Lab Code: 20ECL602

INTERNET OF THINGS LAB

Lab Manual

Department of Electronics & Communication Engineering

Bapatla Engineering College :: Bapatla

(Autonomous)

G.B.C. Road, Mahatmajipuram, Bapatla-522102, Guntur (Dist.)

Andhra Pradesh, India.

Web:**www.becbapatla.ac.in**

Contents

Bapatla Engineering College :: Bapatla

(Autonomous)

Vision

- To build centers of excellence, impart high quality education and instill high standards of ethics and professionalism through strategic efforts of our dedicated staff, which allows the college to effectively adapt to the ever changing aspects of education.

- To empower the faculty and students with the knowledge, skills and innovative thinking to facilitate discovery in numerous existing and yet to be discovered fields of engineering, technology and interdisciplinary endeavors.

Mission

- Our Mission is to impart the quality education at par with global standards to the students from all over India and in particular those from the local and rural areas.

- We continuously try to maintain high standards so as to make them technologically competent and ethically strong individuals who shall be able to improve the quality of life and economy of our country.

Bapatla Engineering College :: Bapatla

(Autonomous)

Department of Electronics and Communication Engineering

Vision

To produce globally competitive and socially responsible Electronics and Communication Engineering graduates to cater the ever changing needs of the society.

Mission

- To provide quality education in the domain of Electronics and Communication Engineering with advanced pedagogical methods.
- To provide self learning capabilities to enhance employability and entrepreneurial skills and to inculcate human values and ethics to make learners sensitive towards societal issues.
- To excel in the research and development activities related to Electronics and Communication Engineering.

Bapatla Engineering College :: Bapatla

(Autonomous)

Department of Electronics and Communication Engineering

Program Educational Objectives (PEO's)

PEO-I: Equip Graduates with a robust foundation in mathematics, science and Engineering Principles, enabling them to excel in research and higher education in Electronics and Communication Engineering and related fields.

PEO-II: Impart analytic and thinking skills in students to develop initiatives and innovative ideas for Start-ups, Industry and societal requirements.

PEO-III: Instill interpersonal skills, teamwork ability, communication skills, leadership, and a sense of social, ethical, and legal duties in order to promote lifelong learning and Professional growth of the students.

Program Outcomes (PO's)

Engineering Graduates will be able to:

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 provide valid conclusions.

PO5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and 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 in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. Individual and Teamwork: Function effectively as an individual, and as a member or leader in diverse teams, 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.

Bapatla Engineering College :: Bapatla

(Autonomous)

Department of Electronics and Communication Engineering

Program Specific Outcomes (PSO's)

PSO1: Develop and implement modern Electronic Technologies using analytical methods to meet current as well as future industrial and societal needs.

PSO2: Analyze and develop VLSI, IoT and Embedded Systems for desired specifications to solve real world complex problems.

PSO3: Apply machine learning and deep learning techniques in communication and signal processing.

INTERNET OF THINGS LAB
III B.Tech. VI Semester (Code: 20ECL602)

Lectures :	0 Hours/Week	Tutorial :	0 Hours/Week	Practical :	3 Hours/Week
CIE Marks :	30	SEE Marks :	70	Credits :	1.5

Pre-Requisite: None.

Course Objectives: Students will be able to	
➢	Demonstrate skills in programming edge devices using Arduino board and Node MCU.
➢	Master skills in programming edge devices using Raspberry Pi.
➢	Interface different Sensors with Arduino, Raspberry Pi and Node MCU.
➢	Design & Interface Edge Devices and actuators using various protocols for IoT applications.

Course Outcomes: At the end of the course, student will be able to	
CO1	Experiment with edge devices like Arduino.
CO2	Select appropriate sensors for designing an IoT Application.
CO3	Choose appropriate components with feasible communication interfaces to realize an application.
CO4	Design & develop IoT applications and solutions using latest controllers

Mapping of Course Outcomes with Program Outcomes & Program Specific Outcomes

	PO's												**PSO's**		
CO	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**1**	**2**	**3**
CO1	3	2		3	3				3				2	3	2
CO2	3	2		3	3				3				2	3	2
CO3	3	2		3	3				3				2	3	2
CO4	3	2	3	3	3				3				2	3	2
AVG	3	2	3	3	3				3				2	3	2

LIST OF EXPERIMENTS 36 Hours

S.No	**List of Experiments**
1	Interface digital I/O switch -LED Turn ON &OFF
2	Automatic Identification Based on IR sensor, Ultrasonic sensor
3	Display entered Keypad message in serial monitor
4	Acquired Analog Sensor signal data(LDR/LM35)and display on LCD

5	Interface Digital I/O-LED-Turn ON LED
6	Send or receive SMS using GSM
7	Interface Arduino with display service(RGB LED) to convey signal information
8	Pi connected with Actuator
9	Pi Connected with LDR sensor
10	Pi connected with PIR sensor and also Cloud to Write data
11	Pi connected with LDR sensor and cloud to Read data

NOTE: *A minimum of 10 (Ten) experiments have to be performed and recorded by the candidate to attain eligibility for Semester End Lab Examination*.

1 Interface digital I/O switch -LED Turn ON &OFF

Aim: -To interface digital, I/O switch -LED Turn ON &OFF LED for 1sec after 2sec.

Software Required: -Arduino IDE

Apparatus: -

Arduino UNO kit

LED's

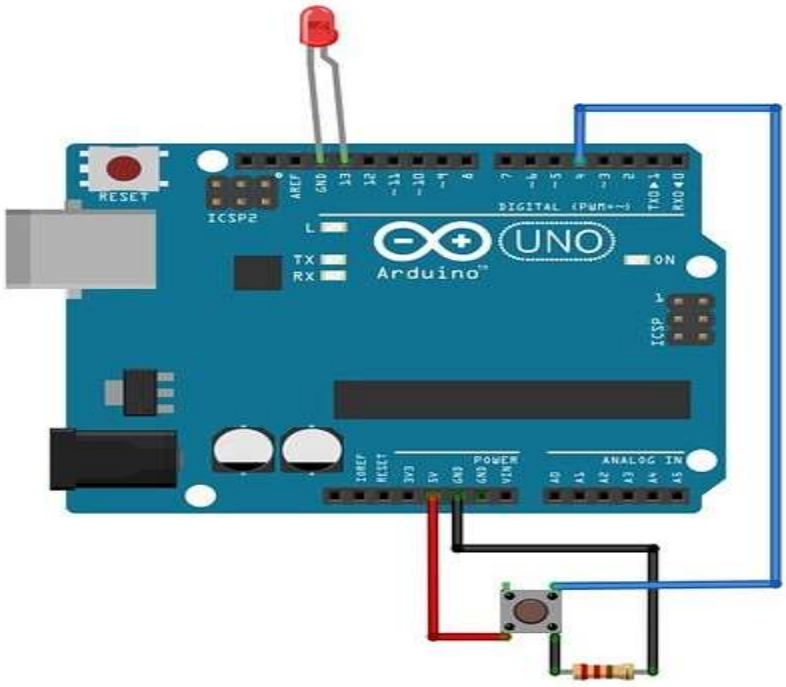
Trainer kit

Connecting wires

Jumping wires

USB cable

Theory: - To turn on a LED, the Arduino needs to send a HIGH Signal to one of its pins. To turn off the LED, needs to send a low signal to send a -low signal to the pin. You can make the LED flash by changing the length of the HIGH and low states. ranging LED should now be blinking on and off at a rate of 1000 milliseconds (1000 milliseconds = (Seconds) The hold the delays function on line tells the Arduino to High signal at pin 13 for 1000ms. The delay() function on line 8 tells is to fold the Low signal delay at pin 13 for 1000ms. You can change delay.



Procedure: -

1) The 4 LEDs are connected to the Arduino at pin numbers A0, A1, A2, A3.

2) The switches are Connected to the Arduin of pins 0,1,2,3.

3) Power jack is connected to the Arduino.

4) USB connector is connected between Arduino and monitor.

5) Connect the 12V power to IOT development board.

6) Then Observe the LED Outputs.

Source code:-

```
int led1=A0;
int led2=A1;
int led3=A2;
int led4=A3;
#define  sw1 0
#define  sw2 1
#define  sw3 2
#define  sw4 3
void setup() {
  // put your setup code here, to run once:
pinMode(led1,OUTPUT);
pinMode(led2,OUTPUT);
pinMode(led3,OUTPUT);
pinMode(led4,OUTPUT);
pinMode(sw1,INPUT);
```

```
pinMode(sw2,INPUT);
pinMode(sw3,INPUT);
pinMode(sw4,INPUT);
}
void loop() {
  // put your main code here, to run repeatedly:
int sw1_sts=digitalRead(sw1);
int sw2_sts=digitalRead(sw2);
int sw3_sts=digitalRead(sw3);
int sw4_sts=digitalRead(sw4);
if(sw1_sts==0)
{
digitalWrite(led1,1);
}
else
{
  digitalWrite(led1,0);
}
if(sw2_sts==0)
{
digitalWrite(led2,1);
}
else
```

```
{
  digitalWrite(led2,0);
}
if(sw3_sts==0)
{
digitalWrite(led3,1);
}
else
{
  digitalWrite(led3,0);
}
if(sw4_sts==0)
{
digitalWrite(led4,1);
}
else
{
  digitalWrite(led4,0);
}
}
```

Precautions: -

1. Avoid loose connections.
2. Check the ports and pins on Arduino Board with respect to code.

Result: -The interfacing of digital I/O switch-LED turn ON by using switch.

2 Automatic Identification Based on IR sensor, Ultrasonic sensor

Aim: -To interface Automatic Identification based on IR sensor, Ultrasonic sensor with Arduino.

Software Required: -Arduino IDE

Apparatus: -

Arduino UNO kit

IR sensor

Ultrasonic sensor

Trainer kit

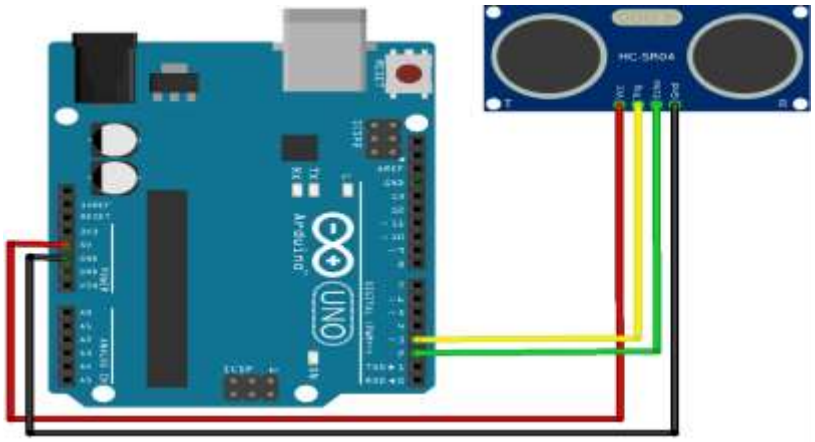
Connecting wires

Jumping wires

USB cable

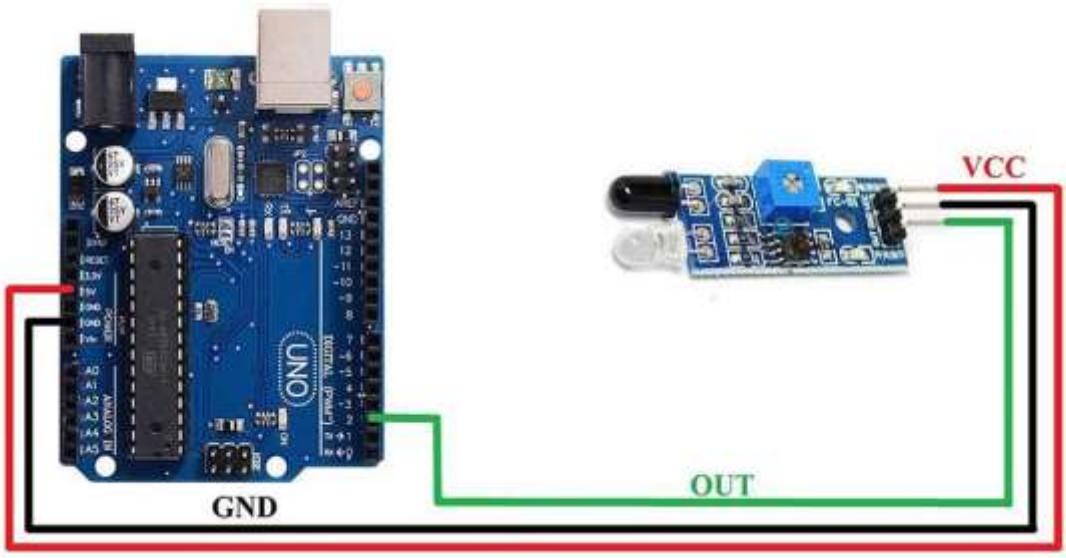
Theory: -

Ultrasonic sensor: Ultrasonic sensor is wed to measure the distance of the objects surroundings us. The principle of these sensors is that they emanate ultrasonic sound which travels and hits the objects surrounding us. The ultrasonic pulse in captured by the sensor by the performing some math on the obtained value we can get the distance of the object from the senses.



IR sensor:

In real-time applications IR sensor for detecting the obstacles. when IR sensor detects on obstacle present in front of it. The sensor gives cut an output voltage which is of order of few millivolts. One has to note this generated voltage is digital, that is either 100 0. So, we can just know whether an obstacle is present or not.



Procedure:-

1) Data pin of IR tensor connected with Arduino board at pin no-8. positive and negative supply pins of sensors are connected at 5V + ground pirs of Arduino boards.
2) Power Jack is connected to the arduino.
3) USB Connected between arduino and monitor.
4) Connect the l2v power supply to IoT development board.
5) For ultrasonic sensor, data pins trig & echo is connected with arduino board of pin no.9 and 8 respectively
6) Remaining steps which are described is It sensor are same as in ultrasonic sensor. Observe the o/p in serial monitor & note down the corresponding values.

Source code: -

```
//Ir sensor
int is=8;
void setup() {
pinMode(is,INPUT);
Serial.begin(9600);
}
void loop() {
 int is_sts=digitalRead(is);
Serial.println(is_sts);
delay(1000);
}
//Ultrasonic sensor
int trig=9;
int echo=8;
void setup() {
  // put your setup code here, to run once:
pinMode(trig,OUTPUT);
pinMode(echo,INPUT);
Serial.begin(9600);
}

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

```
  digitalWrite(trig,1);
  delayMicroseconds(10);
  digitalWrite(trig,0);
  int duration=pulseIn(echo,1);
  int dist=duration/58.2;
 Serial.println(dist);
  delay(1000);
}
```

Precautions: -

1. Avoid loose connections.
2. Check the ports and pins on Arduino Board respectively.

Result: -

The automatic Identification is done by using IR sensor and Ultrasonic sensor and outputs are observed.

3 Display entered Keypad message in serial monitor

Aim: -To display entered Keypad message in serial monitor by interfacing aduino board.

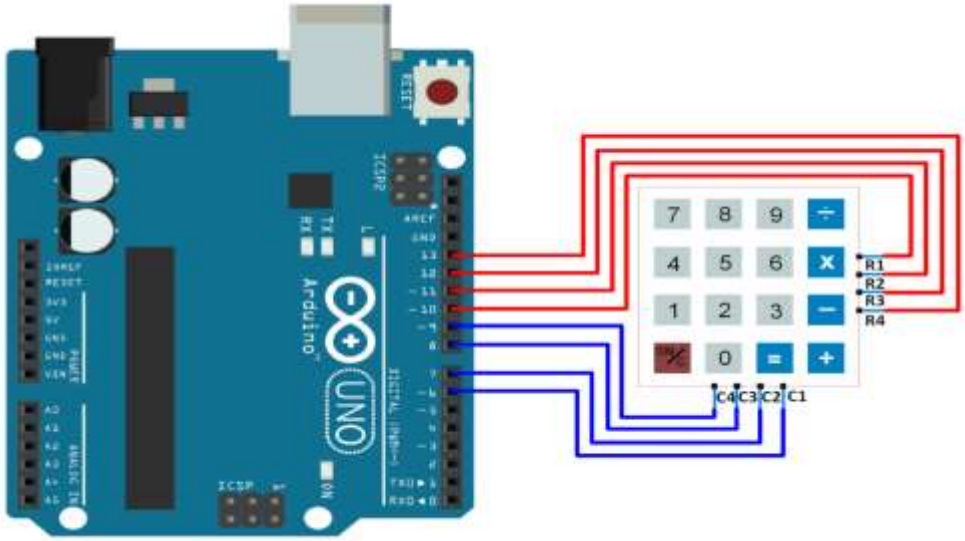
Software Required: -Arduino IDE

Apparatus: - Arduino UNO kit, Trainer kit, connecting wires, Jumping wires, USB cable

Theory: -

keypads are a great way to let users interact with your project. You can use them to navigate menu, enter passwords, and control games and robots. The button on a keypad are arranged in rows and columns. A 4x4 keypad has 4 columns and 4 rows.

Beneath each key is a membrane switch each switch in a row is connected by conductive trace underneath the pad. Each switch in a column is connected the same way. One side of the switch is connected of all of the other switcher in that column by a conductive trace each row i.e and column bought out of a single pin.



Procedure: -

1)Connect the keypad with the Arduino board.

2) Power jack is connected to the Arduino.

3) USB connector is connected between Arduino & monitor.

4) Connect the 12v power to IoT development Board.

5) Press any keys on the keypad and that can be seen as output.

6) Observe the keypad numbers on the screen

Source code:-

```
#include<Keypad.h>
const byte Rows=4;
const byte Cols=4;
char hexaKeys[Rows][Cols]={
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'},
};
byte rowPins[Rows]={1,2,3,4};
byte ColPins[Cols]={A1,A2,A3,A4};
Keypad
customKeyPad=Keypad(makeKeymap(hexaKeys),rowPins,ColPins,Rows,Cols);
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}
void loop() {
```

```
// put your main code here, to run repeatedly:
char customKey=customKeyPad.getKey();
if(customKey)
{
  Serial.println(customKey);
}}
```

Precautions:-

1. Avoid loose connections .
2. Check the ports and pins on Arduino Board with respect to code.

Result:- Entered keypad message is displayed in serial monitor by using Arduino .

4 Acquired Analog Sensor signal data(LDR/LM35)and display on LCD

Aim: - To Acquired analog sensor data (LDR/LM35) and display on LCD.

Software Required: -Arduino IDE

Apparatus: - Arduino UNO kit

LDR sensor

LM35 sensor

Trainer kit

Connecting wires

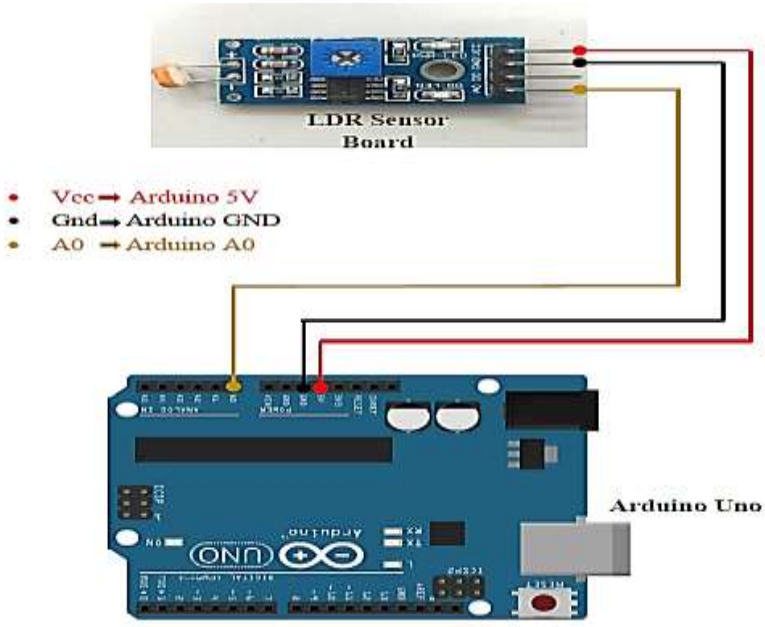
Jumping wires

USB cable

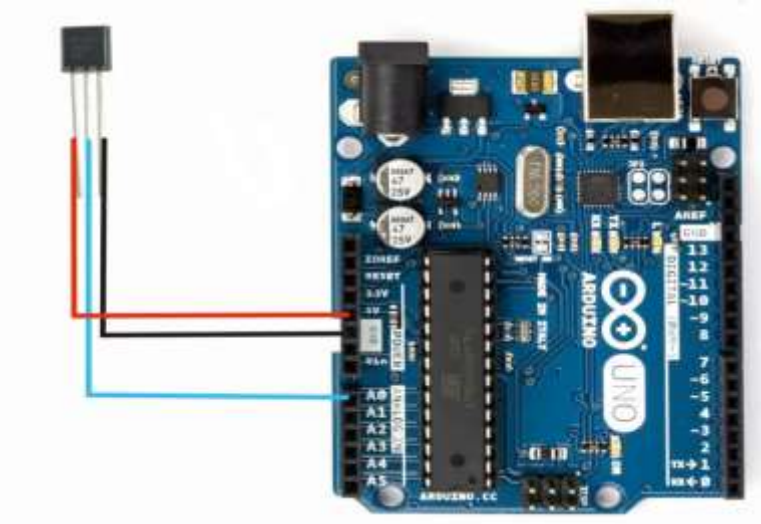
Theory: -

LDR: -

Light dependent resistor (LDR) is a component a (variable) resistance which changes with light intensity that falls upon it. This allows them to be used in light serving circuits. Hence LDR's are the sensors which are employed in the fields where we have a necessary of monitoring the intensity of light.



LM35: - Lm35 is a low-power, low-cost, high precision temperature Sensor designed and manufactured by Texas instrument. This IC provider a voltage output that is linearly proportional to the change in temperature**.**



Procedure: -

1) Data pin is connected to the Arduino Board A0

2 Positive and negative pins of the sensors to 5v and ground pin of Arduino Board.

3) Power jack is connected to the Arduino.

4)USB connector is connected to the Arduino and monitor.

5) Connect the 12v power to IOT development Board.

6) By closing the level we operate LDR sensor and LM35 sensor gives the temperature value by serving the temperature.

7) Observe the output in the serial monitor.

Source code: -

```
//LDR sensor

int ldrs1=A0;

void setup() {

  // put your setup code here, to run once:
```

```
pinMode(ldrs1,INPUT);
Serial.begin(9600);
}
void loop() {
  // put your main code here, to run repeatedly:
int ldrs_val = analogRead(ldrs1) ;
Serial.println(ldrs_val);
delay(1000);
}
//LM35 sensor
int ts=A0;
void setup() {
pinMode(ts,INPUT);
Serial.begin(9600);
}
void loop() {
int ts_val = analogRead(ts) ;
ts_val=ts_val* 0.48;
Serial.println(ts_val);
delay(1000);
}
```

Precautions: -

1. Avoid loose connections.

2. Check the ports and pins on Arduino Board with respect to code.

Result: -Hence acquired analog sensor signal data of LDR and LM35 sensors are on serial board observed.

5 Interface Digital I/O-LED-Turn ON LED

Aim: -To interface digital, I/O-LED-Turn ON LED with Arduino.

Software Required: -Arduino IDE

Apparatus: - Arduino UNO kit

LED’s

Trainer kit

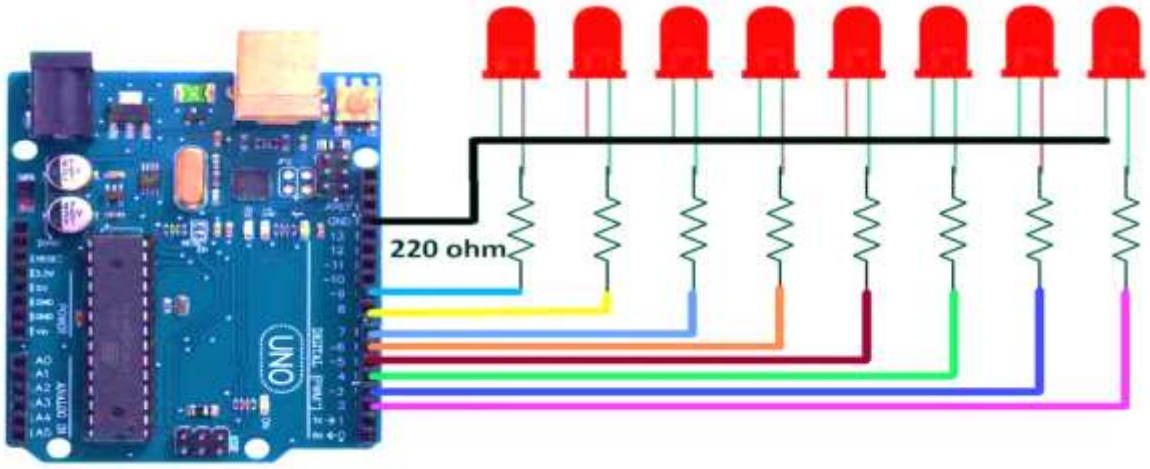
Connecting wires

Jumping wires

USB cable

Theory: -

LED has two leads, a longer one called the anode (+) and a shorter one called the cathode (-). Current flows from the anode to the cathode, and when current flows through the LED, it emits light. LEDs have specific voltage and current requirements. Exceeding these values can damage the LED. Usually, the forward voltage (voltage required for the LED to turn on) and forward current (current required for optimal brightness) are specified in the LED's datasheet. To prevent the LED from drawing too much current from the Arduino's pin, a current-limiting resistor is used in series with the LED. The value of this resistor depends on the LED's forward voltage and the desired current. You can use Ohm's law (V = IR) to calculate the resistance needed. Arduino Uno has digital pins that can be set to output mode to provide voltage to an external component. These pins can supply either 5V or 3.3V depending on the board model. For driving an LED, we typically use the digital pins configured as outputs to provide a voltage source. Connect the longer leg (anode) of the LED to the digital pin of the Arduino Uno via the current-limiting resistor. Connect the shorter leg (cathode) of the LED to the ground (GND) pin of the Arduino Uno.



Procedure: -

1) The 4 LED's are connected to the Arduino at pin numbers A0,A1,A2,A3.

2) Power jack is connected to the Arduino.

3) USB connector is connected between Arduino and monitor.

4) Connect the 12V power to IOT development board.

5) Then Observe the LED Outputs.

Source code: -

```
int led1=A0;
int led2=A1;
int  led3=A2;
int led4=A3;
void setup() {
pinMode(led1,OUTPUT);
pinMode(led2,OUTPUT);
pinMode(led3,OUTPUT);
```

```
pinMode(led4,OUTPUT);
}
void loop() {
digitalWrite(led1,1);
digitalWrite(led2,1);
digitalWrite(led3,1);
digitalWrite(led4,1);
delay(1000000);
digitalWrite(led1,0);
digitalWrite(led2,0);
digitalWrite(led3,0);
digitalWrite(led4,0);
delay(100);
}
```

Precautions: -

1. Avoid loose connections.
2. Check the ports and pins on Arduino Board with respect to code.

Result: -Hence interfacing of digital I/O LED to turn ON using Arduino.

6 Send or receive SMS using GSM.

Aim: - To interface Arduino with GSM module and perform SMS sending and receiving operations.

Software Required: -Arduino IDE

Apparatus: - Arduino UNO kit

GSM module

Trainer kit

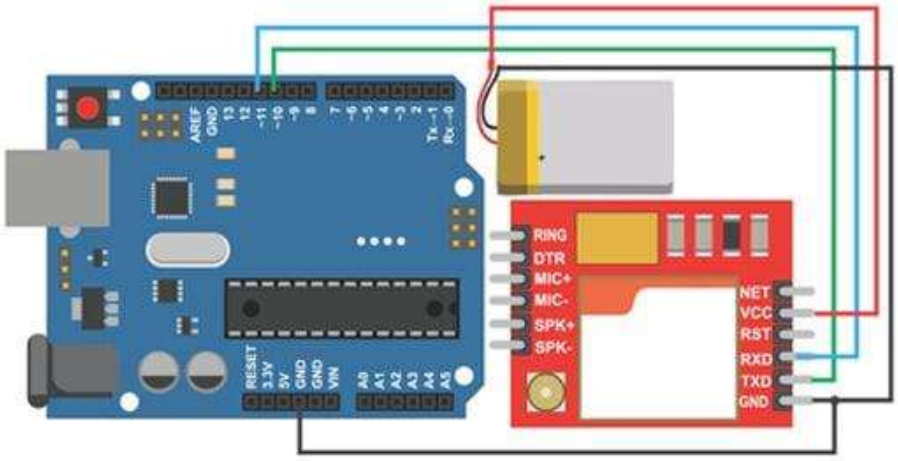
Connecting wires

Jumping wires

USB cable

Theory: -

A GSM module allows your Arduino project to connect to the GSM network, enabling calls, texts, and data. The module communicates with Arduino through serial interface (UART). Arduino sends AT commands to the GSM module, which executes them and responds. The GSM module connects to the network using a SIM card and antenna. AT commands include basic, SMS, and call commands. The Arduino library simplifies communication with the GSM module. It provides functions for sending and receiving SMS, making and receiving calls, and accessing data services. This setup enables remote control and monitoring, IoT projects, automation systems, security systems, and GPS tracking. Ensure compatibility and specific usage instructions for your GSM module and Arduino board. Use the GSM library in Arduino IDE for ease of use. Initialize the GSM module in your Arduino sketch. Use AT commands to configure the module. Send and receive SMS messages. Make and receive voice calls. Access data services like internet. Control your project remotely. Monitor sensors and data. Automate tasks. Enhance security. Track location. Expand your project's capabilities with GSM connectivity.



Procedure: -

1) Interface GSM module pins RXD, TXD with pin number 10,11 of Arduino.

2) Power jack is connected to the Arduino.

3) USB connector is connected between Arduino and monitor.

4) Connect the 12V power to IOT development board.

5) Then Observe the Outputs.

Program:

```
#include <SoftwareSerial.h>

SoftwareSerial Sim (9, 10);

void setup()

{

Sim.begin(9600);

Serial.begin(9600);

delay(100);
```

```
}
void loop()
{
if (Serial.available()>0)
switch(Serial.read())
{
case 's':
SendMessage();
break;
case 'r':
ReceiveMessage();
break;
}
if (Sim.available()>0)
Serial.write(Sim.read());
}
void SendMessage()
{
Sim.println("AT+CMGF=1");
delay(1000);
Sim.println("AT+CMGS=\"+918096176479\"\r");
delay(1000);
Sim.println("SMS Program from IOT Lab");
```

```
delay(100);
Sim.println((char)26);
delay(1000);
}
void ReceiveMessage()
{
Sim.println("AT+CNMI=2,2,0,0,0");
delay(1000);
}
```

Result: - Interfacing Arduino with GSM module to send and receive the SMS.

7 Interface Arduino with display service (RGB LED) to convey signal information

Aim: -To interface Arduino with display service(RGB LED) to convey signal information.

Software Required: -Arduino IDE

Apparatus: - Arduino UNO kit

RGB LED

Trainer kit

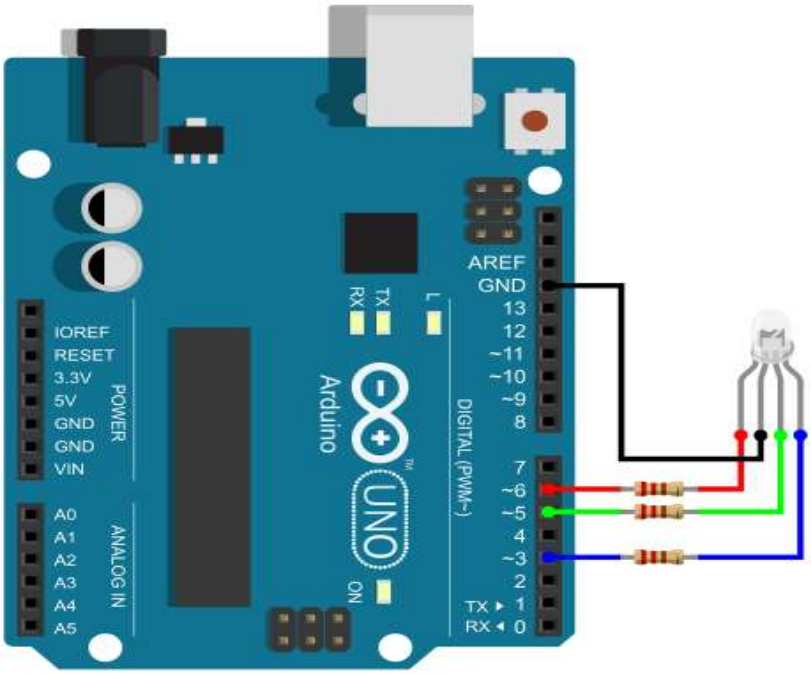
Connecting wires

Jumping wires

USB cable

Theory: -

RGB LED Basics: An RGB LED is a special type of LED that contains three separate LED chips inside – one for red, one for green, and one for blue. By controlling the intensity of each of these three colors, a wide range of colors can be produced. Common Anode/Cathode: RGB LEDs can be either common anode or common cathode. In a common anode RGB LED, the anode of all three internal LEDs is connected together, while in a common cathode RGB LED, the cathode of all three LEDs is connected together. You need to identify which type you have, as the wiring and code will differ. Voltage and Current: RGB LEDs, like single-color LEDs, have specific voltage and current requirements. Exceeding these values can damage the LED. Ensure you know the forward voltage and forward current for each color LED within the RGB LED. As with single-color LEDs, current-limiting resistors are needed to prevent the RGB LED from drawing too much current from the Arduino's pins. Calculate the appropriate resistor values for each color LED using Ohm's law as described earlier. Connect each color pin of the RGB LED to different digital output pins on the Arduino Uno. Make sure to use pins configured as outputs. For a common anode RGB LED, connect the common anode to the Arduino's 5V pin (or another appropriate voltage source). For a common cathode RGB LED, connect the common cathode to the Arduino's GND pin. Connect the respective color pins through current-limiting resistors to the Arduino's digital output pins.



Procedure: -

1) Connect the common anode and cathode pin of RGB LED to the 5V, GND pin on the Arduino.

2) Connect the red pin of the RGB LED to a digital output pin on the Arduino.

3) Connect the greens & blue pin of the RGB LED to another digital output pin on the Arduino.

4) Use Jumper wires to establish these connections.

5) Write the program to control RGB LED based on the desired signal information.

Source code: -

```
const int redLED=0;

const int greenLED=4;

const int blueLED=5;

void setup() {

  // put your setup code here, to run once:

  pinMode(redLED,OUTPUT);
```

```
  pinMode(greenLED,OUTPUT);
  pinMode(blueLED,OUTPUT);
  digitalWrite(redLED,0);
  digitalWrite(greenLED,0);
  digitalWrite(blueLED,0);
}
void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(redLED,1);
  digitalWrite(greenLED,0);
  digitalWrite(blueLED,0);
  delay(200);
  digitalWrite(redLED,0);
  digitalWrite(greenLED,1);
  digitalWrite(blueLED,0);
  delay(200);
  digitalWrite(redLED,0);
  digitalWrite(greenLED,0);
  digitalWrite(blueLED,1);
  delay(200);
}
```

Result: - Interfacing Arduino with display service(RGB LED) to convey signal information observed.

8 Pi connected with Actuator

Aim: -To interface actuator with Raspberry pi and also detect motion.

Hardware required: -

1.Universal IoT trainer kit with Raspberry pi
2.Connecting wires
3.Jumper wires
4.USB cables

Software Required: -Thonny Raspberry pi

Theory: -

PIR sensor detects motion by measuring changes in infrared radiation within its field of view. It consists of a pyroelectric sensor that generates a voltage when exposed to infrared radiation, typically emitted by warm objects in its detection range. When motion is detected, the PIR sensor outputs a digital signal indicating the presence of motion. It has a warm-up time during which it stabilizes, followed by a detection time during which it remains active after detecting motion. PIR sensors typically operate at 3.3V or 5V. Make sure to check the specifications of your sensor and provide the appropriate voltage level from the Raspberry Pi. PIR sensors usually have three pins: VCC (power), GND (ground), and OUT (signal output). Connect the VCC pin to a 3.3V or 5V pin on the Raspberry Pi, the GND pin to a ground pin, and the OUT pin to any GPIO pin configured as an input.

Procedure:-

1) Give the data pin of PIR Sensor to the 8th pin of the Raspberry pi board.

2) Connect Ground and positive pins to positive and negative pins
3) Give power supply connection to IOT kit.
4) Connect USB Jack from monitor to Raspberry Pi.
5) Compile and save the program written in Thony Raspberry Pi.
6) Dump the code on to the microcontroller.
7) Observe the output whether motion is detected or not movement of human.

Source Code: -

```
import RPi.GPIO as GPIO
from time import sleep
GPIO.setmode(GPIO.BOARD)
GPIO.setup(8, GPIO.OUT)
pwm=GPIO.PWM(8, 50)
pwm.start(0)


def SetAngle(angle):
duty = angle / 18 + 2
GPIO.output(8, True)
pwm.ChangeDutyCycle(duty)
sleep(1)
GPIO.output(8, False)
pwm.ChangeDutyCycle(0)


SetAngle(0)
pwm.stop()
GPIO.cleanup()
```

Precautions: -

1.Avoid loose connections.

2.Check the ports and pins on Raspberry pi Board with respect to code.

Result: - Hence PIR sensor is interfaced with Raspberry pi and motion is detected.

9 Pi Connected with LDR sensor

Aim: -To interface LDR sensor with raspberry pi

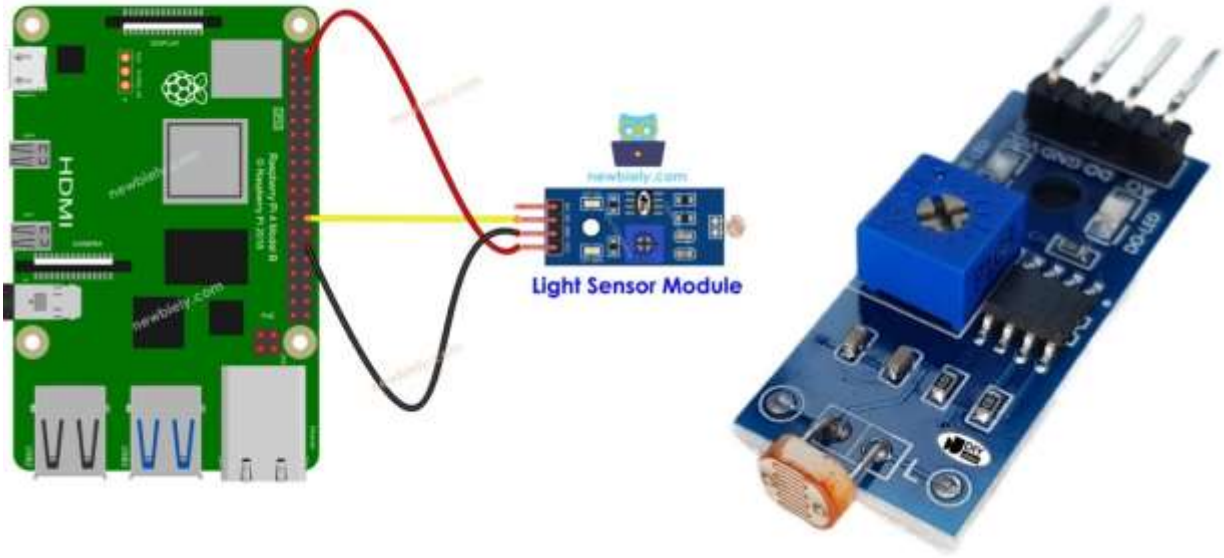
Hardware required: -

1.Universal Iot trainer kit with Raspberry pi
2.Connecting wires
3.Jumper wires
4.USB cables

Software Required: -Thonny Raspberry pi

Theory: -

Light dependent resistor (LDR) is a component a (variable) resistance which changes with light intensity that falls upon it. This allows them to be used in light serving circuits. Hence LDR's are the sensors which are employed in the fields where we have a necessary of monitoring the intensity of light.



Procedure: -

1) Give the data pin of LDR Sensor to the 36 pin of the Raspberry pi board.
2) Connect Ground and positive pins to positive and negative pins.
3) Give power supply connection to IOT kit.
4) Connect USB Jack from monitor to Raspberry Pi.
5) Compile and save the program written in Thony Raspberry Pi.
6) Dump the code on to the microcontroller.
7) Observe the output whether motion is detected or not movement of human.

Source Code: -

```
import RPi.GPIO as GPIO
import time
light = 36 #Board number
def setup():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(light,GPIO.IN)
def read_light():
    while True:
        light_state = GPIO.input(light)
        if light_state == 0:
            print("Light Detected")
        elif light_state == 1:
            print("Light Not Detected")

        time.sleep(.3)
def destroy():   #When program ending, the function is executed.
    GPIO.cleanup()

if __name__ == '__main__': #Program starting from here
    try:
        setup()
        read_light()
    except KeyboardInterrupt:
        destroy()
```

Result: - Hence LDR sensor is interfaced with Raspberry pi and observed.

10 Pi connected with PIR sensor and also Cloud to Write data

Aim: - To interface PIR sensor and also cloud to write data with raspberry pi.

Hardware required: -

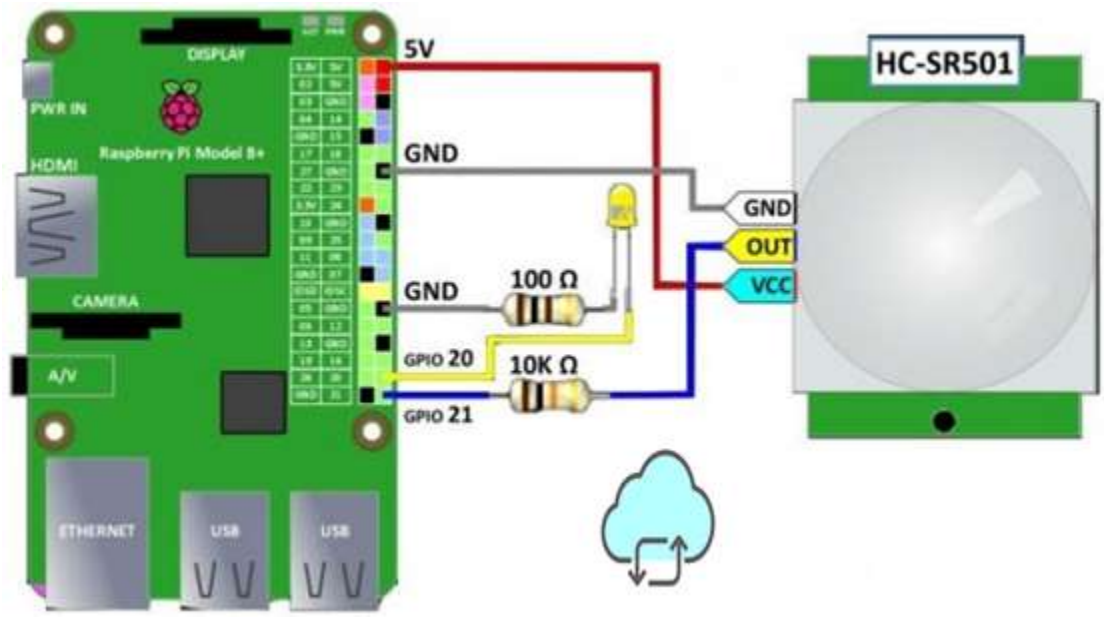
1.Universal Iot trainer kit with Raspberry pi
2.Connecting wires
3.Jumper wires
4.USB cables

Software Required: -Thonny Raspberry pi

Theory: -

Connect the PIR sensor to the Raspberry Pi. The PIR sensor typically has three pins: VCC (power), GND (ground), and OUT (signal output). Connect the VCC pin to a 3.3V pin on the Raspberry Pi, the GND pin to a ground pin, and the OUT pin to any GPIO pin configured as an input.Write a Python script to interact with the GPIO pins on the Raspberry Pi. Use the RPi.GPIO library to detect changes in the PIR sensor's output. When motion is detected, the sensor sends a signal to the Raspberry Pi, which triggers an event in the Python script.

Interfacing a PIR sensor with a Raspberry Pi and sending the sensor data to the cloud involves combining hardware and software components. Connect the PIR sensor to the Raspberry Pi. The PIR sensor typically has three pins: VCC (power), GND (ground), and OUT (signal output). Connect the VCC pin to a 3.3V pin on the Raspberry Pi, the GND pin to a ground pin, and the OUT pin to any GPIO pin configured as an input.Select a cloud service provider where you want to store your sensor data. Popular options include AWS (Amazon Web Services), Google Cloud Platform, Microsoft Azure, or IoT platforms like Adafruit IO or ThingSpeak.Set up an account on your chosen cloud service and create a new project or IoT device. Obtain the necessary credentials (e.g., access keys, endpoints) to authenticate and communicate with the cloud service.



Procedure: -

1)Connect data pin of PIR Sensor to 36th pin of Raspberry pi
2) Give Ground the sensor as well as supply connection to
3) Power Supply it given to the board.
4)USB connections we given to board.
5)Open Thingspeak website and create new channel.
6)Through API key, we connect to the microcontroller write API key along with channel ID to connect data to cloud.
7)In channel view,we can analyse data read from sensor.

Source code: -

```
#importing required library packages
import time
import RPi.GPIO as GPIO
import urllib.request
#selecting the bard mode and disabling default warnings
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
#pin number assignment
ls=36
#selecting pin I/O functionality
GPIO.setup(ls,GPIO.IN)
while True:
  # reading adc channel

  lval=GPIO.input(ls)

  print("LDR:" + str(lval))
  time.sleep(1)
  # checking the value with threshold level
  try:from time import sleep
import time
import urllib.request
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
while True:

r_link='https://api.thingspeak.com/channels/2418448/fields/1/last?api_key=F1CXHXN40LHR25
KU'
   f=urllib.request.urlopen(r_link)
   rcv = (f.readline()).decode()
   print("Retrived:"+str(rcv))

urllib.request.urlopen("https://api.thingspeak.com/update?api_key=LT3XH6MFQVQLW1RY&f
ield1=" + str(lval))
  except:
```

```
print('Check Internet connection')
```

Result: -Hence PIR sensor data is successfully started in cloud through thingspeak website.

11 Pi connected with LDR sensor and cloud to Read data

Aim: -To interface LDR sensor and cloud to read data with raspberry pi.

Hardware required: -

1.Universal IoT trainer kit with Raspberry pi
2.Connecting wires
3.Jumper wires
4.USB cables

Software Required: -Thonny Raspberry pi

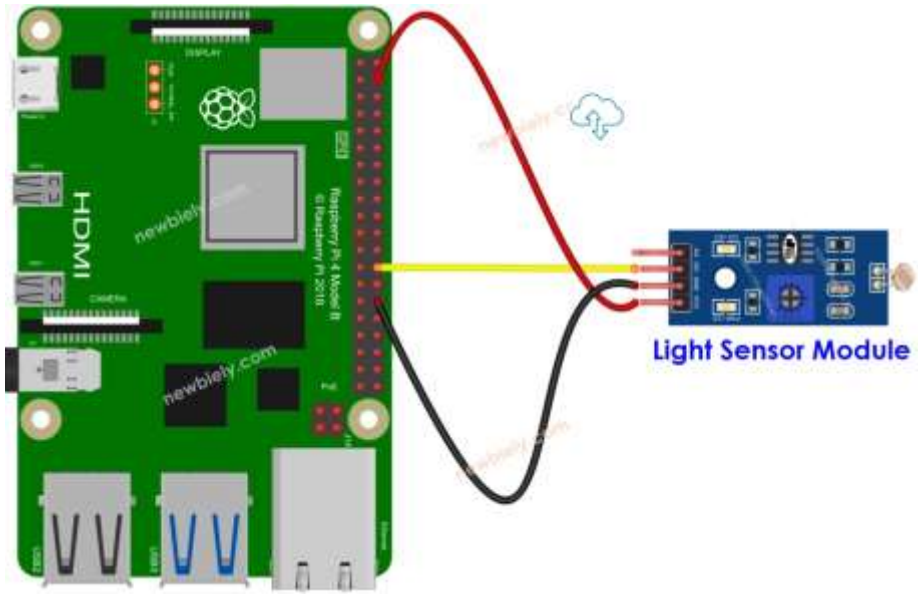
Theory: -

Connect the LDR sensor to the Raspberry Pi. The LDR typically has two legs. One leg is connected to a GPIO pin on the Raspberry Pi, and the other is connected to a voltage source (3.3V or 5V) or ground (GND), depending on the circuit configuration. Write a Python script to read the analog voltage from the LDR sensor using the Raspberry Pi's ADC (Analog-to-Digital Converter) or GPIO pins. If using GPIO pins, configure the pin as an input and use the RPi. GPIO library to read its value. If using an ADC, you may need to install additional libraries and communicate with the ADC via I2C or SPI. Select a cloud service provider where you want to store your sensor data. Popular options include AWS (Amazon Web Services), Google Cloud Platform, Microsoft Azure, or IoT platforms like Adafruit IO or ThingSpeak. Set up an account on your chosen cloud service and create a new project or IoT device. Obtain the necessary credentials (e.g., access keys, endpoints) to authenticate and communicate with the cloud service. Install any SDKs or libraries provided by the cloud service to interact with its APIs. For example, for AWS, you might install the AWS SDK for Python (boto3). For Google Cloud Platform, you would install the Google Cloud Client Library. Modify your Python script to send the sensor data to the cloud at regular intervals or when a threshold is exceeded. Use the credentials obtained earlier to authenticate with the cloud service and send data to the appropriate endpoint or topic. Ensure that your data is encrypted when transmitted to the cloud to prevent unauthorized access. Most cloud service providers offer encryption options for data in transit. Use secure authentication mechanisms (e.g., access tokens, API keys) to authenticate your Raspberry Pi with the cloud service. Avoid hardcoding sensitive credentials in your code. Configure appropriate access control policies on your cloud resources to restrict access to authorized users and devices. The Python script running on the Raspberry Pi reads the analog voltage from the LDR sensor.

The script formats the sensor data and securely transmits it to the cloud service using the cloud SDK or libraries. The cloud service receives the data, stores it in a database, and makes it available for analysis, visualization, or further processing.

Procedure: -

1) Power supply is given to the board
2) USB Connections given to the board.
3) Open thingspeak Website, Create new channel.
4) Through API key, we can connect to microcontroller write API key along with channel Id to connect data to cloud.
5) In channel view, from sensor can analyse data read from the sensor.



Source code: -

```
from time import sleep
import time
import urllib.request
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
while True:

r_link='https://api.thingspeak.com/channels/2418448/fields/1/last?api_key=F1CXHXN40LHR25
KU'
    f=urllib.request.urlopen(r_link)
    rcv = (f.readline()).decode()
    print("Retrived:"+str(rcv))
```

Result: - Hence Raspberry Pi connected with LDR sensor and cloud to Read data is done.