NEHRU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(AUTONOMOUS)

An ISO 9001:2015 and ISO 14001:2015 Certified Institution
Affiliated to Anna University, Chennai, Approved by AICTE, New Delhi &
Recognized by UGC with 2(f) & 12(B), Re-accredited by NAAC with "A+"
NBA Accredited UG Courses: AERO | CSE | ECE | EEE | MECH | MCT
Nehru Gardens, T.M.Palayam, Coimbatore-641 105.

DEPARTMENT OF MECHATRONICS ENGINEERING

LABORATORY RECORD

OCS352 IOT CONCEPTS AND APPLICATIONS LABORATORY

NEHRU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(AUTONOMOUS)

An ISO 9001:2015 and ISO 14001:2015 Certified Institution
Affiliated to Anna University, Chennai, Approved by AICTE, New Delhi &
Recognized by UGC with 2(f) & 12(B), Re-accredited by NAAC with "A+"
NBA Accredited UG Courses: AERO | CSE | ECE | EEE | MECH | MCT
Nehru Gardens, T.M.Palayam, Coimbatore-641 105.

DEPARTMENT OF MECHATRONICS ENGINEERING

OCS352 - IOT CONCEPTS AND APPLICATIONS LABORATORY

NAME :

REGNO :

DEGREE :

YEAR / SEMESTER :

DEPARTMENT OF MECHATRONICS ENGINEERING

VISION AND MISSION OF THE INSTITUTION

VISION

Our Vision is to mould the youngsters to acquire sound knowledge in technical and scientific fields to face the future challenges by continuous upgradation of all resources and processes for the benefit of humanity as envisaged by our great leader Pandit Jawaharlal Nehru.

MISSION

- To build a strong centre of learning and research in engineering and technology.
- To facilitate the youth to learn and imbibe discipline, culture and spirituality.
- To produce quality engineers, dedicated scientists and leaders.
- To encourage entrepreneurship.
- To face the challenging needs of the global industries.

VISION AND MISSION OF THE DEPARTMENT

VISION

Our Vision is to strive the students to foster rigorous academic emphasis with rich diversity of skills for the ability and passion to work sensibly and ethically for the betterment of humankind

MISSION

- To prepare excellent Mechatronics Engineers with leading edge technology
- To achieve blending of knowledge attainment and application
- To achieve blending of knowledge attainment and application
- To develop the future engineers with invaluable entrepreneurial skill
- To build a strong integrated team of Mechatronics professionals

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

- Application of mathematical modeling, scientific and automation concepts to formulate problems in mechatronics systems and provide solutions employing modern tools
- Professional practice driven by value based education committed to ethical principles, environmental concerns and social issues.
- Ability to work in a team as a mber/leader possessing technical and organizational capabilities to manage/initiate an enterprise

PROGRAM OUTCOMES (PO)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to providevalid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineeringand IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions insocietal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development. **PO8: Ethics:** Apply ethical principles and commit to professional ethics, responsibilities, and norms of theengineering practice.

PO9: Individual and Teamwork: Function effectively as an individual, and as a member or leader in diverseteams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

● To understand the concepts of engineering fundamentals, design and problem analysts to arrive multiple solutions for the complex problems using classical methods and modern IT tools.

● To provide an opportunity to identify the responsibilities of social engineering practices by knowing the ethical and environmental values for the sustainable development.

● To persist with life-long learning and effective communication to lead a team to promote managerial skills and entrepreneurship in multidisciplinary environment.

NEHRU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(AUTONOMOUS)

An ISO 9001:2015 and ISO 14001:2015 Certified Institution
Affiliated to Anna University, Chennai, Approved by AICTE, New Delhi &
Recognized by UGC with 2(f) & 12(B), Re-accredited by NAAC with "A+"
NBA Accredited UG Courses: AERO | CSE | ECE | EEE | MECH | MCT
Nehru Gardens, T.M.Palayam, Coimbatore-641 105.

BONAFIDE CERTIFICATE

This is to certify that this is the bonafide record of bearing Register no………………………………….. in the **OCS352 - IOT CONCEPTS AND APPLICATIONS LABORATORY** for the partial fulfillment of **B.E.** in the branch of **MECHATRONICS ENGINEERING** for **III Year /VI semester** during the academic year **2024-2025**.

HEAD OF THEDEPARTMENT **FACULTYIN-CHARGE**

DATE: DATE:

This record is submitted to the Anna University Practical Examination held on………………… at **NEHRU INSTITUTE OF ENGINEEERING AND TECHNOLOGY,** COIMBATORE.

INTERNALEXAMINER **EXTERNALEXAMINER**

DATE: DATE:

LIST OF EXPERIMENT

S. No.	Experiment
01	Introduction to Arduino platform and programming I
02	Introduction to Arduino platform and programming II
03	Interfacing Arduino to Zigbee module
04	Interfacing Arduino to Bluetooth Module
05	Interfacing Arduino to GSM module
06	Introduction to Raspberry PI platform and python programming
07	Interfacing sensors to Raspberry PI
08	Communicate between Arduino and Raspberry PI using any wireless medium
09	Setup a cloud platform to log the data
10	Log Data using Raspberry PI and upload to the cloud platform

INDEX

EX.NO	DATE	NAME OF THE EXPERIMENT	MARKS	STAFF SIGN
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				

Experiment 1	**Introduction to Arduino platform and programming - I**

Aim:

To test Arduino IDE using Arduino platform and programming.

Apparatus Required:

1. Arduino board (e.g., Arduino Uno)
2. Ultrasonic sensor
3. Breadboard and jumper wires
4. Downloading cable

Procedure:

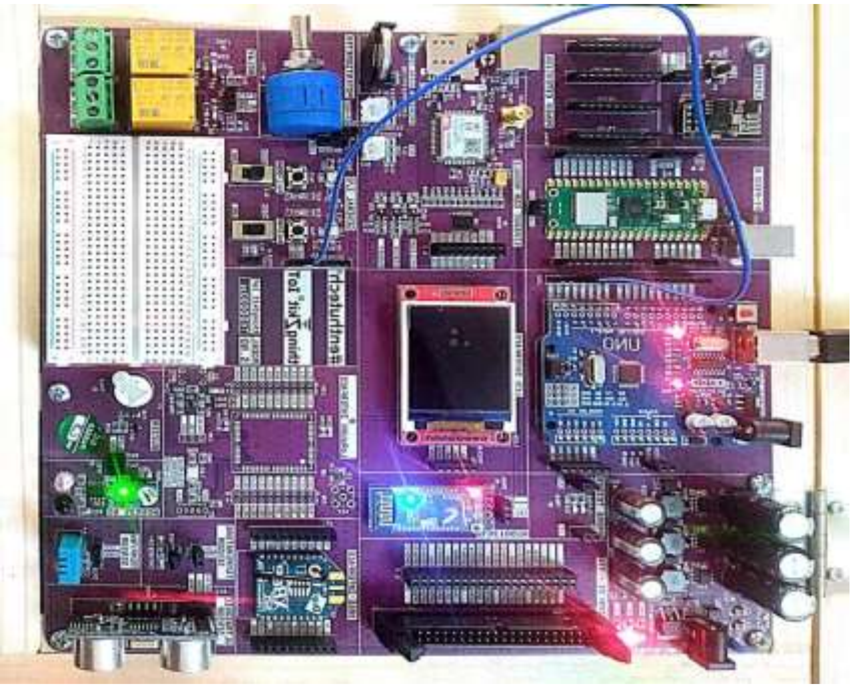
1. Write the code using the Arduino programming language. The code consists of two main functions: setup() (for initialization) and loop() (for the main program execution).
2. There are various Arduino boards with different specifications and features, but they all share a common architecture.
3. Connect your Arduino board to your computer using a USB cable. Select the correct board and port in the Arduino IDE. Click the "Upload" button to upload the code to the Arduino.

Program:

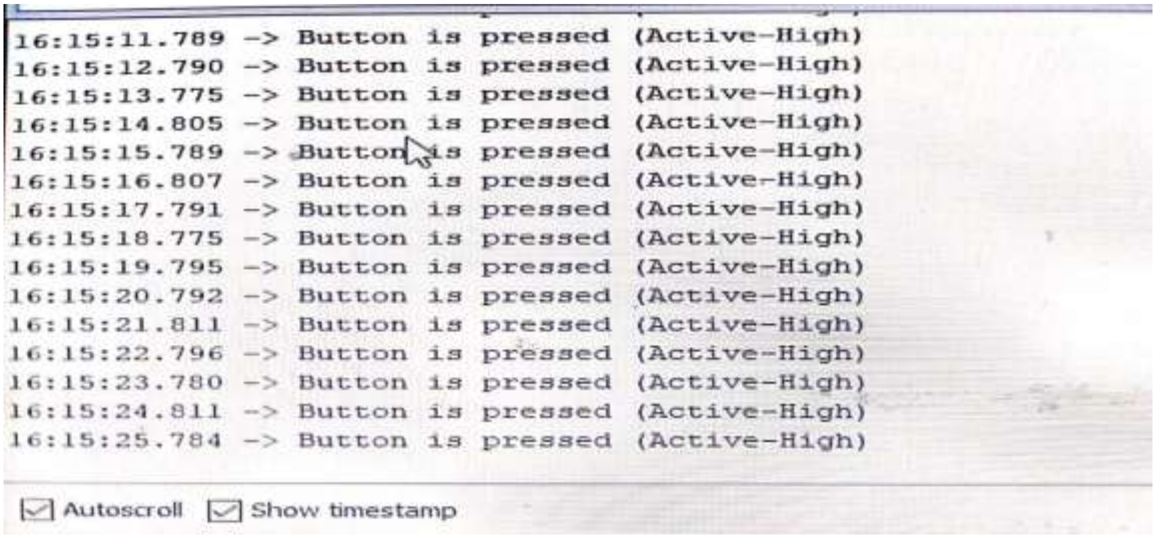
PUSHBUTTON USING ARDUINO

```
#include <Arduino.h>
#define buttonPin  2

void setup()
{
   pinMode(buttonPin, INPUT);                    // Initialize the button pin as an input
   Serial.begin(9600);                                // Initialize serial communication
   Serial.println("Active-High Push Button Status:");
}

void loop()
{
   bool buttonState = digitalRead(buttonPin);          /*Read the status of the button*/
   if (buttonState == HIGH || buttonState == LOW)  // Check for valid button state
  {
     if (buttonState == LOW)
     {
     Serial.println("Button is pressed (Active-High)");        // Print the button status
     }
     else
     {
     Serial.println("Button is not pressed");
     }
   }
  else
  {
   Serial.println("Error: Invalid button state");             // Handle invalid button state
  }
  delay(1000);
}
```

Connection:



OUTPUT:



```
16:15:11.789 -> Button is pressed (Active-High)
16:15:12.790 -> Button is pressed (Active-High)
16:15:13.775 -> Button is pressed (Active-High)
16:15:14.805 -> Button is pressed (Active-High)
16:15:15.789 -> Button is pressed (Active-High)
16:15:16.807 -> Button is pressed (Active-High)
16:15:17.791 -> Button is pressed (Active-High)
16:15:18.775 -> Button is pressed (Active-High)
16:15:19.795 -> Button is pressed (Active-High)
16:15:20.792 -> Button is pressed (Active-High)
16:15:21.811 -> Button is pressed (Active-High)
16:15:22.796 -> Button is pressed (Active-High)
16:15:23.780 -> Button is pressed (Active-High)
16:15:24.811 -> Button is pressed (Active-High)
16:15:25.784 -> Button is pressed (Active-High)
```

RESULT:

The test was successfully completed with ultrasonic sensor

Experiment 2	**Introduction to Arduino platform and programming - II**

Aim:

To test Arduino IDE using Arduino platform and programming.

Apparatus Required:

1. Arduino board (e.g., Arduino Uno)
2. Ultrasonic sensor
3. Breadboard and jumper wires
4. Downloading cable

Procedure:

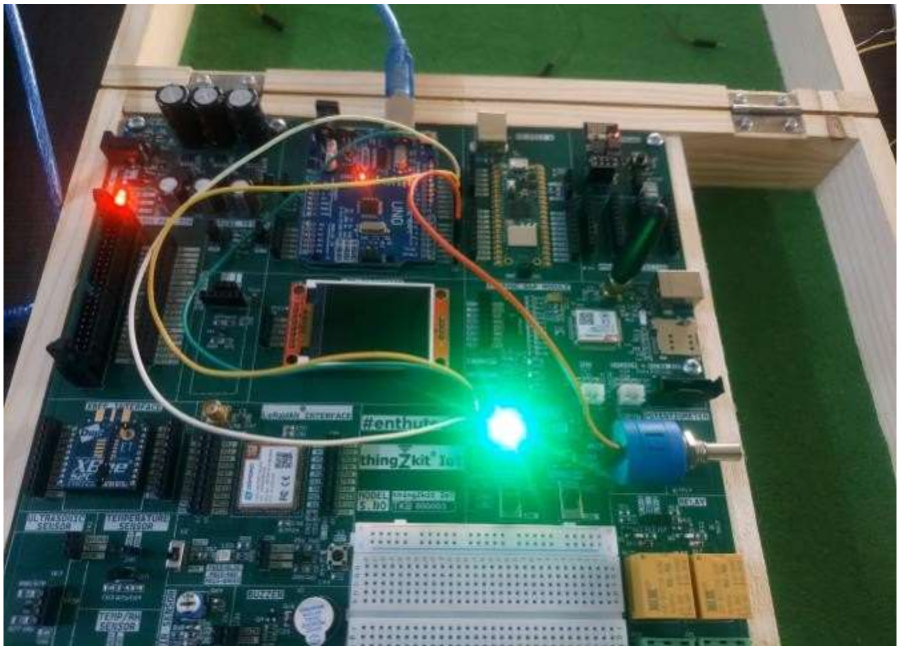
1. Write the code using the Arduino programming language. The code consists of two main functions: setup() (for initialization) and loop() (for the main program execution).
2. There are various Arduino boards with different specifications and features, but they all share a common architecture.
3. Connect your Arduino board to your computer using a USB cable. Select the correct board and port in the Arduino IDE. Click the "Upload" button to upload the code to the Arduino.

Program:

RGB LED using Arduino

```
#include <Arduino.h>

#define Red_Pin 9                                  /*RGB LED pins (Common Cathode)*/
#define Green_Pin 10
#define Blue_Pin 11
#define COLOR_CHANGE_DELAY 1000                     /*Delay  (in milliseconds)*/

void setup()
{
 pinMode(Red_Pin, OUTPUT);                  /*Initialize the RGB LED pins as outputs*/
 pinMode(Green_Pin, OUTPUT);
 pinMode(Blue_Pin, OUTPUT);
}

void loop()
{
 digitalWrite(Red_Pin,  HIGH);               /*Turn on Red, turn off Green and Blue*/
 digitalWrite(Green_Pin, LOW);
 digitalWrite(Blue_Pin, LOW);
 delay(COLOR_CHANGE_DELAY);

 digitalWrite(Red_Pin,  LOW);             /*Turn off Red, turn on Green, turn off Blue*/
 digitalWrite(Green_Pin, HIGH);
 digitalWrite(Blue_Pin, LOW);
 delay(COLOR_CHANGE_DELAY);

 digitalWrite(Red_Pin, LOW);              /*Turn off Red, turn off Green, turn on Blue*/
 digitalWrite(Green_Pin, LOW);
 digitalWrite(Blue_Pin, HIGH);
 delay(COLOR_CHANGE_DELAY);

 digitalWrite(Red_Pin,  HIGH);                         /*Turn ON Red,Green,Blue*/
 digitalWrite(Green_Pin, HIGH);
 digitalWrite(Blue_Pin, HIGH);
 delay(COLOR_CHANGE_DELAY);
}
```

Connection:



RESULT:

The test was successfully completed with ultrasonic sensor.

Experiment 3

Interfacing Arduino to Zigbee module

Aim:

To test Arduino IDE using Zigbee.

Apparatus Required:

1. Arduino board (e.g., Arduino Uno)
2. Breadboard and jumper wires
3. Zigbee

Procedure:

1. Connect your Arduino board to your computer using a USB cable. Select the correct board and port in the Arduino IDE. Click the "Upload" button to upload the code to the Arduino.
2. Test the Zigbee communication between devices, ensuring a stable connection and data transfer. Zigbee is suitable for short-range applications like wearables, smart home devices, and local sensor networks.
3. Consider the power requirements of each communication method, especially for battery-operated IoT devices.

Program:

Zigbee

```
#include <SoftwareSerial.h>
SoftwareSerial xbee(2, 3); // RX, TX void
setup()
{
 Serial.begin(9600);
 xbee.begin(9600);                    // Set the baud rate to match your XBee module
}
void loop()
{
 if (Serial.available())
{
  String dataToSend = Serial.readString();
  xbee.println(dataToSend);                              // Send the data to XBee
 }

 if (xbee.available())
 {
  String receivedData = "";
  while (xbee.available())
 {
   char c = xbee.read();
   receivedData += c;
  }
  Serial.println("Received: " + receivedData);
 }
}
```

RESULT:

The test was successfully completed.

Experiment 4	**Interfacing Arduino to Bluetooth module**

Aim:

To test Arduino IDE using Bluetooth.

Apparatus Required:

1. Arduino board (e.g., Arduino Uno)
2. Breadboard and jumper wires
3. Bluetooth

Procedure:

1. Connect your Arduino board to your computer using a USB cable. Select the correct board and port in the Arduino IDE. Click the "Upload" button to upload the code to the Arduino.
2. Test the Bluetooth communication between devices, ensuring a stable connection and data transfer. Bluetooth is suitable for short-range applications like wearables, smart home devices, and local sensor networks.
3. Consider the power requirements of each communication method, especially for battery-operated IoT devices.

Program:

Bluetooth

```
#include <SoftwareSerial.h>
#define LED_pin 13
SoftwareSerial bluetooth(2, 3);                                    // RX, TX
float data=25.98;
char charArray[10];
void setup()
{
 Serial.begin(9600);                             // Initialize the serial monitor
 bluetooth.begin(9600);                    // Initialize the Bluetooth communication
 pinMode(LED_pin,OUTPUT);
}
void loop()
{
  if (bluetooth.available())          // Check if data is available from Bluetooth module
{
  char receivedChar = bluetooth.read();                 // Read the character received
via
  Serial.print("Received: ");                                         Bluetooth
  Serial.println(receivedChar); // Example: Send a response back to the Bluetooth
  if (receivedChar == '1')
module
  {
   digitalWrite(LED_pin,
   HIGH);
   Serial.print("LED_ON");
  }
  else if(receivedChar == '0')
  {
   digitalWrite(LED_pin, LOW);
   Serial.print("LED_OFF");
  }
  else if(receivedChar == '2')
  {
   dtostrf(data, 6, 2, charArray);                        // float to char conversion
   bluetooth.write(charArray);
   delay(100);
  }
 }
}
```

RESULT:

The test was successfully completed.

Experiment 5	**Interfacing Arduino to GSM module**

Aim:

To test Arduino IDE using GSM.

Apparatus Required:

1. Arduino board (e.g., Arduino Uno)
2. Breadboard and jumper wires
3. GSM

Procedure:

1. Connect your Arduino board to your computer using a USB cable. Select the correct board and port in the Arduino IDE. Click the "Upload" button to upload the code to the Arduino.
2. Test the GSM communication between devices, ensuring a stable connection and data transfer. GSM is suitable for short-range applications like wearables, smart home devices, and local sensor networks.
3. Consider the power requirements of each communication method, especially for battery-operated IoT devices.

Program:

GSM

```
#include <SoftwareSerial.h>
#include <DFRobot_DHT11.h>
DFRobot_DHT11 DHT;
#define DHT11_PIN 8
#define DEBUG true
SoftwareSerial GSM_Serial(2, 3); // Pin 2 and 3 act as RX and TX. Connect them to TX
                                                                      and RX of ESP8266
String Host_URL = "console.thingzmate.com";                     /*user Credential*/
String Sub_URL_POST="/api/v1/device-types/thingzmate-basic- kit/devices/demo-
1/http-uplink";
String Sub_URL_GET="/api/v1/device-types/thingzmate-basic-
kit/devices/demo-1/http-downlink";
String PORT = "80";
String Bearer_Key= "Bearer 4e3dac123dbf812530fe0af613ff4aab";

int sendVal;                                                          /*Variables*/
String response = "";
bool dataStarted=0,RECV=0;
String jsonData="";

void setup()
{
 Serial.begin(9600);
 GSM_Serial.begin(9600);
 Serial.println("Thinzkit_Basic_GSM");
 GSM_Module_init();
}

void loop()
{
  Sending_data_to_the_cloud();                                          /*Uplink*/
  delay(10000);
  receiveDataFromCloud();                                             /*Downlink*/
  delay(10000);                         // Delay before making the next request
}

void Sending_data_to_the_cloud()                           /*HTTP POST Function*/
{
 RECV=0;
 GSM_Data("AT+CIPMUX=1", 1000, DEBUG);
 delay(2000);
 GSM_Data("AT+CSTT=\"airtelgprs.com\"", 1000, DEBUG);
                                                              // Connect to WiFi network
 delay(3000);
 GSM_Data("AT+CIICR", 1000, DEBUG);                  // Connect to WiFi network
 delay(3000);
 GSM_Data("AT+CIFSR", 1000, DEBUG);                  // Connect to WiFi network
 delay(3000);
 GSM_Data("AT+CIPSPRT=0", 1000, DEBUG);            // Connect to WiFi network
 delay(3000);
 Sensor_Data();
 delay(200);

 String sendData = "POST "+Sub_URL_POST+" HTTP/1.1\r\nHost: " + Host_URL
 + "\r\nAuthorization: "+Bearer_Key+"\r\nContent-Type:
 application/json\r\nContent-Length: " + String(jsonData.length()) +
 "\r\n\r\n" + jsonData;
 GSM_Data("AT+CIPSTART=0,\"TCP\",\"" + Host_URL + "\"," + PORT, 1000,
 DEBUG);
 delay(2000);
 GSM_Data("AT+CIPSEND=0," + String(sendData.length()), 1000, DEBUG);
 delay(2000);
 GSM_Serial.print(sendData);
 GSM_Serial.print((char)26);
 for(int i=0;i<10000;i++);
 GSM_Data("AT+CIPCLOSE=0", 4000, DEBUG);
}

void receiveDataFromCloud()                              /*HTTP GET Function*/
{
 RECV=1;
 String send_Data = "GET "+Sub_URL_GET+" HTTP/1.1\r\nHost:" + Host_URL +
 "\r\nAuthorization: "+Bearer_Key+"\r\n\r\n";
 GSM_Data("AT+CIPSTART=0,\"TCP\",\"" + Host_URL + "\"," + PORT, 1000,
 DEBUG);
 delay(3000);
 GSM_Data("AT+CIPSEND=0," + String(send_Data.length()), 1000, DEBUG);
 delay(2000);
 GSM_Serial.print(send_Data);
 GSM_Serial.print((char)26);
 for(int i=0;i<20000;i++);
 GSM_Data("AT+CIPCLOSE=0", 8000, DEBUG);
 delay(2000);
}
```

```
void Downlinkaction()              /*We can write our downlink action in this function*/
{
 if(response=="01"){Serial.println("LED_ON");}
 if(response=="02"){Serial.println("LED_OFF");}
}

void GSM_Module_init()
{
 GSM_Data("AT", 1000, DEBUG);                              // Reset the ESP8266 module
 delay(3000);
 GSM_Data("AT+CPIN?", 1000, DEBUG);        // Set the ESP mode as station mode
 delay(3000);
 GSM_Data("AT+CREG?", 1000, DEBUG);                   // Connect to WiFi network
 delay(3000);
 GSM_Data("AT+CGATT?", 1000, DEBUG);                // Connect to WiFi network
 delay(3000);
 GSM_Data("AT+CIPSHUT", 1000, DEBUG);               // Connect to WiFi network
 delay(3000);
 GSM_Data("AT+CIPSTATUS", 1000, DEBUG);           // Connect to WiFi network
 delay(3000);
}

String GSM_Data(String command, const int timeout, boolean debug)
{
 if (debug)
 {
 Serial.print("AT Command ==> ");
 Serial.println(command);
 }
 GSM_Serial.println(command);
 long int time = millis();
 while ((time + timeout) > millis())
 {
  while (GSM_Serial.available())
  {
  char c = GSM_Serial.read();
  if (debug)
  {
   Serial.write(c);
  }
  if (c == '{' && RECV==1)
  {
   dataStarted = true;                                          // Start capturing data
   response = "";
  }
```

```
    if (dataStarted == true && c != '{' && c != '}' )
    {
      response += c;                                        // Start capturing data
    }
    if(c == '}'&& dataStarted == true)
    {
        dataStarted = false;
        Serial.println("Received LED Status Data: " + response);
        Downlinkaction();
        response = "";
      }
    }
   }
  if (debug)
  {
    Serial.print(response);
  }
  return response;
}

void Sensor_Data()
{
  DHT.read(DHT11_PIN);
  Serial.print("temp:");
  Serial.print(DHT.temperature);
  Serial.print("  humi:");
  Serial.println(DHT.humidity);
  jsonData = "{\"Temperature\":" + String(DHT.temperature) + ",\"Humidity\":"
 + String(DHT.humidity) + "}";
}
```

RESULT:

The test was successfully completed.

Experiment 6

Introduction to Raspberry PI platform and python programming

Aim:

To interface LED sensor with a Raspberry Pi with python IDE of the Raspberry Pi and then writing a Python script to read data from the sensor.

Apparatus Required:

1. Raspberry pi board
2. Blinking LED
3. power supply
4. cables
5. Internet Connectivity

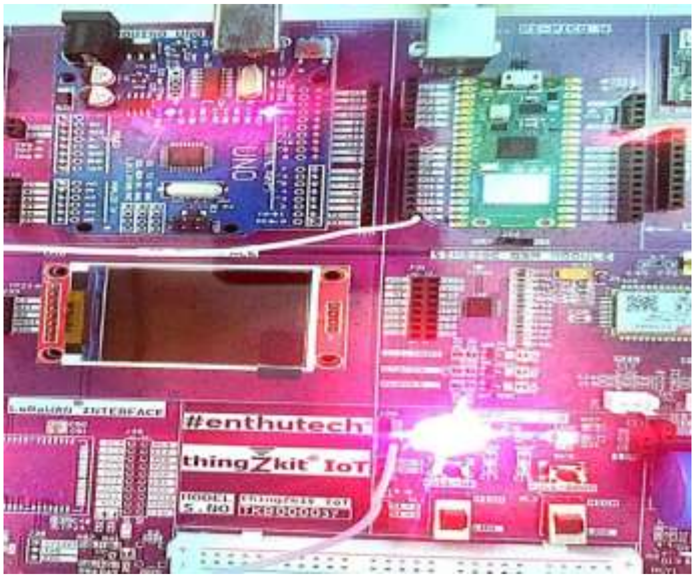
Procedure:

1. Using Thonny python IDE and create code in python format.
2. Connect the VCC pin of 5V on the Arduino.Connect the GND pin of the IR sensor to the GND on the Arduino.
3. Connect the OUT pin of the LED sensor to a digital pin on the Arduino (e.g., D2). Connect the positive (longer) leg of the LED to a digital pin through a 220-ohm resistor (e.g., D13) Connect the negative (shorter) leg of the LED to GND.

Program:

```
import machine as Gpio
import utime as TM

# Define the RGB LED pins (common cathode)
Red_Pin = Gpio.Pin(15, Gpio.Pin.OUT)

def set_rgb_color(red, green, blue):
   Red_Pin.value(red)
# Main loop while
True:
   # Red set_rgb_color(1,
   0, 0) TM.sleep(1)
   set_rgb_color(0, 0, 0)
   TM.sleep(1)
```

Connection:



RESULT:

The test was successfully completed.

Experiment 7	**Interfacing sensors to Raspberry PI**

Aim:

To interface the DHT11 sensor with a Raspberry Pi by connecting the sensor to the GPIO pins of the Raspberry Pi .

Apparatus Required:

1. Raspberry pi board
2. Blinking LED
3. power supply
4. cables
5. Internet Connectivity

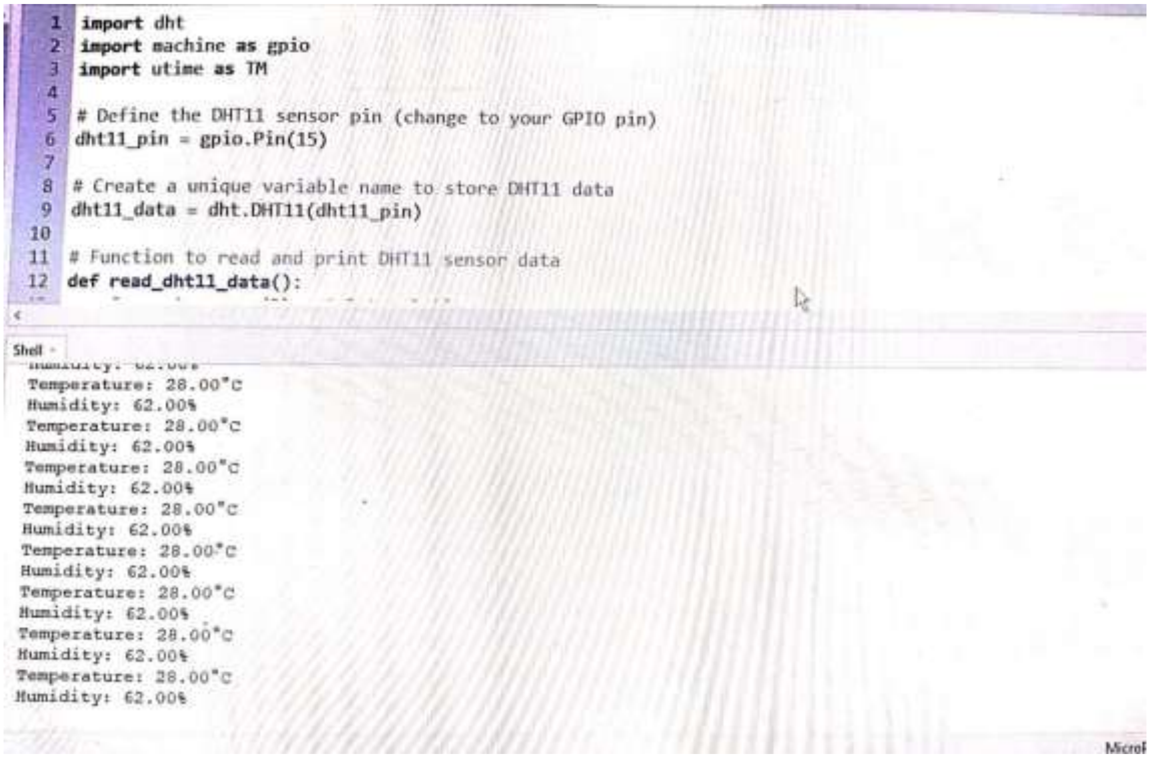
Procedure:

1. Connect the VCC pin of the DHT11 to the 5V pin on the Raspberry Pi. Connect the data pin of the DHT11 to a GPIO pin on the Raspberry Pi (e.g., GPIO 17).Connect the ground (GND) pin of the DHT11 to the GND pin on the Raspberry Pi.
2. Save the Python script with a .py extension (e.g., DHT 11_sensor.py).
3. Open a terminal and navigate to the directory containing the script.

Program:

```
import dht
import machine as gpio import
utime as TM

# Define the DHT11 sensor pin (change to your GPIO pin)
dht11_pin = gpio.Pin(15)

# Create a unique variable name to store DHT11 data dht11_data =
dht.DHT11(dht11_pin)

# Function to read and print DHT11 sensor data
def read_dht11_data():
   for _ in range(3):  # Retry 3 times try:
         dht11_data.measure()  # Measure temperature and humidity temperature =
         dht11_data.temperature()
         humidity = dht11_data.humidity() print("Temperature:
         {:.2f}Â°C".format(temperature)) print("Humidity:
         {:.2f}%".format(humidity))
         return  # If successful, exit the loop except
      OSError as e:
         print("Error reading DHT11:", e)
         TM.sleep(2)  # Add a delay before retrying

# Main loop while
True:
   read_dht11_data()
   TM.sleep(2)  # Add a delay between readings
```

Output:



```
import dht
import machine as gpio
import utime as TM

# Define the DHT11 sensor pin (change to your GPIO pin)
dht11_pin = gpio.Pin(15)

# Create a unique variable name to store DHT11 data
dht11_data = dht.DHT11(dht11_pin)

# Function to read and print DHT11 sensor data
def read_dht11_data():
```

```
Humidity: 62.00%
Temperature: 28.00°C
Humidity: 62.00%
Temperature: 28.00°C
Humidity: 62.00%
Temperature: 28.00°C
Humidity: 62.00%
Temperature: 28.00°C
Humidity: 62.00%
Temperature: 28.00°C
Humidity: 62.00%
Temperature: 28.00°C
Humidity: 62.00%
Temperature: 28.00°C
Humidity: 62.00%
Temperature: 28.00°C
Humidity: 62.00%
```

RESULT:

The test was successfully completed.

Experiment 8	Communication between Arduino and Raspberry PI using any wireless medium

Aim:

To communicate between Arduino and Raspberry PI using any wireless medium.

Apparatus Required:

1. Raspberry Pi 4
2. Arduino Uno
3. SX1278 433MHz LoRa Transmitter- Receiver Module
4. DHT11

Procedure:

1. Arduino act as Transmitter/Server and Raspberry Pi as Receiver/Client.
2. DHT 11 sensor is connected to the transmitter side which will send temperature and humidity data to the receiver side.
3. On the receiving side, Raspberry pi publish these readings on the Cayenne dashboard.
4. The DHT11 sensor is connected to the transmitting side where Arduino will get temperature and humidity values from DHT11 and then sends it to Raspberry Pi via the LoRa SX1278 module.
5. These humidity and temperature values will be uploaded to the Cayenne IoT platform which can be monitored from anywhere in the world using the internet.

Program:

```
from time import sleep from
SX127x.LoRa import *
from SX127x.board_config import BOARD import
paho.mqtt.client as mqtt

username = "20f70690-4976-11ea-84bb-8f71124cfdfb" password =
"3d7eaaf9a7c9e28626fcab4ec5a61108cfbb8be0" clientid =
"cccb41b0-4977-11ea-b73d-1be39589c6b2"

mqttc = mqtt.Client(client_id=clientid)
mqttc.username_pw_set(username, password=password)
mqttc.connect("mqtt.mydevices.com", port=1883, keepalive=60)
mqttc.loop_start()

topic_dht11_temp = "v1/" + username + "/things/" + clientid + "/data/1"
topic_dht11_humidity = "v1/" + username + "/things/" + clientid + "/data/2"

BOARD.setup()
class LoRaRcvCont(LoRa):
    def __init__(self, verbose=False):
       super(LoRaRcvCont, self).__init__(verbose)
       self.set_mode(MODE.SLEEP)
       self.set_dio_mapping([0] * 6)

   def start(self):
       self.reset_ptr_rx()
       self.set_mode(MODE.RXCONT)
       while True:
           sleep(.5)
           rssi_value   =   self.get_rssi_value()
           status   =   self.get_modem_status()
           sys.stdout.flush()

   def on_rx_done(self): print
       ("\nReceived: ")
       self.clear_irq_flags(RxDone=1)
       payload = self.read_payload(nocheck=True) #print
       (bytes(payload).decode("utf-8",'ignore')) data =
       bytes(payload).decode("utf-8",'ignore') print (data)
       temp = (data[0:4])
       humidity = (data[4:6])
```

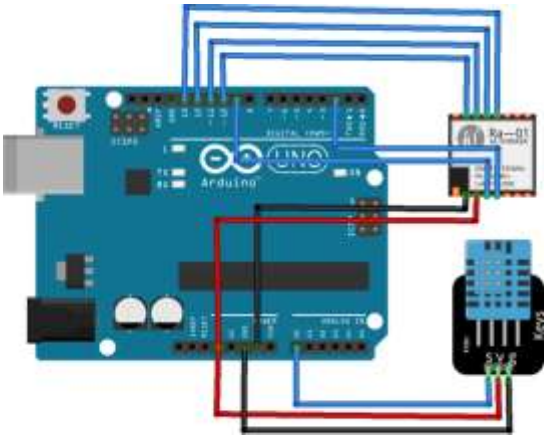
```
        print ("Temperature:")
        print (temp)
        print ("Humidity:")
        print (humidity)
        mqttc.publish(topic_dht11_temp, payload=temp, retain=True)
        mqttc.publish(topic_dht11_humidity, payload=humidity, retain=True) print
        ("Sent to Cayenne")
        self.set_mode(MODE.SLEEP)
        self.reset_ptr_rx()
        self.set_mode(MODE.RXCONT)
lora = LoRaRcvCont(verbose=False)
lora.set_mode(MODE.STDBY)
#  Medium Range  Defaults after init are 434.0MHz, Bw = 125 kHz, Cr = 4/5, Sf =
128chips/symbol, CRC on 13 dBm
lora.set_pa_config(pa_select=1)
try:
    lora.start()
except KeyboardInterrupt:
    sys.stdout.flush()
    print ("")
    sys.stderr.write("KeyboardInterrupt\n")
finally:
    sys.stdout.flush()
    print ("")
    lora.set_mode(MODE.SLEEP)
    BOARD.teardown()
```

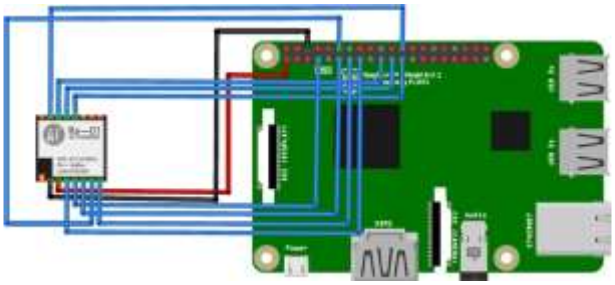
Arduino Code:

```
#include <SPI.h>
#include <RH_RF95.h>
#include "DHT.h"

#define DHTPIN A0     // what pin we're connected to
#define DHTTYPE DHT11  // DHT type
DHT dht(DHTPIN, DHTTYPE);
int hum;  //Stores humidity value
int temp; //Stores temperature value

RH_RF95 rf95;

void setup()
{
  Serial.begin(9600);
  dht.begin();
  if (!rf95.init())
```

```
   Serial.println("init failed");
 // Defaults after init are 434.0MHz, 13dBm, Bw = 125 kHz, Cr = 4/5, Sf =
128chips/symbol, CRC on
}

void loop()
{
 temp = dht.readTemperature(); hum =
 dht.readHumidity();
 String humidity = String(hum); //int to String
 String temperature = String(temp);
 String data = temperature + humidity;
 Serial.print(data);
 char d[5];
 data.toCharArray(d, 5); //String to char array
 Serial.println("Sending to rf95_server");
 rf95.send(d, sizeof(d));  rf95.waitPacketSent();
 delay(400);
}
```

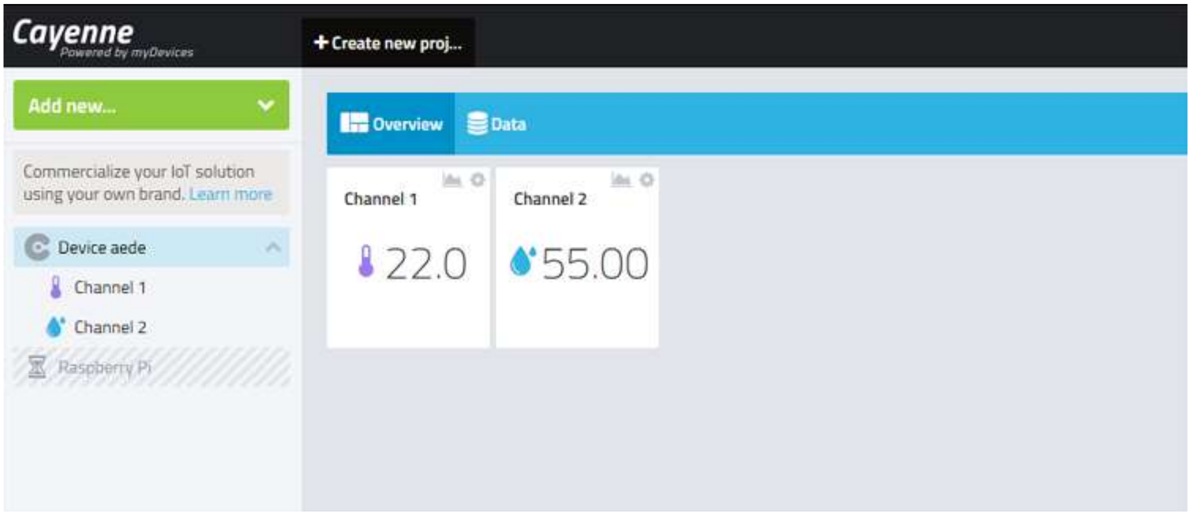
Circuit Diagram:





Transmitter Circuit Diagram **Receiver Circuit Diagram**

Output:



Commercialize your IoT solution using your own brand. Learn more

Channel 1 22.0

Channel 2 55.00

RESULT:

The test was successfully completed.

Experiment 9	**Setup a cloud platform to log the data**

Aim:

To setup a cloud platform to log the data.

Apparatus Required:

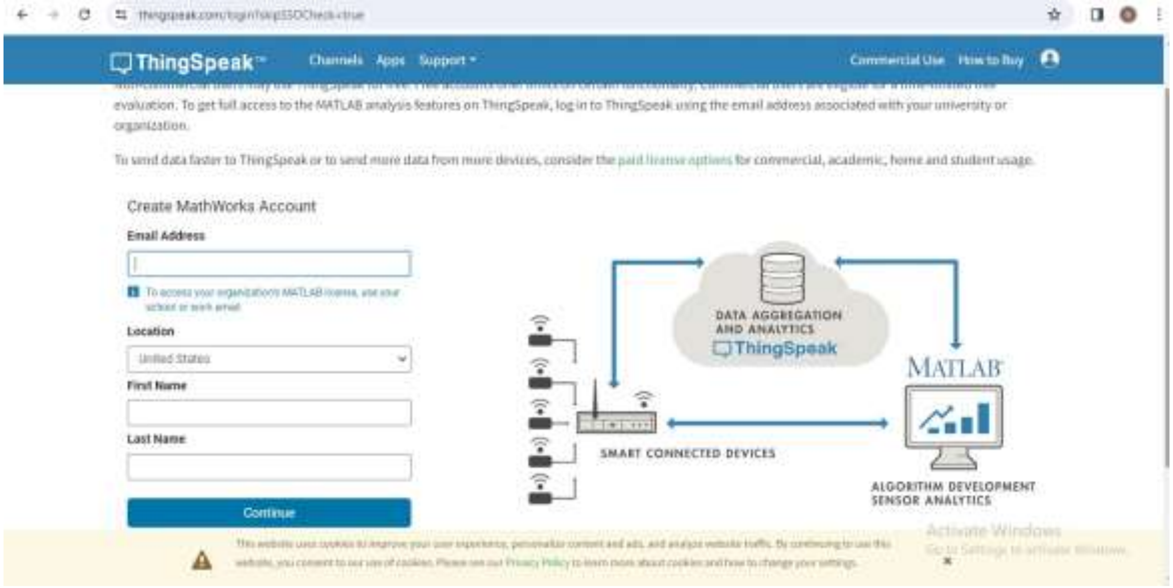
1. Arduino IDE
2. NODE MCU
3. IR Sensor
4. Bread board

Procedure:

1. Using an IR (Infrared) sensor with a platform like ThingSpeak involves connecting the IR sensor to a microcontroller, reading the sensor data, and then sending that data to ThingSpeak for visualization and analysis.
2. Install the necessary libraries for the IR sensor. For example, a common IR sensor like the TSOP382, might need the "IRremote" library.

Program:

```
#include <IRremote.h>
#include <ESP8266WiFi.h>
#include <ThingSpeak.h>

const char *ssid = "silicon systems"; const
char *password = "Silicon@2017";
const char *apiKey = "QWD5TTQ08RBGQFVH";

const int IR_PIN = 2;            // Change this to the pin connected to your IR sensor

IRrecv irrecv(IR_PIN);
decode_results results;

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

 irrecv.enableIRIn();                                        // Start the IR receiver
 WiFi.begin(ssid, password);

 ThingSpeak.begin(client);                                   // Initialize ThingSpeak
}

void loop() {
 if (irrecv.decode(&results))
{
   int irValue = results.value;               // Read and send the data to ThingSpeak
   Serial.print("IR Value: ");
   Serial.println(irValue);

                                                              // Send data to ThingSpeak
   ThingSpeak.writeField(YOUR_CHANNEL_ID, 1, irValue, apiKey);

   irrecv.resume();                                           // Receive the next value
   delay(10000);                         // Delay to avoid sending data too frequently
 }

                                                  // Other loop logic can be added here
}
```



thingspeak.com/login?skipSSOCheck=true

ThingSpeak™ Channels Apps Support ▾ Commercial Use How to Buy

Non-commercial users may use ThingSpeak for free. Free accounts offer limits on certain functionality. Commercial users are eligible for a time-limited free evaluation. To get full access to the MATLAB analysis features on ThingSpeak, log in to ThingSpeak using the email address associated with your university or organization.

To send data faster to ThingSpeak or to send more data from more devices, consider the paid license options for commercial, academic, home and student usage.

Create MathWorks Account

Email Address

To access your organization's MATLAB license, use your school or work email.

Location

United States

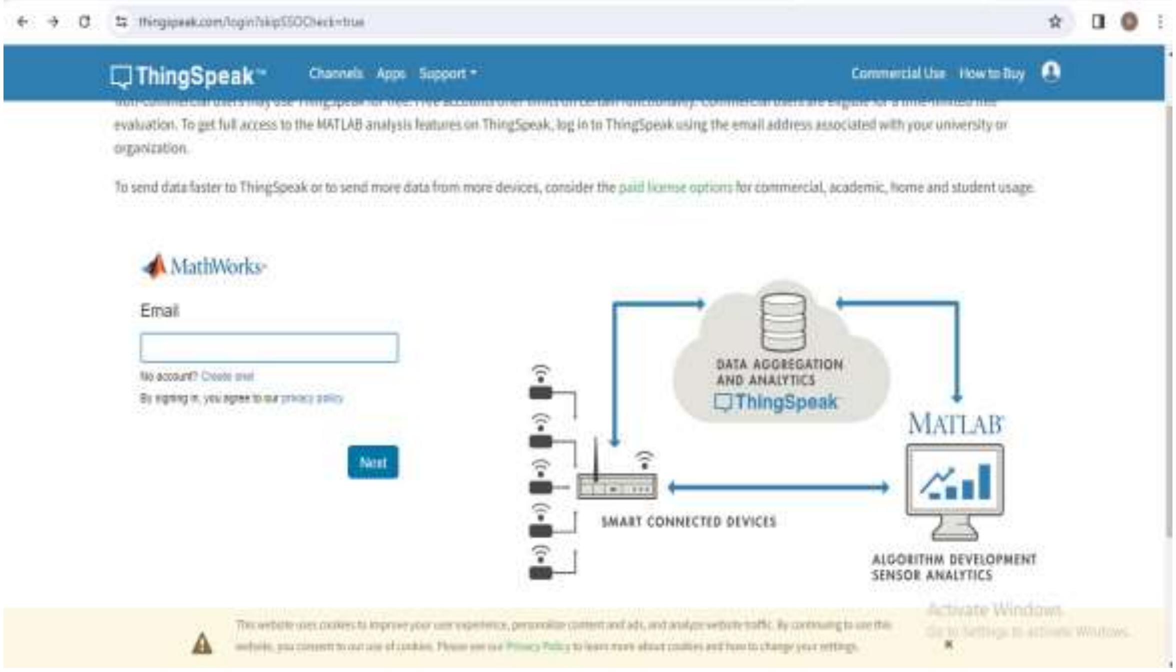
First Name

Last Name

Continue

DATA AGGREGATION AND ANALYTICS
ThingSpeak
MATLAB
SMART CONNECTED DEVICES
ALGORITHM DEVELOPMENT SENSOR ANALYTICS

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.



thingspeak.com/login?skipSSOCheck=true

ThingSpeak™ Channels Apps Support ▾ Commercial Use How to Buy

Non-commercial users may use ThingSpeak for free. Free accounts offer limits on certain functionality. Commercial users are eligible for a time-limited free evaluation. To get full access to the MATLAB analysis features on ThingSpeak, log in to ThingSpeak using the email address associated with your university or organization.

To send data faster to ThingSpeak or to send more data from more devices, consider the paid license options for commercial, academic, home and student usage.

MathWorks

Email

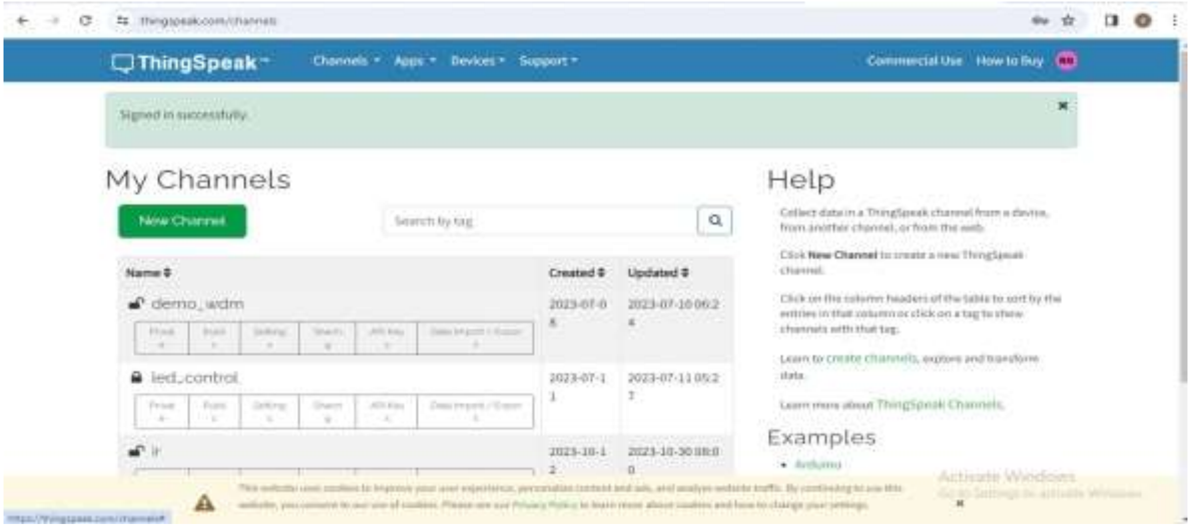
No account? Create one!

By signing in, you agree to our privacy policy

Next

DATA AGGREGATION AND ANALYTICS
ThingSpeak
MATLAB
SMART CONNECTED DEVICES
ALGORITHM DEVELOPMENT SENSOR ANALYTICS

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.



Signed in successfully.

My Channels

Name ⇕	Created ⇕	Updated ⇕
demo_wdm	2023-07-08	2023-07-10 06:24
led_control	2023-07-11	2023-07-11 05:27
ir	2023-10-12	2023-10-30 08:00

Help

Collect data in a ThingSpeak channel from a device, from another channel, or from the web.

Click **New Channel** to create a new ThingSpeak channel.

Click on the column headers of the table to sort by the entries in that column or click on a tag to show channels with that tag.

Learn to create channels, explore and transform data.

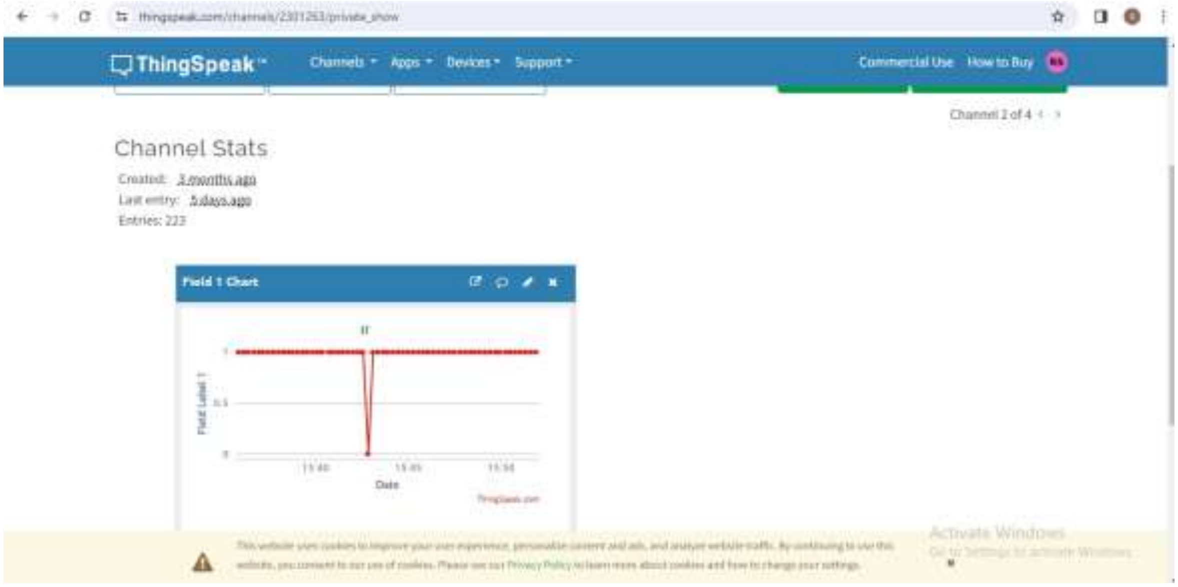
Learn more about ThingSpeak Channels.

Examples

- Arduino

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

Output:



Channel Stats

Created: 3 months ago

Last entry: 5 days ago

Entries: 223

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

RESULT:

The test was successfully completed.

Experiment 10	**Log Data using Raspberry PI and upload to the Cloud Platform**

Aim:

To test the log data using raspberry pi and upload to the cloud platform.

Apparatus Required:

1. Arduino IDE
2. NODE MCU
3. IR Sensor
4. Bread board

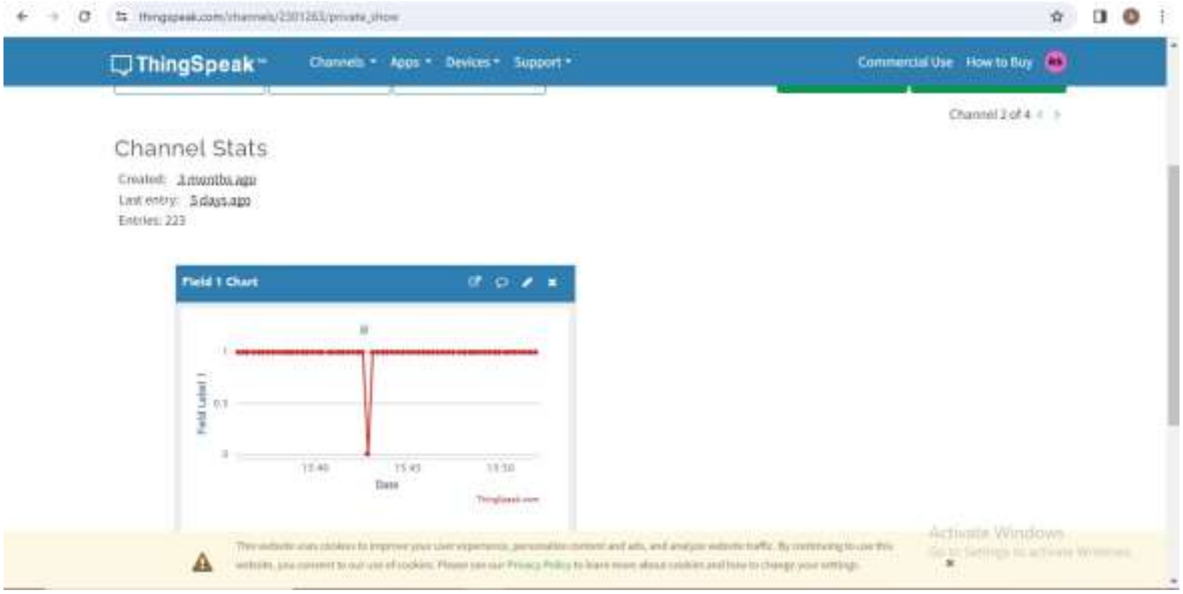
Procedure:

1. Connect the IR sensor to the Arduino.
2. The connections may vary based on your specific IR sensor model, but typically, it involves connecting the sensor's signal pin to a digital pin on the Arduino and power/ground pins appropriately.
3. Install the necessary libraries for the IR sensor.
4. For example, a common IR sensor like the TSOP382, you might need the "IRremote" library.

Program:

```
import machine
import utime
import urequests
import network

IR_SENSOR_PIN = 2  # GPIO pin for the IR
sensor API_KEY = "ZSX5SRTSRJ61XVU6"
THINGSPEAK_UR="https://api.thingspeak.com/update?api_key={}".format(API
_KEY)
WIFI_SSID = "@ms.Balaji"
WIFI_PASSWORD =
"@msbalaji"

ir_sensor = machine.Pin(IR_SENSOR_PIN, machine.Pin.IN)

wlan = network.WLAN(network.STA_IF)

def connect_to_wifi():
   wlan.active(True)
   if not wlan.isconnected():
      print("Connecting to WiFi...")
      wlan.connect(WIFI_SSID,
      WIFI_PASSWORD) while not
      wlan.isconnected():
      pass
      print("Connected to WiFi")

def read_ir_sensor():
   return ir_sensor.value()

def send_to_thingspeak(data): #
   Send data to ThingSpeak
   url = "{}&field1={}".format(THINGSPEAK_URL,
   data) response = urequests.get(url)
   print("ThingSpeak response:", response.text)

                                                                  # Connect to WiFi
connect_to_wifi()

while True:
   ir_data = read_ir_sensor() print("IR
   Sensor Data:", ir_data)
   # Send data to ThingSpeak via HTTP
   send_to_thingspeak(ir_data) utime.sleep(15)
```

Output:



Channel Stats

Created: 3 months ago

Last entry: 5 days ago

Entries: 223

This website uses cookies to improve your user experience, personalize content and ads, and analyze website traffic. By continuing to use this website, you consent to our use of cookies. Please see our Privacy Policy to learn more about cookies and how to change your settings.

RESULT:

The test was successfully completed.