led) parallel to each other on the same row where the wire from GND is placed. Similarly place the two resistor's one end to the anode (long end of led) of each of the LEDs. Finally, connect the jumper wires from Pin 10 and Pin 9 of the Arduino board to each of the resistors unconnected end in the breadboard

PROGRAM:

```
void setup()
{
  pinMode(9, OUTPUT);
}
```

```
pinMode(10, OUTPUT);
analogWrite(9, 0);
analogWrite(10, 255);
}
void loop()
{
int i = 0;
for (i = 50; i <= 255; i += 10)
{
analogWrite(9, i);
analogWrite(10, 255 - i);
delay(500);
}
for (i = 50; i <= 255; i += 10)
{
analogWrite(9, 255 - i);
analogWrite(10, i);
delay(500);
}
}
```

SAMPLE INPUT AND OUTPUT:



RESULT:

Thus the intensity of the given LED's were controlled using Arduino Uno with the appropriate program.

AD22511-INTERNET OF THINGS AND ITS APPLICATIONS LABORATORY

EX.NO:3

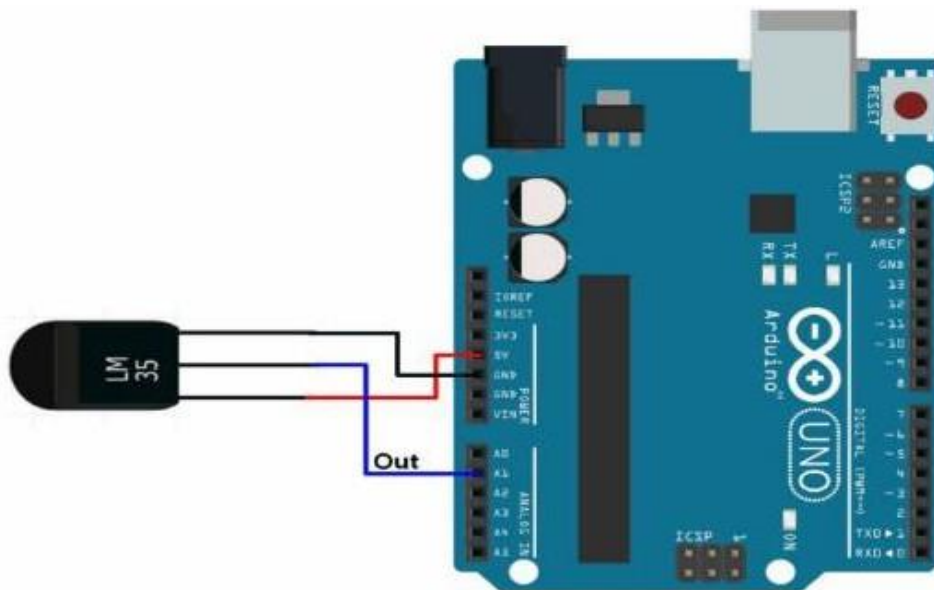
DATE:

Temperature Monitoring

AIM:

To Study the Temperature sensor and Write Program for monitoring temperature using Arduino

CIRCUIT DIAGRAM:



EQUIPMENT REQUIRED:

Arduino Uno R3, LM35 sensor, bread board and jumper wires

DESCRIPTION:

LM35 Sensor:

- LM35 is a temperature sensor which can measure temperature in the range of -55°C to 150°C.
- It is a 3-terminal device that provides analog voltage proportional to the temperature. Higher the temperature, higher is the output voltage.
- The output analog voltage can be converted to digital form using ADC so that a microcontroller can process it.

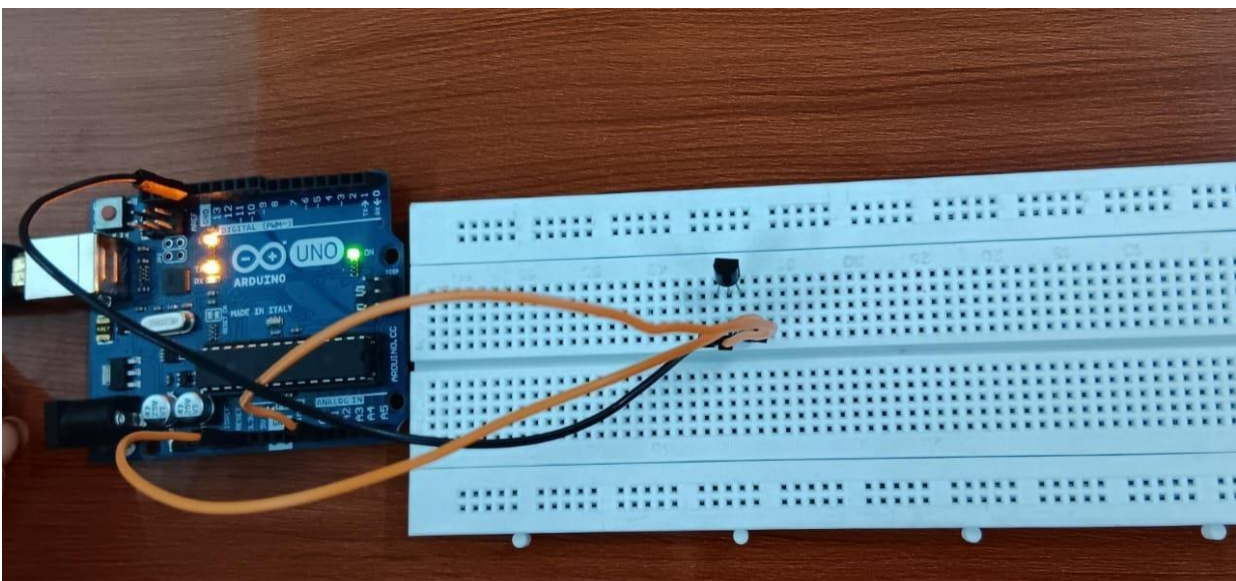
PROCEDURE:

1. Insert LM35 sensor in the breadboard. Connect LM35 sensor middle point from breadboard to Analog input A1 in Arduino
2. Insert one end of LM35 sensor pin from breadboard to Gnd in Arduino
3. Insert other end of LM35 sensor Pin from breadboard to 5v in Arduino
4. Upload the program and View the output in the serial monitor in the Arduino IDE software.
5. We can see the changes in the temperature when the variations are noted down.

PROGRAM:

```
const int lm35_pin = A1; /* LM35 O/P pin */
void setup() {
  Serial.begin(9600);
}
void loop() {
  int temp_adc_val;
  float temp_val;
  temp_adc_val = analogRead(lm35_pin); /* Read Temperature */
  temp_val = (temp_adc_val * 4.88); /* Convert adc value to equivalent voltage */
  temp_val = (temp_val/10); /* LM35 gives output of 10mv/° C */
  Serial.print("Temperature = ");
  Serial.print(temp_val);
  Serial.print(" Degree Celsius\n");
  delay(1000);
}
```

SAMPLE INPUT AND OUTPUT:





```
temperature = 46.36 Degree Celsius
temperature = 47.34 Degree Celsius
temperature = 48.31 Degree Celsius
temperature = 50.75 Degree Celsius
temperature = 54.17 Degree Celsius
temperature = 56.12 Degree Celsius
temperature = 56.61 Degree Celsius
temperature = 54.66 Degree Celsius
temperature = 53.19 Degree Celsius
temperature = 52.22 Degree Celsius
temperature = 51.24 Degree Celsius
temperature = 50.26 Degree Celsius
temperature = 48.31 Degree Celsius
temperature = 46.85 Degree Celsius
temperature = 45.38 Degree Celsius
temperature = 43.43 Degree Celsius
temperature = 42.46 Degree Celsius
temperature = 41.48 Degree Celsius
```

RESULT:

Thus the Temperature sensor was integrated with Arduino Uno using an appropriate program.

AD22511-INTERNET OF THINGS AND ITS APPLICATIONS LABORATORY

EX.NO:4

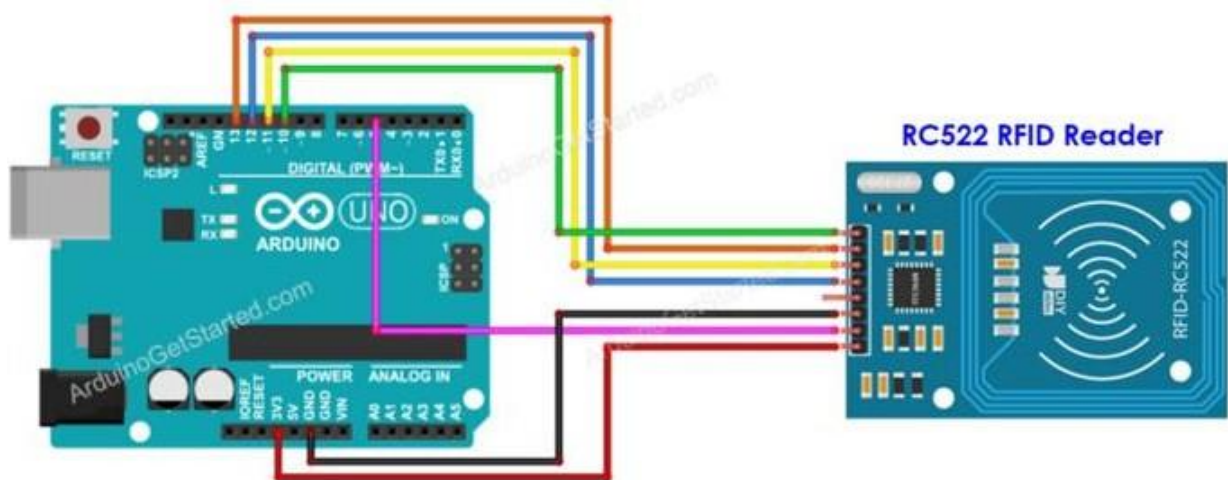
DATE:

Implementation of RFID, NFC

AIM:

To implement a access of ID using RFID/ NFC sensor and check the permission.

CIRCUIT DIAGRAM:



EQUIPMENT REQUIRED:

Arduino UNO R3, RC522 Reader, jumper wires, breadboard, RFID/NFC sensor

PROCEDURE:

1. Connect the RFID/NFC reader module to the breadboard.
2. Give the following connection from Breadboard to Arduino

RFID / NFC	ARDUINO
SS(SDA)	D10
SCK	D13
MOSI	D11
MISO	D12
IRQ	NOT CONNECTED
GND	GND
RST	D5
VCC	3.3 V

3. After giving connections type the program to the arduino module.

4. Install the MFRC522 header file from library (Tools-> Manage Libraries)
5. Now upload the program in the arduino IDE.
6. Place the RFID/ NFC card near the RFID sensor.
7. Change the UID number from the serial monitor based on the card detected.
8. UID card will be detected and access is granted / denied in the serial monitor will be displayed.

PROGRAM:

```
#include <SPI.h>
#include <MFRC522.h>
#define SS_PIN 10
#define RST_PIN 9

MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance.

void setup() {
  Serial.begin(9600);
  SPI.begin(); // Initiate SPI bus
  mfrc522.PCD_Init(); // Initiate MFRC522
  Serial.println("Approximate your card to the reader...");
  Serial.println();
}

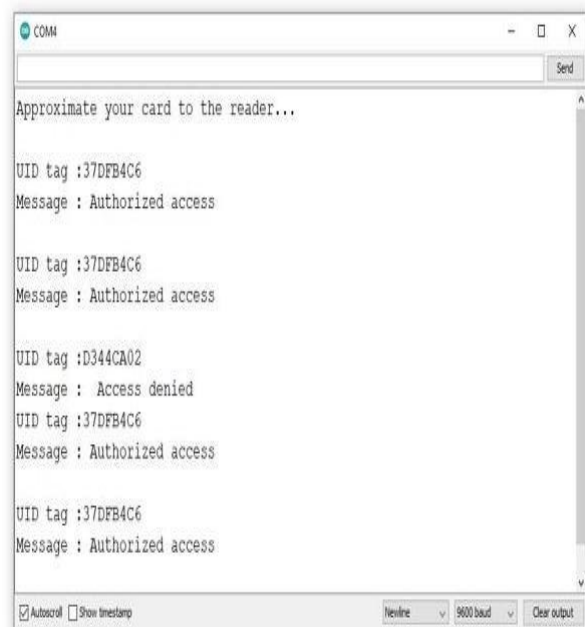
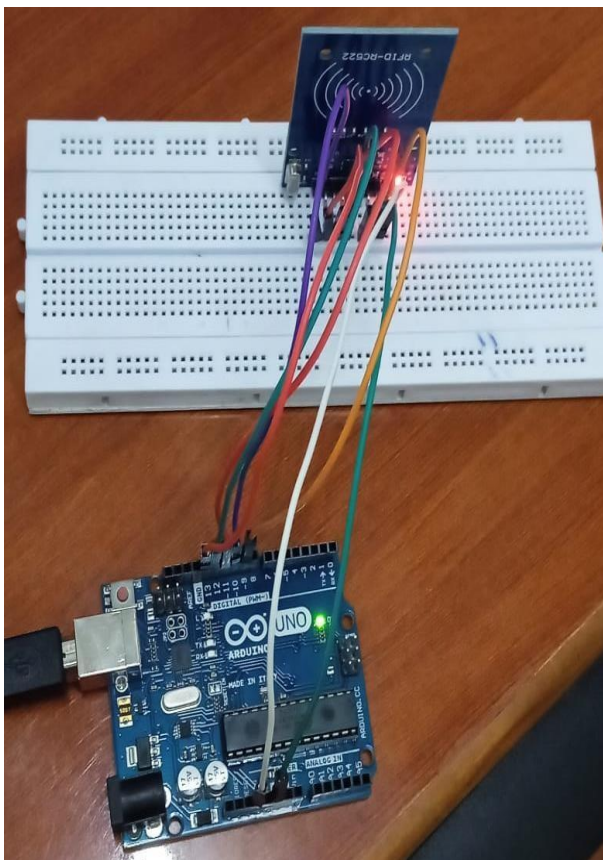
void loop() {
  // Look for new cards
  if (!mfrc522.PICC_IsNewCardPresent()) {
    return;
  }
  // Select one of the cards
  if (!mfrc522.PICC_ReadCardSerial()) {
    return;
  }
  // Show UID on serial monitor
  Serial.print("UID tag :");
  String content = "";
  byte letter;
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
    content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
    content.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
  Serial.println();
  Serial.print("Message : ");
  content.toUpperCase();
```

```

    if (content.substring(1) == "2C 1F F7 21") { // Replace with your UID
Serial.println("Authorized access");
Serial.println();
delay(3000);
    } else {
Serial.println("Access denied");
delay(3000);
    }
}

```

SAMPLE INPUT AND OUTPUT:



RESULT:

Thus the RFID/ NFC reader is implemented successfully and the NFC tag & reader is detected.

AD22511-INTERNET OF THINGS AND ITS APPLICATIONS LABORATORY

EX.NO:5

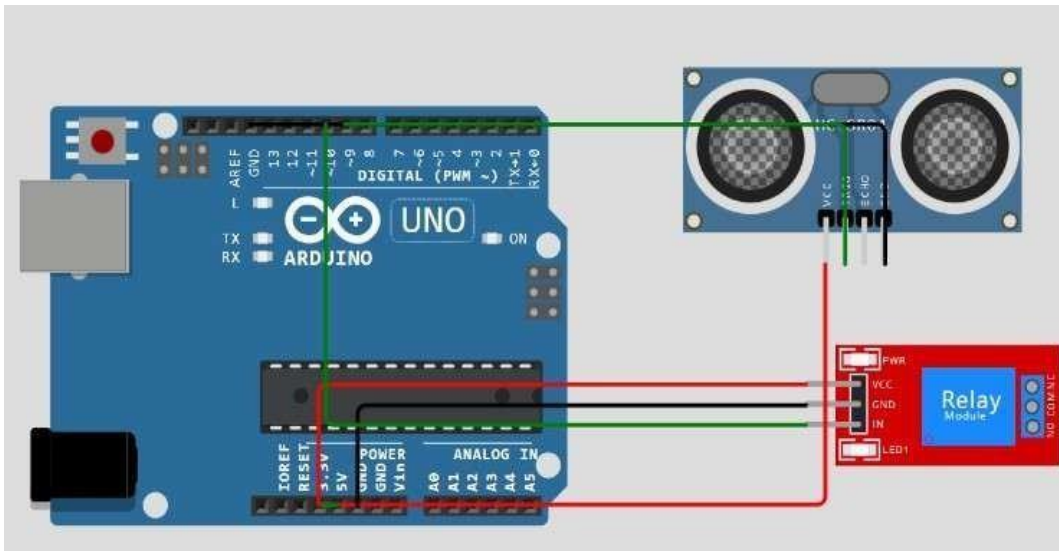
DATE:

Motion detection using ultrasonic sensor

AIM:

To create an experimental setup using an ultrasonic sensor to detect motion and control a relay.

CIRCUIT DIAGRAM:



EQUIPMENT REQUIRED:

- 1) Arduino UNO (1x)
- 2) Ultrasonic Sensor (HC-SR04) (1x)
- 3) Breadboard and Jumper Wires
- 4) Power Supply (5V for Arduino and relay module)
- 5) Load (e.g., a light bulb or motor) (Optional, connected through the relay)

DESCRIPTION:

1. Connect the ultrasonic sensor and relay module to the breadboard.
2. Connect ultrasonic sensor pins Vcc to 3.3 v to arduino ,Gnd to Gnd ,Echo pin to 8, Trig pin to 9 in arduino.
3. Connect Relay Pin to vcc to 5 v in arduino ,gnd to gnd in arduino,IN to pin 10 in arduino.
4. Connect bulb to NO/ COM in Relay module if necessary.
5. Upload the program in Arduino and the Distance measurement is shown in the Serial Monitor.

PROGRAM:

```
#define trigPin 9
#define echoPin 8
#define relayPin 10
void setup() {

    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    pinMode(relayPin, OUTPUT);
    Serial.begin(9600);
}
void loop() {
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

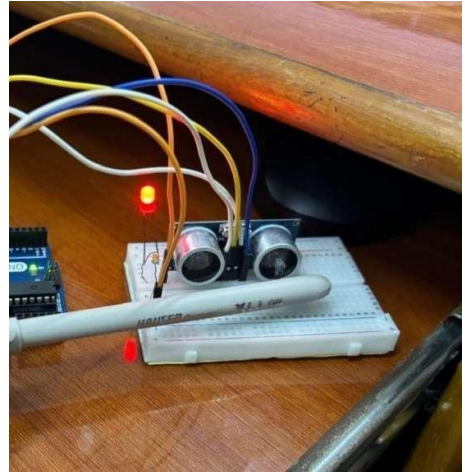
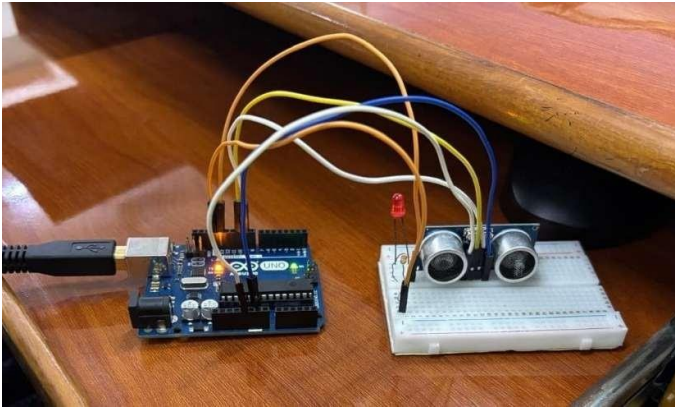
    long duration = pulseIn(echoPin, HIGH);
    float distance = (duration * 0.034) / 2;

    Serial.print("&quot;Distance: &quot;");
    Serial.print(distance);
    Serial.println("&quot; cm&quot;");

    if (distance < 20) {
        digitalWrite(relayPin, HIGH);
    } else {
        digitalWrite(relayPin, LOW);
    }

    delay(100);
}
```


SAMPLE INPUT AND OUTPUT:



```
14:58:08.011 -> Distance: 4
14:58:08.058 -> Distance: 5
14:58:08.199 -> Distance: 5
14:58:08.292 -> Distance: 4
14:58:08.386 -> Distance: 4
14:58:08.480 -> Distance: 5
14:58:08.574 -> Distance: 7
14:58:08.667 -> Distance: 5
14:58:08.761 -> Distance: 6
14:58:08.902 -> Distance: 96
14:58:08.995 -> Distance: 10
14:58:09.089 -> Distance: 17
```

☐ Autoscroll ☒ Show timestamp

Newline

9600 baud

Clear output

RESULT:

Thus, the ultrasonic sensor is implemented successfully, and motion detection is achieved through accurate distance measurement and real-time response to object movement.

AD22511-INTERNET OF THINGS AND ITS APPLICATIONS LABORATORY

EX.NO:6

DATE:

Exploring the data from the sensors to Cloud Environment.

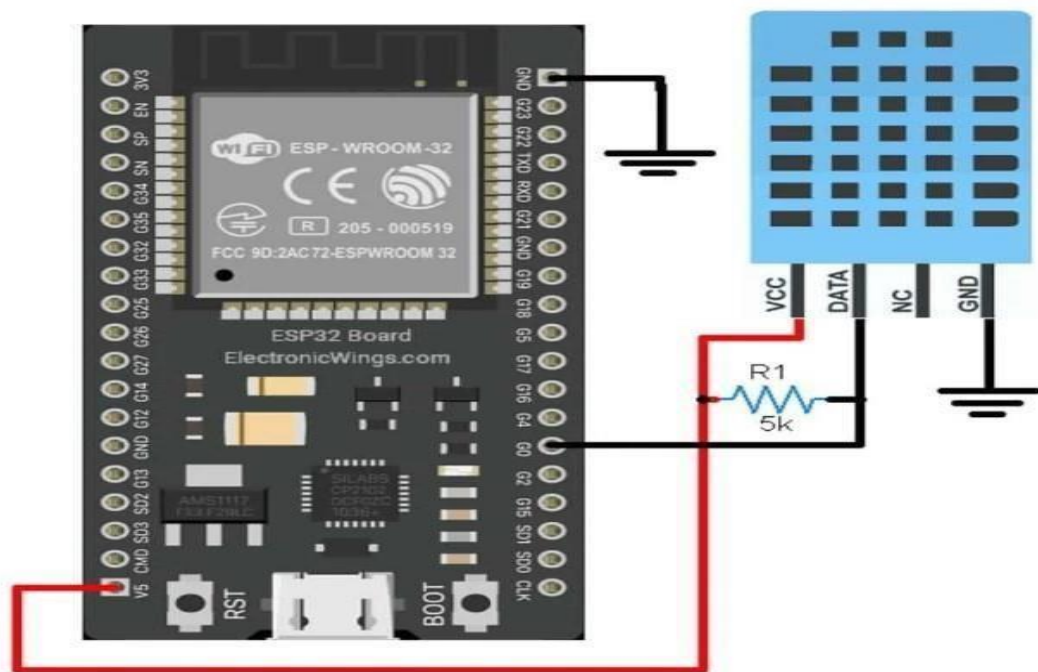
Aim:

Using ThingSpeak Cloud Reading Temperature Sensor Monitoring with NodeMCU /Raspberry Pi.

Components Required:

- Esp 32
- DHT11 Sensor
- Jumper Wires •Breadboard
- Computer with Arduino IDE Software.
- USB Cable

Layout Diagram



ESP32 interfaced with DHT11

Procedure:

- Temperature Monitoring using DHT11 & ESP32 on ThingSpeak.
- Connect ESP32 with DHT11 sensor as in the above diagram.
- It explains how to log Temperature data on the cloud.

- We can use Thingspeak as a cloud service provider and DHT11 to measure temperature and Humidity
- We use ESP32 on Arduino IDE. You can also configure the ESP32 board with Arduino to monitor temperature. NodeMCU is an open source platform based on ESP32 which can connect objects and let data transfer using the Wi-Fi protocol.

PROGRAM:

```
#include<WiFi.h>#include
<ThingSpeak.h>
#define ADC_VREF_mV      3300.0
#define ADC_RESOLUTION 4096.0
#define PIN_LM3536
const char* ssid = "jaya";
char* password = "12345678";
WiFiClient client;
unsigned long myChannelNumber=2703584;
const char* myWriteAPIKey="8QBJZ2LG4UIQGZEF";
lastTime = 0;
unsigned long timerDelay=30000;
void setup(){
    Serial.begin(115200);
    WiFi.begin(ssid,password);
    while (WiFi.status() != WL_CONNECTED) {
        Serial.print("Connecting to WiFi.. ");
        delay(500);
    }
    Serial.println("\nConnected to WiFi.");
    ThingSpeak.begin(client);
}

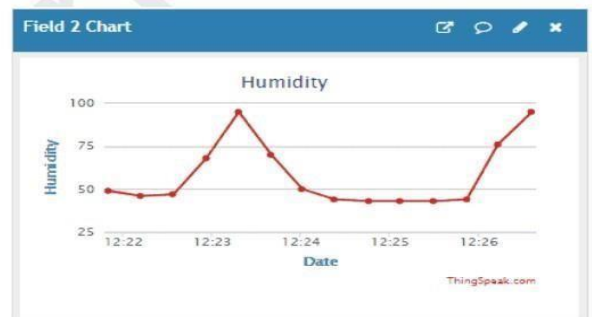
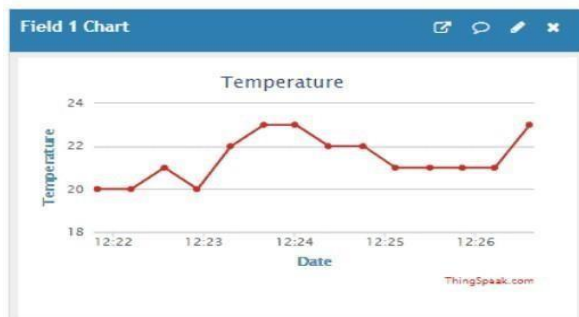
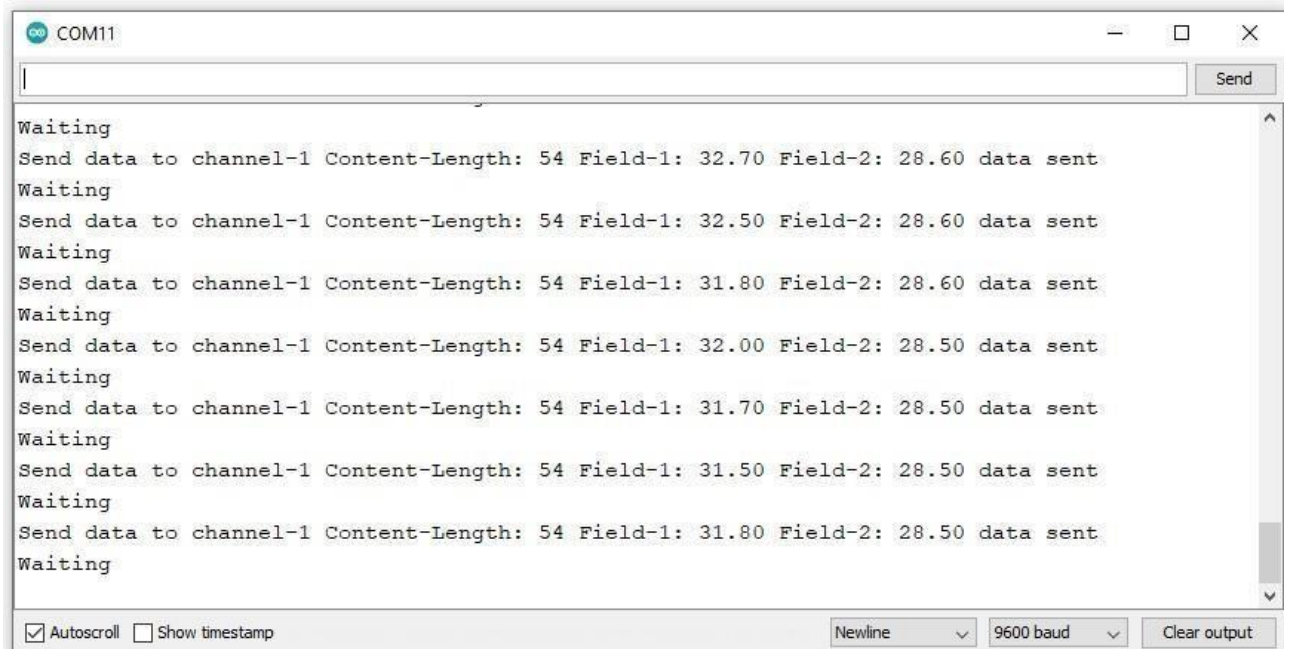
void loop(){
    if(millis()-lastTime>timerDelay){
        float milliVolt=adcVal*(ADC_VREF_mV/ADC_RESOLUTION);
        tempC = milliVolt / 10;
        float tempF=tempC *9/5+ 32;
        Serial.print("Temperature:");Serial.print(tempC);
        Serial.print("°C~");
        Serial.print(tempF);
        Serial.println("°F");
        ThingSpeak.setField(1, tempF);
        int x=ThingSpeak.writeFields(myChannelNumber,myWriteAPIKey);
        if(x=tempF){
            Serial.println("Channel update successful.");
        }else{
            Serial.print("Problem updating channel. HTTP error code:");
            Serial.println(x);
        }
    }
}
```

```

    }
    lastTime = millis();
  }
}

```

Output



RESULT:

Thus, the exploring the data from the sensors to Cloud Environment program was executed successfully.

AD22511-INTERNET OF THINGS AND ITS APPLICATIONS LABORATORY

EX.NO:8

DATE:

WAP for LED blink using Raspberry Pi

Aim:

To Configure the LED Blink in the RaspberryPi OS using the PythonIDE Environment.

Components Required

- RaspberryPi3
- LED(anycolor)
- 220 Ω resistor(to protect the LED)
- Breadboard
- Jumperwires

Procedure:

Step1:SetUpthe Hardware

1. Connect the LED to the Breadboard:

- Place the LED on the breadboard. The longer leg(anode) is the positive side, and the shorter leg (cathode) is the negative side.

2. Connect the Resistor:

- Connect a 220 Ω resistor between the short leg (cathode) of the LED and ground (GND) on the Raspberry Pi.

3. Connect the LED to the GPIO Pin:

- Use a jumper wire to connect the long leg (anode) of the LED to GPIO pin18

4. Complete the Circuit:

- Connect another jumper wire from the GND pin on the RaspberryPi to the negative rail of the breadboard where the resistor is connected.

Program:

1. Install RPi.GPIO(if not installed):

- Open a terminal on your RaspberryPi and install the GPIO library if it's not already installed:

```
sudo apt-get update  
sudo apt-get install python3-rpi.gpio
```

2. Create a Python file using a text editor (such as `nano` or `Thonny`):

```
nanoled_blink.py
```


3. Write the Following Python Code:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
LED_PIN = 18
GPIO.setup(LED_PIN, GPIO.OUT)
while True:
    GPIO.output(LED_PIN, GPIO.HIGH)
    time.sleep(1)
    GPIO.output(LED_PIN, GPIO.LOW)
    time.sleep(1)
except KeyboardInterrupt:
    pass
GPIO.cleanup()
```

4. Save and Exit:

- After typing the code, save the file (`Ctrl+O` in `nano`) and exit (`Ctrl+X`).

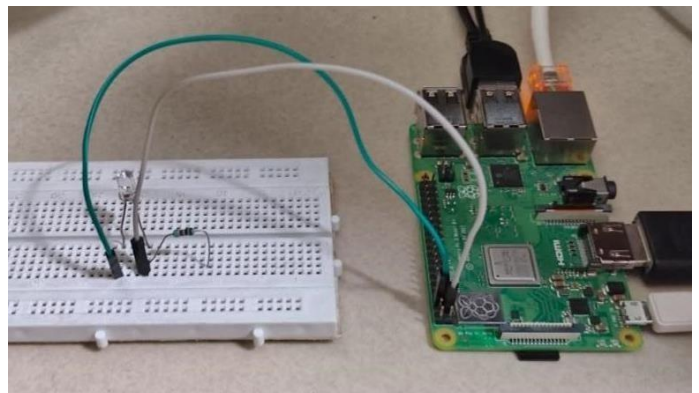
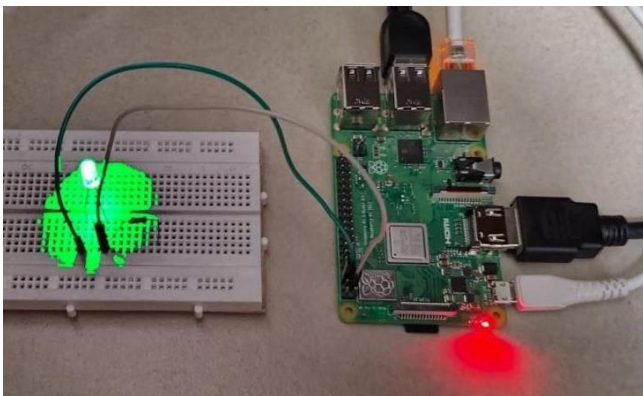
5. Run the Program

`python3 led_blink.py`

- The LED should start blinking, turning on for 1 second and off for 1 second in a continuous loop.

6. You can stop the program by pressing `Ctrl + C`.

Output:



Result:

Thus, LED Blink is executed by using Raspberry Pi Successfully.

AD22511-INTERNET OF THINGS AND ITS APPLICATIONS LABORATORY

EX.NO:7

DATE:

Study and Configure Raspberry Pi

AIM:

To study and install the OS in the RaspberryPi and Configure it.

COMPONENTSREQUIRED:

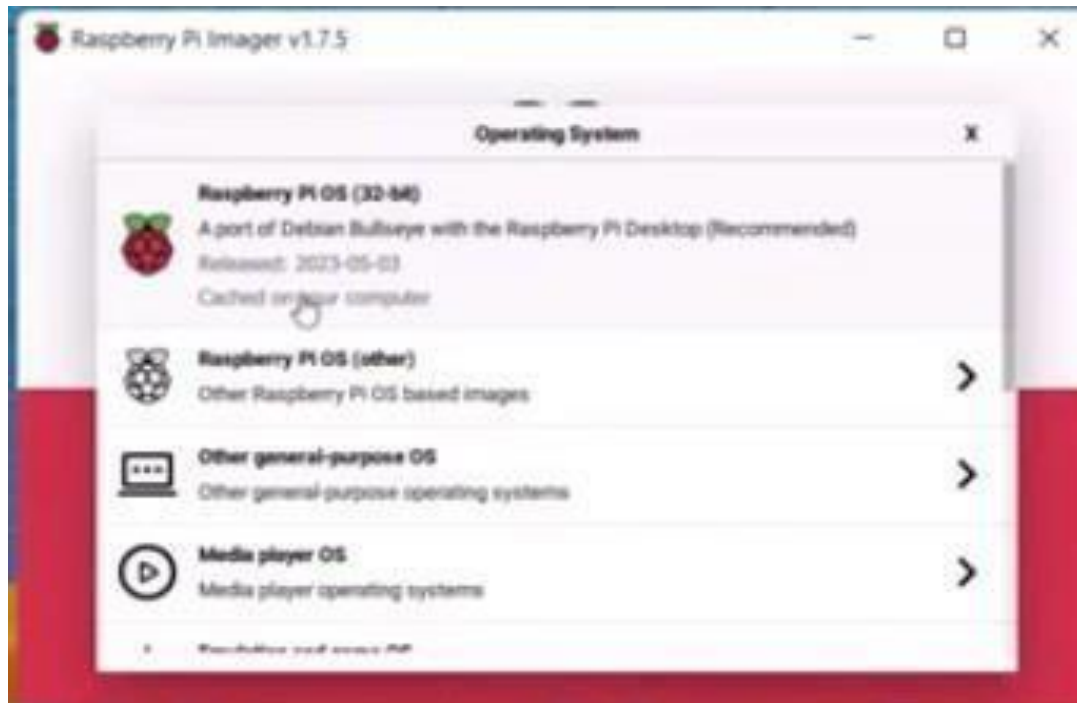
- Raspberry Pi
- SD card reader with SD card for storing RaspberryPI OS
- Data Cable to connect Raspberry Pi

PROCEDURE:

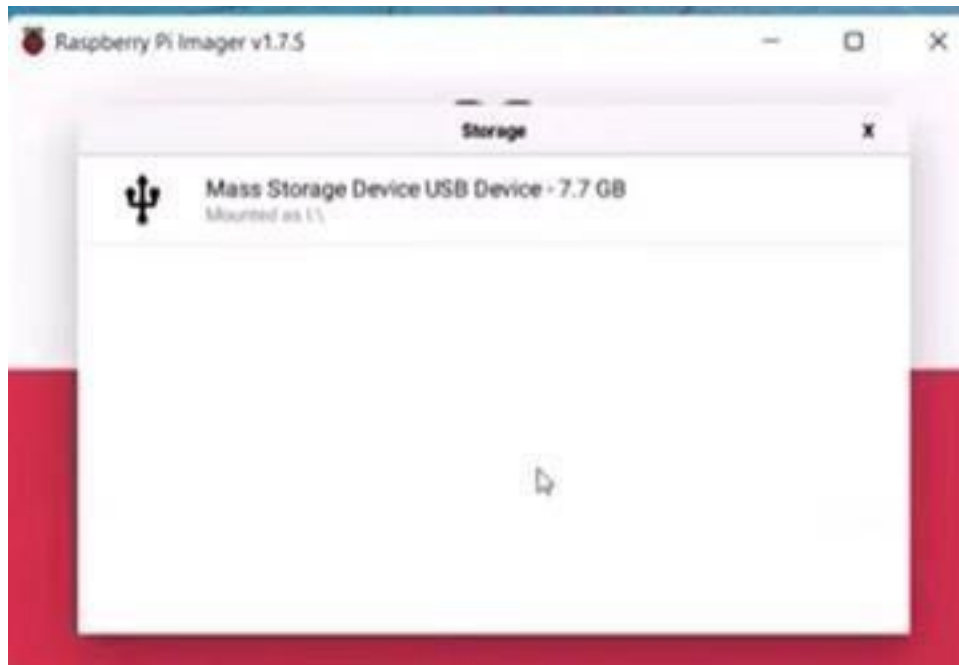
1. Before Starting the Procedure we must download the following software to install the Raspberry Pi.
2. The link to download the softwares are as follows

Raspberry Pi Imager - <https://www.raspberrypi.com/software/> Putty Software - <https://www.putty.org/>
VNCviewer/Server-<https://www.realvnc.com/en/connect/do...>

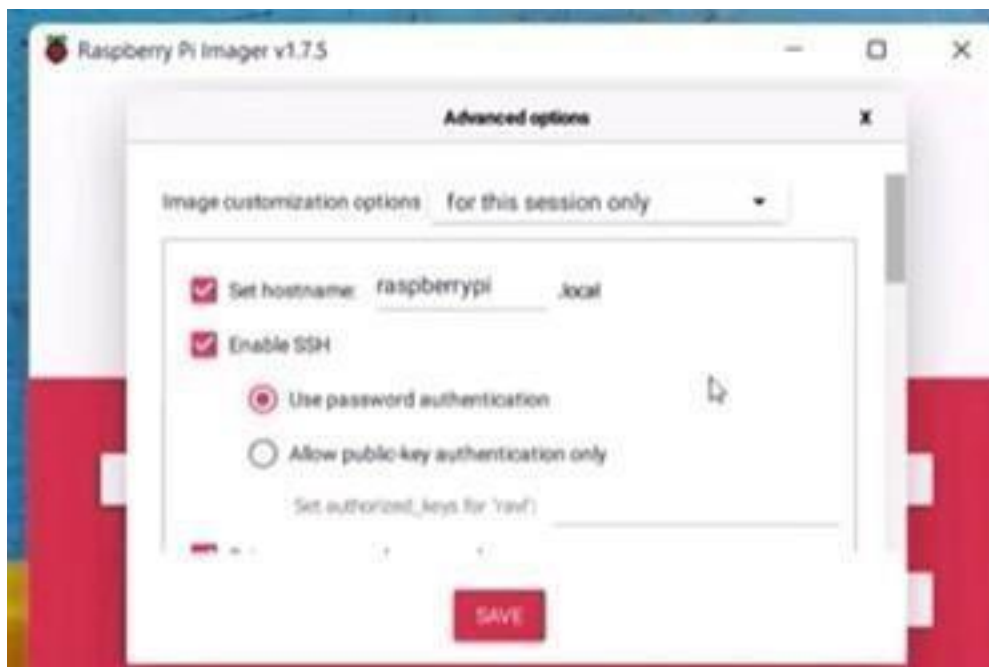
3. Download the files and install it into your Computer
4. Connect the SD card Reader which has SD card of minimal storage of 16GB into the System
5. Open RaspberryPi Imager, Choose RaspberryPi OS32bit.



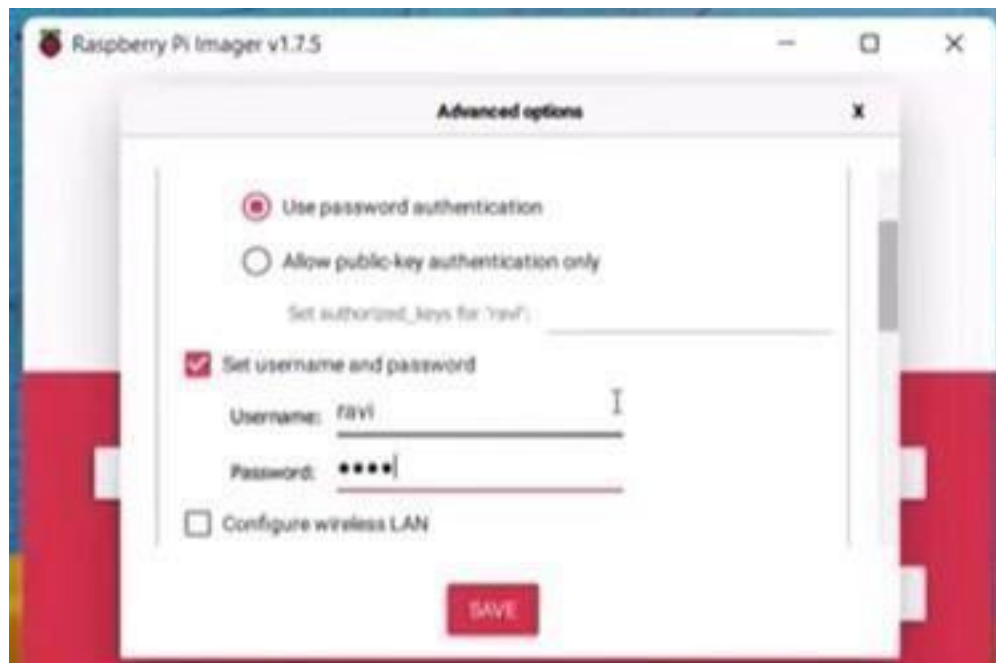
1. Choose Storage as Mass Storage device(SD Card)



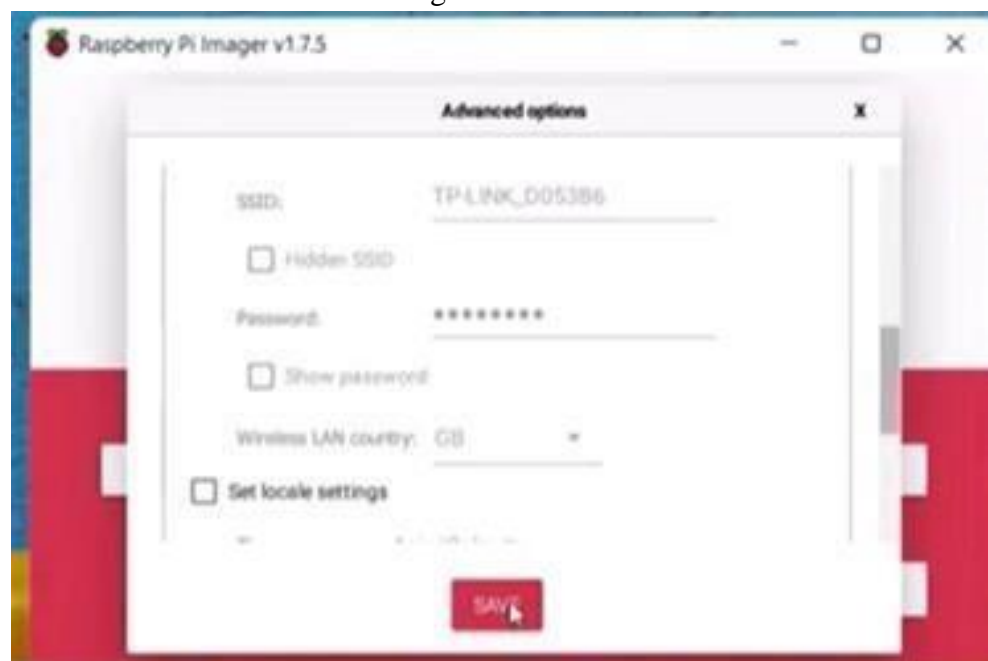
2. Goto settings->Check Host name and give name as Raspberry Pi.local



3. Check Enable SSH, Choose Password Authentication
4. Set username and password and click on save

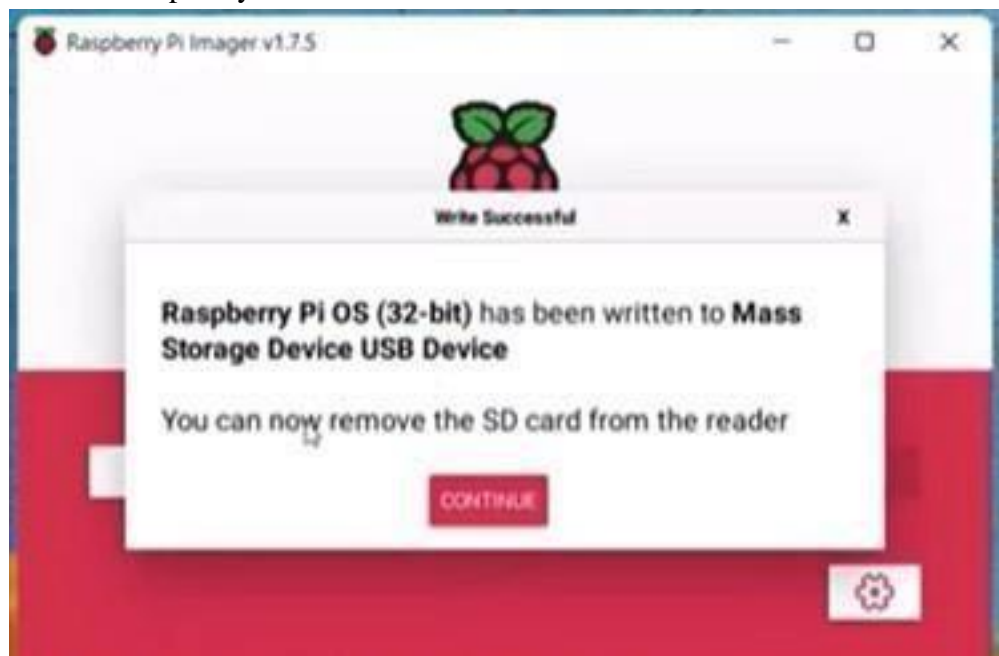


5. Then Click on write .Continue for erasing data in SD card.



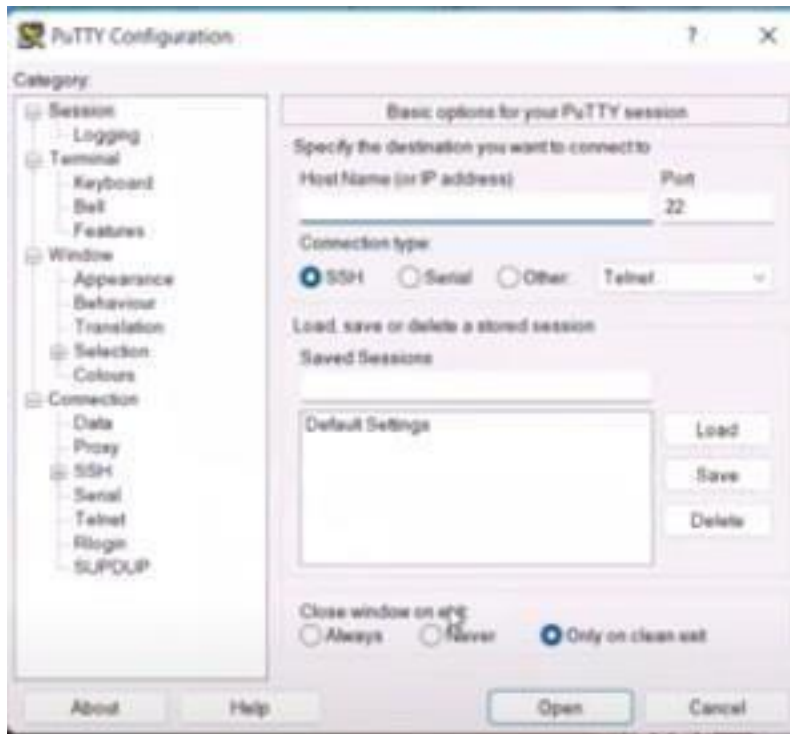


6. It will write the Raspberry Pi os into the SD card.

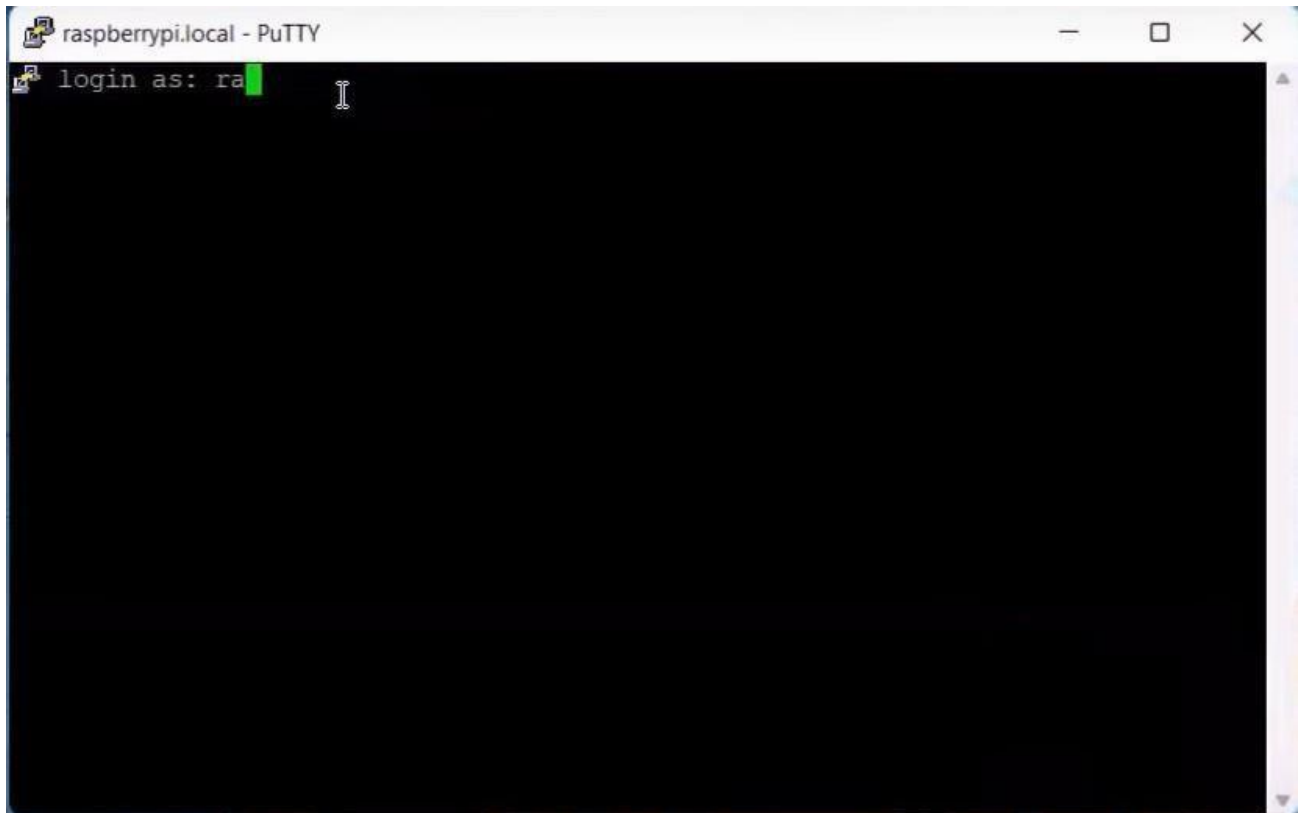


7. Eject the card reader and insert the card reader into the computer
8. Create a file called SSH. It should be simply a File. It should be named as SSH file. (Right click -> New -> text document. Delete .txt extension)
9. Eject the memory card from reader and insert into the raspberry Pi.
10. Connect the LAN cable (Ethernet cable) from your computer and power cable to raspberry pi

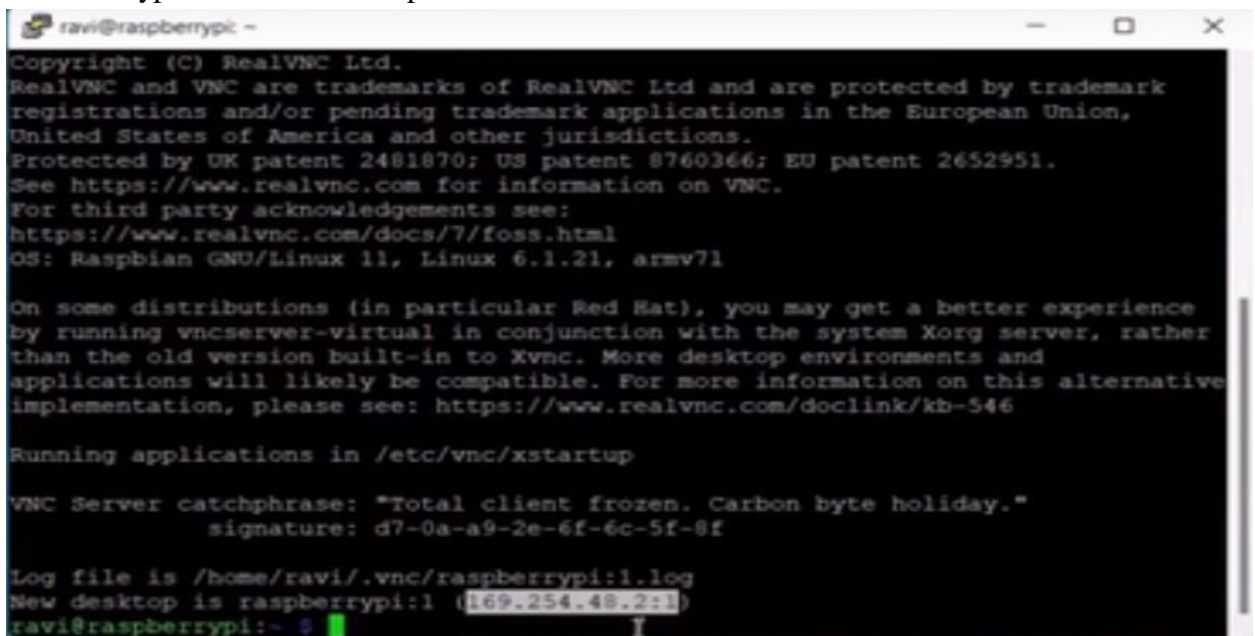
11. Open Putty Configuration and type the host name Raspberry Pi.local and select connection type as SSH, Click on open



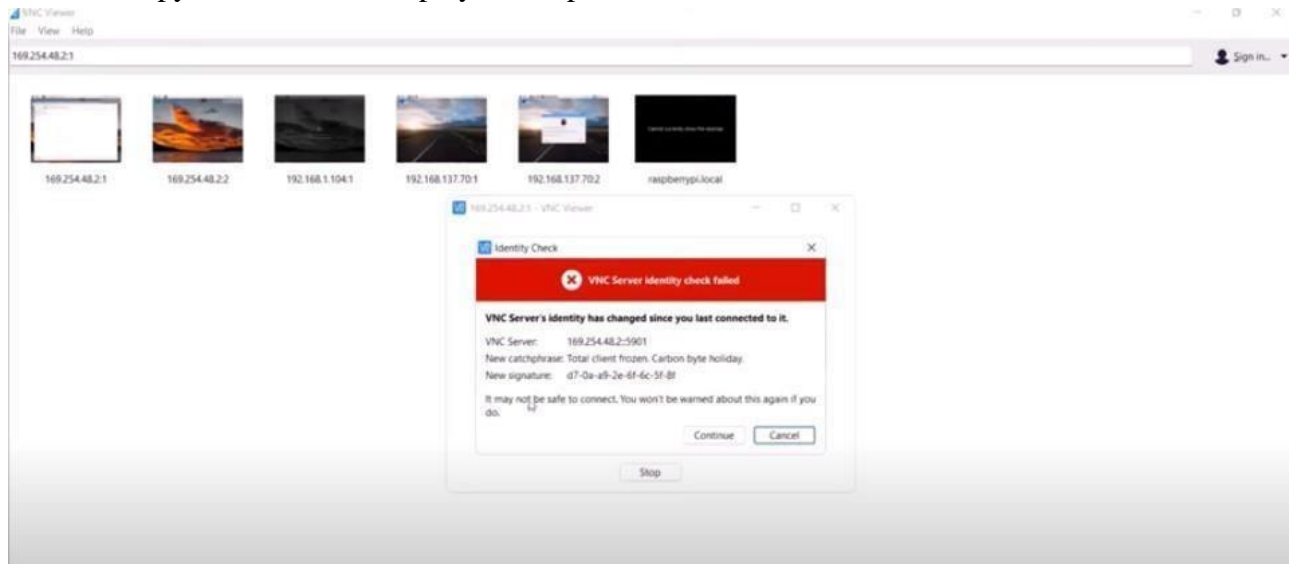
12. Login page will be displayed. Enter username and password



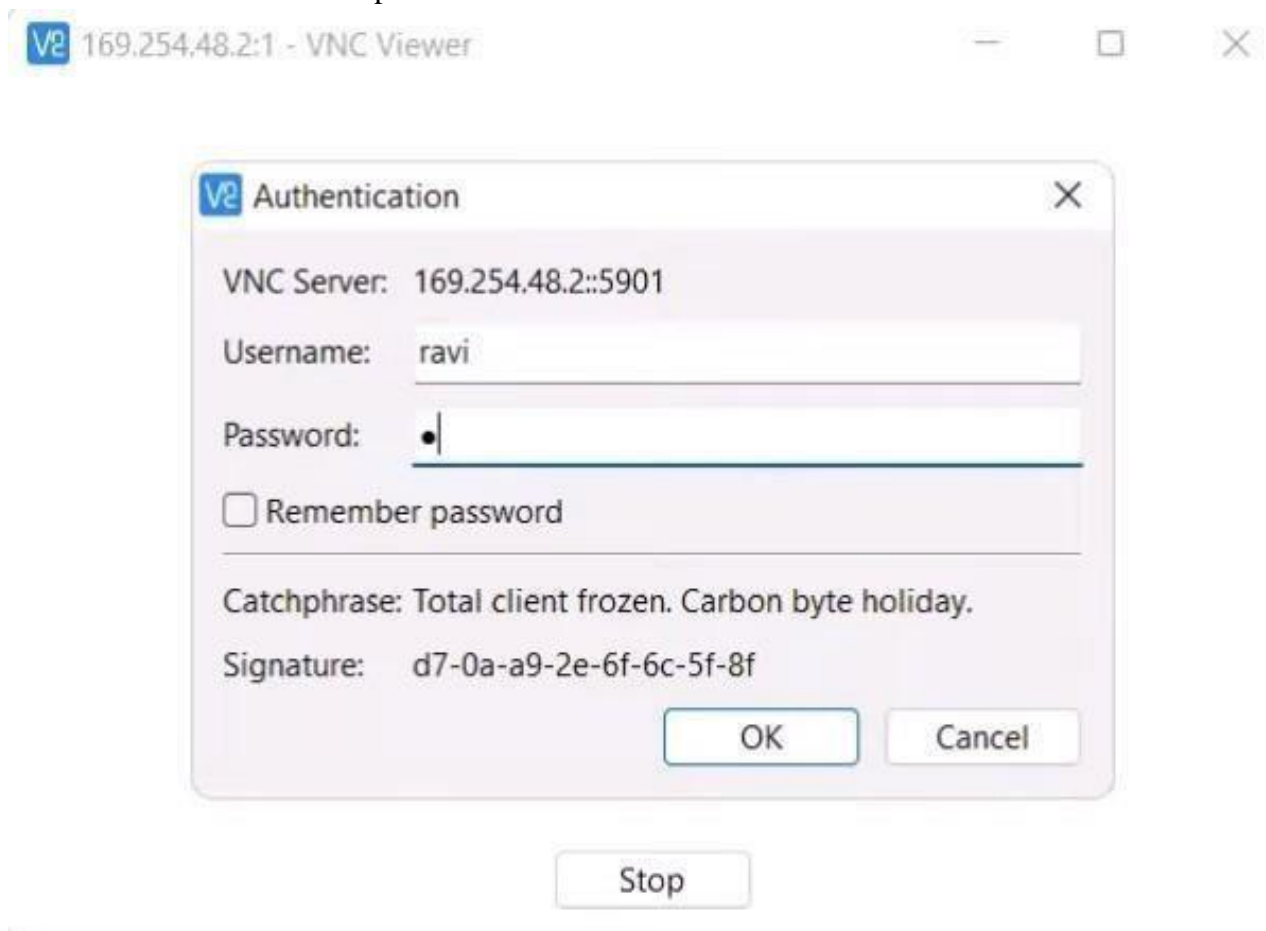
13. Type VNC server and press enter



14. Copy the IP address displayed and paste it into the VNC server ,click on continue

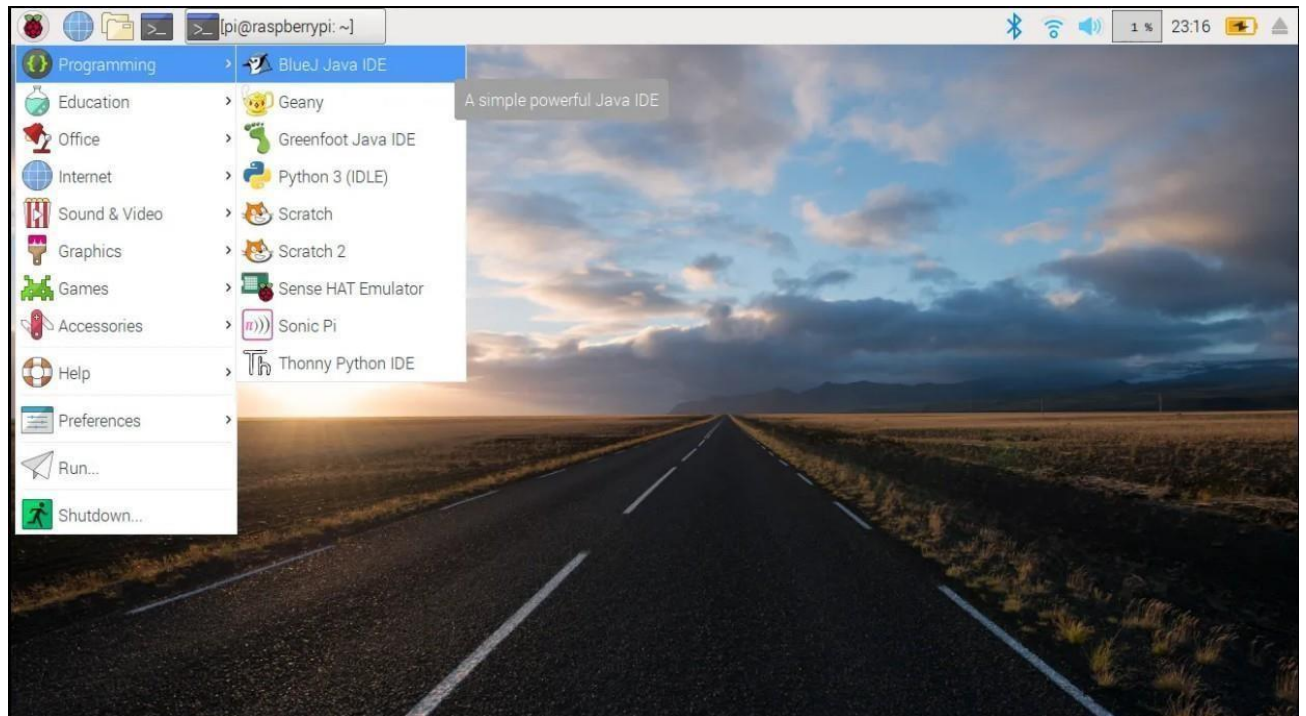


15. Give username and password .



16. Now it will be directly redirected to Raspberry Pi OS .Finally the Osis installed in the Raspberry Pi .Now it is ready to use the Raspberry Pi

Output



Result

Thus, the Raspberry Pi is configured and studied successfully.

AD22511-INTERNET OF THINGS AND ITS APPLICATIONS LABORATORY

EX.NO: 9

DATE:

Implement MQTT protocol using Arduino

Aim:

To establish MQTT communication within Raspberry Pi, enabling message publishing and subscription between the devices for IoT applications.

Description of the MQTT Protocol:

MQTT (Message Queuing Telemetry Transport) is a lightweight, publish-subscribe messaging protocol widely used in IoT. It enables devices with constrained resources to communicate over networks with minimal bandwidth. The protocol operates on a client-server model where devices (clients) connect to a central broker to publish or subscribe to message topics. The MQTT broker manages and routes messages, ensuring efficient data exchange in IoT applications.

Materials Required:

- Raspberry Pi with MQTT broker (Mosquitto) installed
- Internet connection
- Arduino IDE with necessary libraries (e.g., PubSubClient)

Procedure:

STEP 1: Set Up MQTT Broker on Raspberry Pi

1. Open the Raspberry Pi terminal.
2. Install Mosquitto MQTT broker by giving the following commands.

```
sudo apt update  
  
sudo apt install mosquitto mosquitto-clients
```

STEP 2: Publish and Subscribe with MQTT Commands on Raspberry Pi

3. Open two terminal windows on Raspberry Pi.
4. In the first terminal window, subscribe to a topic

SYNTAX: mosquitto_sub -t <ANY NAME>

Query : mosquitto_sub -t test

In the second terminal window, publish a message to the same topic:

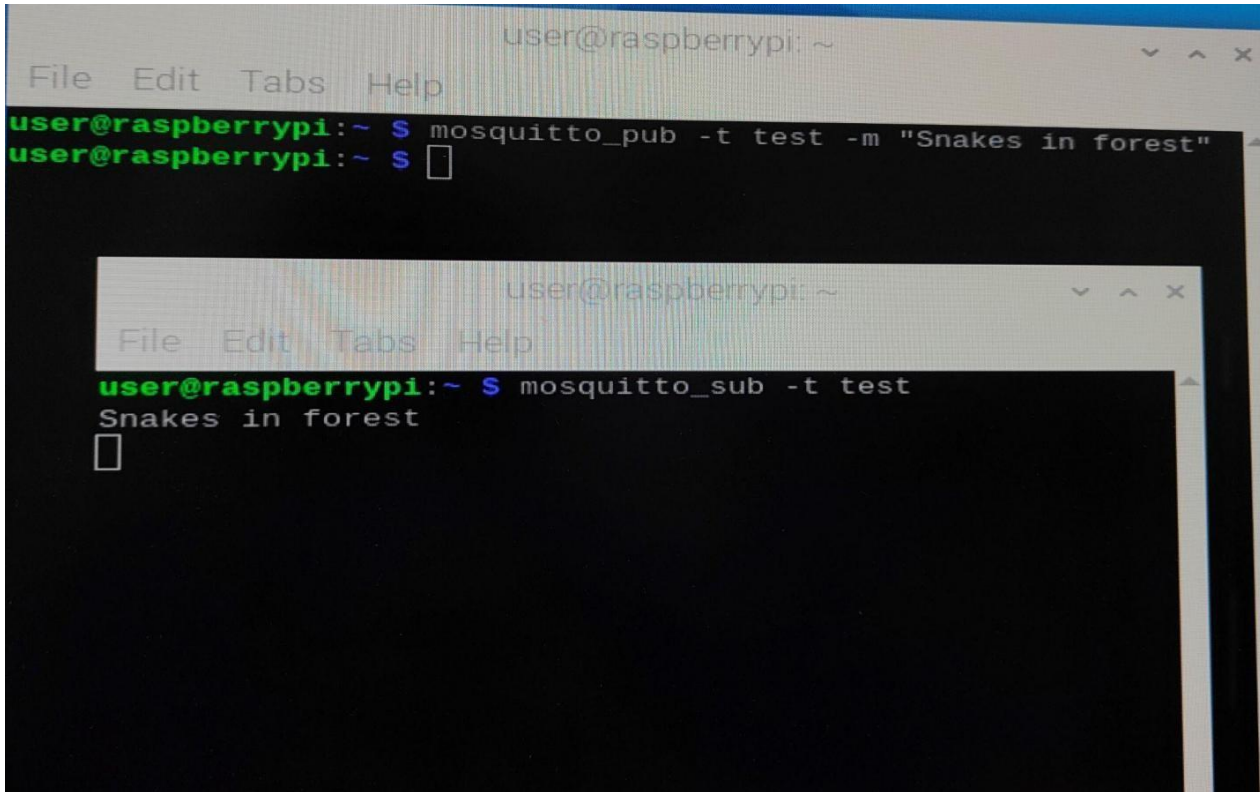
Syntax: `mosquitto_pub -t <any name> -m "any message"`

Query : `mosquitto_pub -t test -m "Hello to Raspberry Pi"`

5. Verify that the message appears in the subscription window.

6. Displays "Hello to Raspberry Pi!" in the subscription terminal.

Output :



```
user@raspberrypi: ~  
File Edit Tabs Help  
user@raspberrypi:~$ mosquitto_pub -t test -m "Snakes in forest"  
user@raspberrypi:~$  
  
user@raspberrypi: ~  
File Edit Tabs Help  
user@raspberrypi:~$ mosquitto_sub -t test  
Snakes in forest  
user@raspberrypi:~$
```

Result :

The experiment successfully established an MQTT communication channel between Raspberry Pi terminals. This setup allows for efficient message exchange between devices, illustrating the practicality of MQTT in IoT applications.

AD22511-INTERNET OF THINGS AND ITS APPLICATIONS LABORATORY

EX.NO: 10

DATE:

Study and Implement Zigbee Protocol using Arduino / Raspberry Pi

Aim:

To send temperature data between two Arduino boards using Zigbee modules, you'll need to connect a temperature sensor (like an LM35 or DHT11) to the transmitter Arduino and send this data to the receiver Arduino via Zigbee. Here's a step-by-step guide, including the code for both the transmitter and receiver

INTRODUCTION TO ZIGBEE PROTOCOL

- **Overview:** Zigbee is a wireless communication protocol based on the IEEE 802.15.4 standard, specifically designed for applications requiring secure, reliable, low-power communication in personal area networks. It supports mesh networking, allowing devices to form a network where data can be relayed through intermediate nodes.
- **Key Characteristics:**
 - **Low Power Consumption:** Designed to support battery-powered devices with extended operation.
 - **Mesh Networking:** Allows multiple devices to connect in a flexible network where data can be routed through multiple paths, improving reliability and coverage.
 - **Device Types:** Zigbee networks consist of three types of devices:
 - **Coordinator:** The primary controller of the network that initiates and manages devices.
 - **Router:** Extends the range by routing data between devices.
 - **End Device:** Communicates only with the coordinator or router; often battery-powered with limited functionality.
 - **Data Rate:** Zigbee typically operates with data rates of up to 250 kbps, which is suitable for low-data applications like sensor readings.
- **Applications:** Zigbee is used widely in home automation, industrial monitoring, medical data collection, smart metering, and security systems.

Components Required

- 2 Arduino boards (e.g., Arduino Uno)
- 2 XBee modules (configured as Coordinator and Router)
- 2 XBee shields (or adapters)
- 1 Temperature sensor (e.g., LM35 or DHT11)
- Jumper wires

Step 1: Configure XBee Modules

1. **Set up Coordinator and Router** using XCTU (Digi's configuration software):
 - Assign the same PAN ID for both modules (e.g., 1234).
 - Set one module as **Coordinator** and the other as **Router**.
 - Set **DL** (Destination Low) of the Coordinator to the Router's address, and the **DL** of the Router to the Coordinator's address.

Step 2: Connect the Hardware

- **Transmitter Setup (Coordinator):**
 - Connect the temperature sensor to the Arduino. For example, if using **LM35**:
 - **VCC** of LM35 to **5V** on Arduino
 - **GND** of LM35 to **GND** on Arduino
 - **OUT** of LM35 to **A0** on Arduino
- **Receiver Setup (Router):**
 - Connect the XBee module to the Arduino board using the shield or jumper wires as described.

Step 3: Arduino Code

Code for Transmitter (Coordinator)

The transmitter Arduino reads the temperature data from the sensor, formats it, and sends it via the XBee module.

```
#include <SoftwareSerial.h>

SoftwareSerial xbee(2, 3);

const int tempPin = A0;

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

void loop() {
  int rawValue = analogRead(tempPin);

  float temperature = (rawValue / 1024.0) * 500.0;

  xbee.print("Temperature: ");
  xbee.print(temperature);
  xbee.println(" C");

  Serial.print("Sent: Temperature: ");
```

```
Serial.print(temperature);  
  
Serial.println(" C");  
  
delay(2000); // Send data every 2 seconds  
  
}
```

Code for Receiver (Router)

The receiver Arduino reads incoming data from the XBee module and prints it to the Serial Monitor.

```
#include <SoftwareSerial.h>  
  
SoftwareSerial xbee(2, 3); // RX, TX pins for XBee  
  
void setup() {  
  
    Serial.begin(9600);  
  
    xbee.begin(9600);  
  
}  
  
void loop() {  
  
    if (xbee.available()) {  
  
        String receivedData = xbee.readString();  
  
        Serial.println("Received: " + receivedData);  
  
    }  
  
}
```

Step 4: Upload and Test

1. Upload Code:

- Upload the **transmitter code** to the Arduino connected to the **Coordinator XBee** module.
- Upload the **receiver code** to the Arduino connected to the **Router XBee** module.

2. Open Serial Monitor:

- Open the Serial Monitor on both Arduino boards and set the baud rate to 9600.
- On the transmitter's Serial Monitor, you should see temperature data being sent.
- On the receiver's Serial Monitor, you should see the temperature data received from the transmitter.

OUTPUT :

31.28a		31.28a	
31.77a		Chara recieved	
31.77a		31.28a	
31.28a	Sending Data	Chara recieved	Recieving Data
31.28a		31.77a	
31.77a		Chara recieved	
31.28a		31.28a	
31.28a		Chara recieved	
31.28a		31.77a	
31.28a		Chara recieved	
31.28a		31.28a	
		Chara recieved	
		31.28a	

Result

The implementation of the Zigbee protocol using Arduino was successful. Data was transmitted wirelessly from the Coordinator (transmitter) Arduino to the Router (receiver) Arduino using Zigbee modules.

AD22511-INTERNET OF THINGS AND ITS APPLICATIONS LABORATORY

EX.NO: 11

DATE:

Develop Video Surveillance application using IoT

Aim:

To capture and display live video feed from the Raspberry Pi camera using OpenCV, resize each frame, convert it to grayscale, and display it in real time.

Procedure on Raspberry Pi

1. Install OpenCV and Dependencies:

- Ensure OpenCV is installed on your Raspberry Pi. You can install it using the following commands:

```
sudo apt update
```

```
sudo apt install python3-opencv
```

2. Set Up the Camera:

- Connect a camera module to your Raspberry Pi's camera port or use a USB camera.
- Enable the camera interface by opening the Raspberry Pi configuration

```
sudo raspi-config
```

- Navigate to **Interfacing Options > Camera > Enable**. Reboot the Raspberry Pi if prompted.

3. Python Code to Capture Video with OpenCV:

- Use the provided code to access the camera feed, resize the frames, and convert them to grayscale.

```
import cv2
import numpy as np
cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    if not ret:
        break
    frame = cv2.resize(frame, (640, 480))
```

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
cv2.imshow('Live Camera Feed', frame)
if cv2.waitKey(1) == ord('q'):
    break
cap.release()
cv2.destroyAllWindows()
```

4. Running the Code:

- Save the code (e.g., as `video_capture.py`) and execute it
- `python3 video_capture.py`

5. Stop the Program:

- Press 'q' in the display window to exit the live feed.

When run on the Raspberry Pi, this code will:

- Display a real-time video feed from the camera in a 640x480 resolution window.
- Show each frame in grayscale, and close the window when 'q' is pressed.

OUTPUT :



Result

The program successfully captured and displayed a real-time video feed from the Raspberry Pi camera using OpenCV. Each frame was resized to 640x480 resolution and converted to grayscale before being displayed.

**AD22501 - INTERNET OF THINGS AND ITS APPLICATIONS
LABORATORY**

EX.NO:12

DATE:

MINI PROJECT – IoT MOTION DETECTION ALERT SYSTEM

TEAM MEMBERS :

1. AKASH V
2. AVVUDAIYAPPAN RM
3. HARISH S
4. HARISH RAJ S
5. BALAKRISHNAN R

MENTORS :

1. MS. R. SAMUNDI
2. MS. P. PREMA

AIM :

To design and develop an IoT-based motion detection alert system using a PIR sensor and Arduino that detects movement in real-time and triggers visual/audio alerts while simulating cloud notifications, enhancing home security and automation in a cost-effective manner suitable for residential applications.

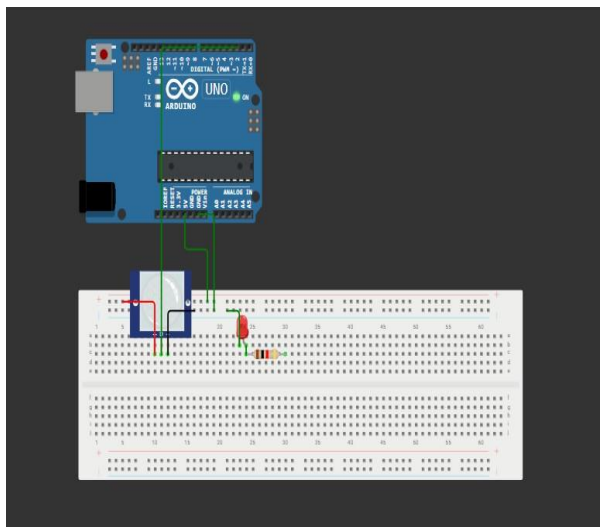
INTRODUCTION :

Home security faces challenges such as unauthorized intrusions, delayed responses to threats, and the need for constant monitoring without human intervention. Traditional alarm systems often rely on fixed setups that may miss subtle movements or fail to notify remotely. To address these, an IoT motion detection alert system is proposed, utilizing a PIR (Passive Infrared) sensor with an Arduino microcontroller to detect motion, activate local alerts, and simulate data transmission to the cloud for remote access. This approach improves response times, energy efficiency, and overall safety for smart homes.

RELATED WORK :

1. IoT-based home security systems integrate AI with devices like motion sensors and cameras to enhance detection accuracy and automate responses, aligning with the project's focus on real-time motion alerts.
2. The use of smart sensors in IoT home security enables remote monitoring and threat protection through interconnected devices, similar to the proposed system's PIR-based intrusion detection.
3. Modern home security trends emphasize automation via motion sensors to trigger lights, locks, and alarms instantly, reflecting the adaptive alert mechanism in this mini project.
4. Research on IoT motion detection using low-cost PIR sensors with Raspberry Pi demonstrates effective remote monitoring for home security, supporting the Arduino-based prototype's cloud simulation.

PROPOSED WORK :



```
IoT Motion Detection System Initialized...  
Motion detected!  
Uploading motion status to IoT cloud...  
Motion ended!
```


PROTOTYPE

1. The PIR sensor continuously monitors for infrared changes indicating motion
2. When motion is detected in low-activity areas (such as nighttime or unoccupied spaces), the Arduino activates the LED alert and simulates cloud upload.
3. If no motion persists, the system resets the alert to conserve energy and avoid false alarms.
4. The relay module (optional extension) serves as a switching interface to control external sirens or lights based on detection signals.
5. This automation provides timely, remote-accessible security alerts, ideal for affordable home protection and integration with smart ecosystems.

MODULES :

1. **Arduino UNO**
Serves as the main microcontroller that processes sensor inputs and triggers alerts accordingly.
2. **PIR (Passive Infrared) Sensor**
Detects motion through infrared radiation changes, allowing the Arduino to identify potential intrusions.
3. **Relay Module**
Works as an electronic switch to control external alert devices like buzzers based on Arduino signals.
4. **LED Headlight Module**
Acts as the visual indicator for motion detection, simulating headlight or status lights in a security setup
5. **Jumper Wires**
Used to establish connections between Arduino, relay, PIR, and LED components.
6. **Chassis**
Used as the base to mount all components for the prototype demonstration.
7. **Power Supply (Battery Pack / USB)**
Provides electrical power to the Arduino and other components in the system.

PROGRAM :

```
int pirPin = 2;
int ledPin = 13;
int pirState = LOW;
int val = 0;

void setup() {
  pinMode(pirPin, INPUT);
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
  Serial.println("IoT Motion Detection System Initialized...");
}

void loop() {
  val = digitalRead(pirPin);

  if (val == HIGH) {
    digitalWrite(ledPin, HIGH);
    if (pirState == LOW) {
      Serial.println("Motion detected!");
      pirState = HIGH;

      // Simulate sending data to cloud
      Serial.println("Uploading motion status to IoT cloud...");
      delay(500);
    }
  } else {
    digitalWrite(ledPin, LOW);
    if (pirState == HIGH) {
      Serial.println("Motion ended!");
      pirState = LOW;
    }
  }
}
```

RESULT :

The prototype successfully demonstrated real-time motion detection and alert triggering based on PIR input. The LED activated on movement, with serial logs simulating cloud notifications. This validated the system's effectiveness in providing instant security responses, remote monitoring potential, and energy-efficient operation.

CONCLUSION :

The developed IoT Motion Detection Alert System offers an intelligent, efficient, and cost-effective solution for enhancing home security. By leveraging simple components like an Arduino and PIR sensor, the system dynamically responds to intrusions with local and simulated remote alerts without manual oversight. This innovation can be extended for real-world applications with advanced integrations (e.g., camera feeds or mobile apps) for comprehensive smart security.

REFERENCE :

1. <https://www.intuz.com/blog/iot-smart-home-security-benefits-use-cases-and-top-devices>
2. <https://ieeexplore.ieee.org/document/9214684/>
3. <https://www.semanticscholar.org/paper/Motion-Detection-with-IoT-Based-Home-Security-Tiong-Ahmad/6d0456ec1837e6b0f3b0b5707214665e47a4f8c3>