SMART HOME
AUTOMATION & SECURITY SYSTEM

PROJECT

Arab Academy for Science and Technology
College of Computing and Information Technology
Artificial Intelligence Department

SUBMITTED BY:

- Laila Tarek — 231003520
- Hana Tariq — 231000481
- Miran Samer — 231010991
- Hana Mabrouk — 231002230
- Lojaine Mohamed — 231000453

SUPERVISED BY:

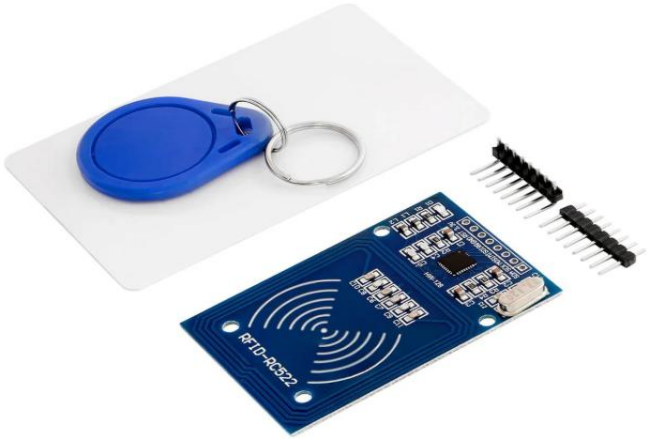
Dr. Omar Shalash , Eng Samar

RC522 RFID Module User Manual

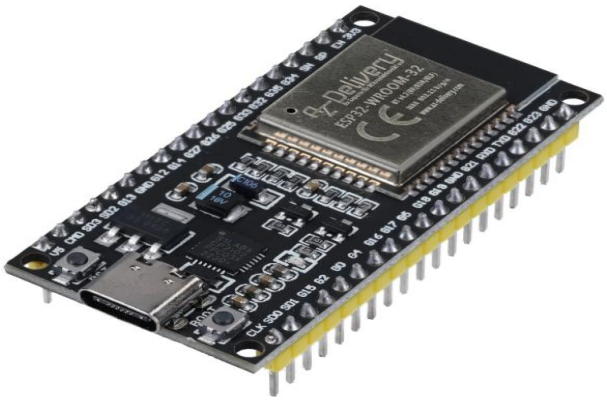
Section: ESP32 Setup Guide

1. Hardware Requirements

- **RC522 RFID Module + Tags/Cards** (13.56 MHz)

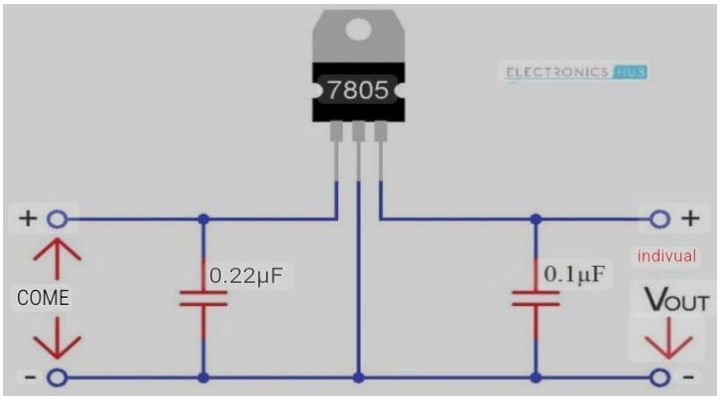


- **ESP32 WROOM** (instead Arduino Uno for connection to WIFI , via SPI)

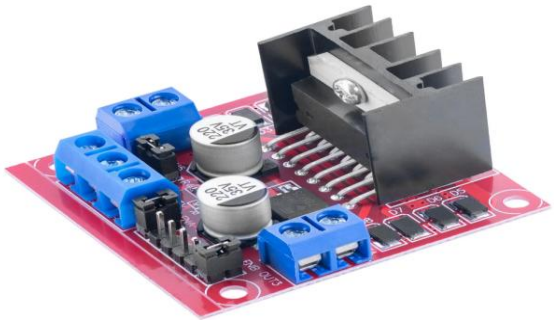


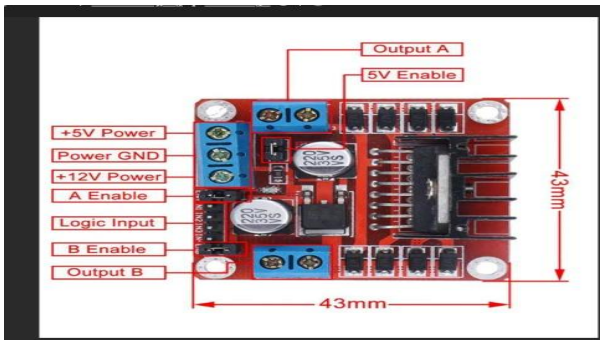
- **Jumper Wires** (for SPI connection)

- **Regulator 7805** (takes 12volt and gives 5volt)



- **H-Bridge Motor Driver L298N** (switches between 12volt and 0volt aka grnd)





- **Center-Lock** (opens/closes the door)

- **16x2 I2C LCD Display** (to display the output)

- **Buzzer** (12volt)

2. Wiring (SPI Connection)

Component's Pins	ESP32 Pins	Notes
RC522 (SDA/SS)	GPIO5 (VSPI_SS)	SPI Chip Select
RC522 (SCK)	GPIO18 (VSPI_SCK)	SPI Clock
RC522 (MOSI)	GPIO23 (VSPI_MOSI)	SPI Data Out
RC522 (MISO)	GPIO19 (VSPI_MISO)	SPI Data In
RC522 (RST)	GPIO21	Reset (Flexible pin)
I^2C LCD (SDA)	GPIO21	I^2C Data (Shared with RST)
I^2C LCD (SCL)	GPIO22	I^2C Clock
H-Bridge (IN1)	GPIO12	Lock actuator control
H-Bridge (IN2)	GPIO13	Unlock control
Buzzer	GPIO15	Active high

3. Code Setup (ESP32-Specific)

3.1 Required Libraries

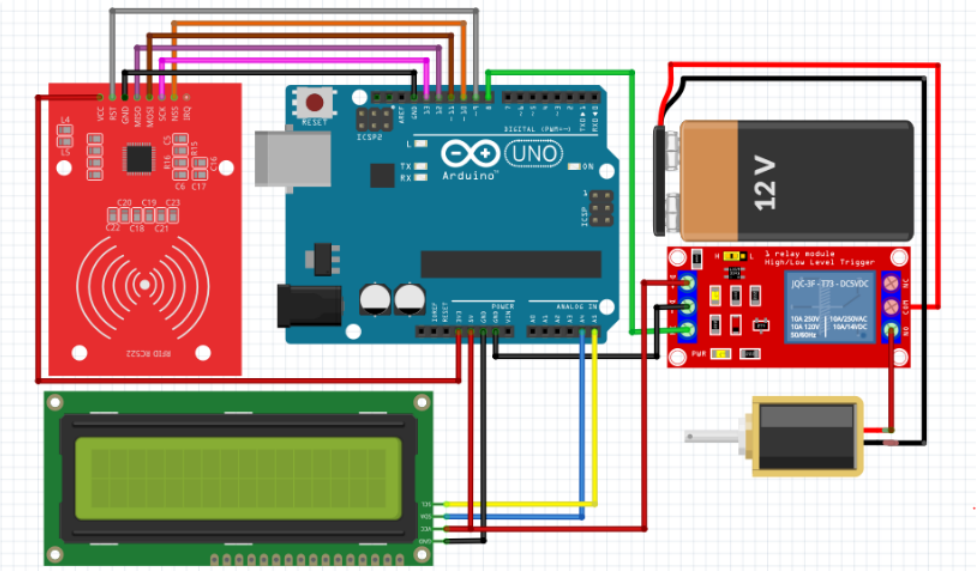
1. MFRC522 (RFID) → Install via Arduino Library Manager.
2. LiquidCrystal_I2C (LCD) → Use this version.
3. SPI.h (Built-in for ESP32).

3.2 ESP32-Compatible code:

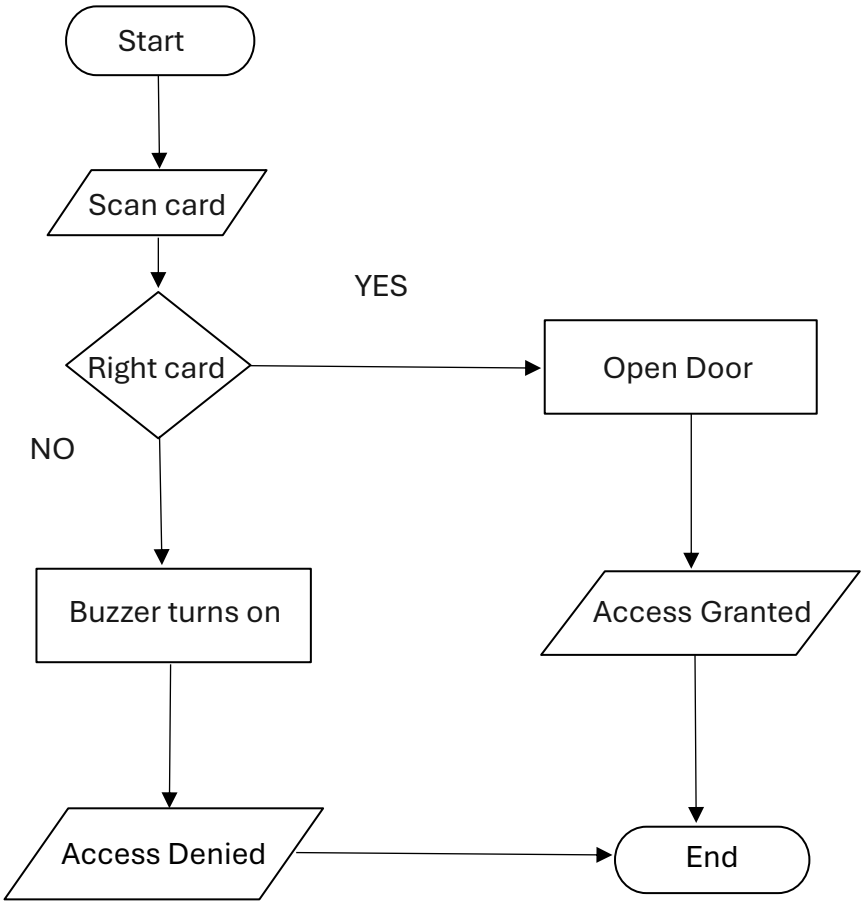
```
#include <SPI.h>
#include <MFRC522.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Pins (ESP32-WROOM)
#define RST_PIN    21     // RC522 Reset (shared with I2C SDA)
#define SS_PIN     5      // RC522 Chip Select (VSPI_SS)
#define BUZZER_PIN 15     // Buzzer
#define LOCK_PIN   12     // H-Bridge IN1
#define UNLOCK_PIN 13     // H-Bridge IN2

MFRC522 mfrc522(SS_PIN, RST_PIN);
LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address (adjust if needed)

// Replace with your RFID tag UIDs
byte authorizedUIDs[][4] = {
  {0x01, 0xA3, 0xB2, 0xC4} // Example UID
};

void setup() {
  Serial.begin(115200);
  SPI.begin();              // Initialize SPI (VSPI by default on ESP32)
  mfrc522.PCD_Init();       // Init RC522
  Wire.begin(21, 22);       // ESP32 I2C (SDA=GPIO21, SCL=GPIO22)
  lcd.init();
  lcd.backlight();
  pinMode(LOCK_PIN, OUTPUT);
  pinMode(UNLOCK_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);
  lcd.print("RFID Door Lock");
  delay(2000);
  lcd.clear();
}
```

```
void loop() {
  lcd.setCursor(0, 0);
  lcd.print("Scan Tag...");

  if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial()) {
    lcd.clear();
    lcd.print("Tag Detected");

    bool isAuthorized = false;
    for (byte i = 0; i < sizeof(authorizedUIDs)/4; i++) {
      if (memcmp(mfrc522.uid.uidByte, authorizedUIDs[i], 4) == 0) {
        isAuthorized = true;
        break;
      }
    }

    if (isAuthorized) {
      lcd.setCursor(0, 1);
      lcd.print("Access Granted!");
      digitalWrite(UNLOCK_PIN, HIGH); // Unlock
      tone(BUZZER_PIN, 1000, 500);
      delay(3000);
      digitalWrite(UNLOCK_PIN, LOW);
    } else {
      lcd.setCursor(0, 1);
      lcd.print("Access Denied!");
      tone(BUZZER_PIN, 300, 1000);
    }
    delay(2000);
    lcd.clear();
    mfrc522.PICC_HaltA();
  }
}
```

4. Schematic:



5. Flowchart:



ESP32 IoT LED Control Project :

1. Hardware Requirements

- **ESP32 WROOM** - Wi-Fi enabled microcontroller

- **LED** – For status indication

- **220Ω Resistor** – For LED current limiting.

- **Jumper Wires** – For prototyping.

- **USB Cable** – For programming and powering ESP32.

2 . Wiring

Component	ESP32 Pin	Notes
Resistor	GPIO 2	One side of resistor to pin 2
LED Anode	Other side of resistor	Long leg (positive)
LED Cathode	GND	Short leg (negative)

3. Code Setup (ESP32-Specific)

3.1 Required Libraries

WiFi.h : Connects ESP32 to your Wi-Fi network.

ArduinoIoTCloud.h : Enables your ESP32 to connect to the Arduino IoT Cloud platform.

Arduino_ConnectionHandler.h: Handles Wi-Fi reconnections and network management.

ThingProperties.h : Auto-generated by Arduino IoT Cloud. Stores cloud variable and callback definitions.

Arduino_secrets.h: Stores your Wi-Fi SSID and password securely.

4-Setting up the Arduino IOT Cloud Enviroment :

Step 1 : I went to https://app.arduino.cc/ and sign in to my account.

Step 2 : Clicked on "Devices", then "Add Device" → Choose "ESP32" → Follow the instructions to connect your board via USB.

Step 3: After successful pairing, the device appeared in my dashboard.

Step 4 : I went to "Things", clicked "Create Thing", and gave it a name

Step 5 : Under Network, selected my ESP32 device.

Step 6: Under Variables, clicked "Add Variable"

Name: Led , Type: Boolean, Variable Permission: Read & Write, Update: On change

Step 7: Linked the variable to the callback onLEDChange().

Step 8: Downloaded the auto-generated files (thingProperties.h) and copy them into your project folder.

Step 9: Uploaded my.ino sketch from Arduino IDE.

Now, the ESP32 connects to the cloud, and I can control the LED remotely via the online dashboard.

3.2 Code Setup ：

MAIN CODE:

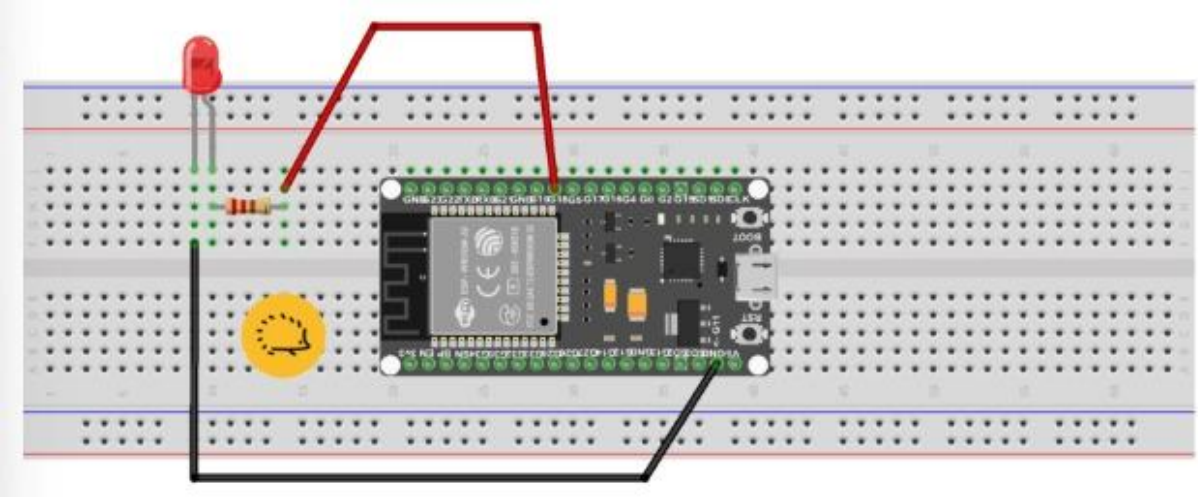
```
#include "thingProperties.h"
#define LED_PIN 2
void setup() {
  Serial.begin(9600);

  pinMode(LED_PIN,OUTPUT);

  delay(500);

  initProperties();

  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  digitalWrite(LED_PIN,LOW);

  setDebugMessageLevel(2);

  ArduinoCloud.printDebugInfo();
}

void loop() {
  ArduinoCloud.update();
}

void onLEDChange()  {
  if (lED==1){
    digitalWrite(LED_PIN,HIGH);
  }

  else{
    digitalWrite(LED_PIN,LOW);
  }
```

THINGPROPERTIES.H CODE - IoT Cloud Configuration and Variable Declaration:

```
#include <ArduinoIoTCloud.h>
#include <Arduino_ConnectionHandler.h>
#include "arduino_secrets.h"
#define SECRET_DEVICE_KEY "O9cq7ibT19gtqYuFUmIvnGMNr"

const char DEVICE_ID[] = "f0991364-27f3-42cc-befb-ac24e54d190e";
const char SSID[] = "WE6E1A08"; #( the network name )
const char PASS[] = "08060e49"; # ( the network password)

bool lED;
void onLEDChange();

void initProperties(){

  ArduinoCloud.setBoardId(DEVICE_ID);

  ArduinoCloud.setSecretDeviceKey(SECRET_DEVICE_KEY);

  ArduinoCloud.addProperty(lED,READWRITE,ON_CHANGE,onLEDChange);

}

WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS);
```

4. Schematic :

To visualize and test the circuit before physical implementation



Component Connections:

ESP32 GPIO2 (D2) → Connected to one end of a 220Ω resistor

The other end of the resistor → Connected to the anode (long leg) of the LED

The cathode (short leg) of the LED → Connected to GND on the breadboard

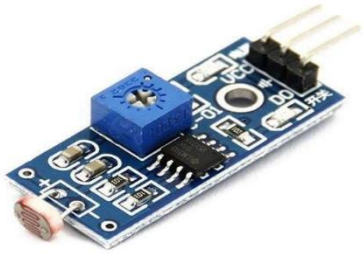
ESP32 GND pin → Connected to the breadboard's ground rail

Light Sensor Module User Manual

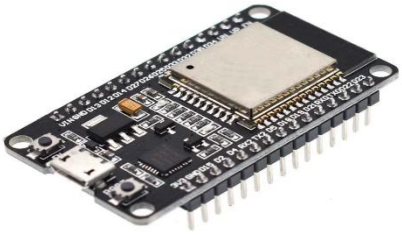
Section: ESP32 Setup Guide

Hardware Requirements

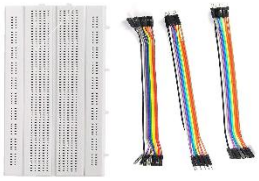
- **Light Sensor Module** (e.g., photoresistor/photodiode-based module like KY-018 or TEMT6000)



- **ESP32** Development Board



- **Breadboard & Jumper Wires**



1) **USB Cable** (for ESP32 programming and power)
2) **Computer with Arduino IDE or Platform IO**

Wiring Diagram: Light Sensor Module to ESP32

Light Sensor Pin	Connects To (ESP32)	Description
GND	GND	Ground
VCC	3.3V	Power supply
OUT	GPIO34 (or any ADC pin)	Analog signal to ESP32

3. Code Setup (ESP32-Specific)

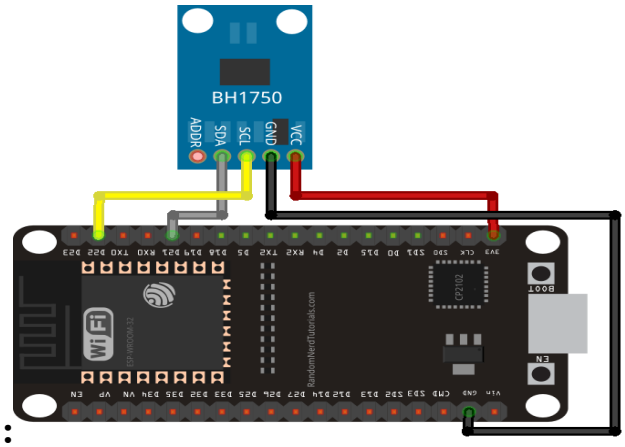
3.1 Required Libraries

1. **No external library required** for basic analog input from the light sensor.
2. **(Optional)** `LiquidCrystal_I2C` – if you want to display light levels on an LCD.
3. **Arduino.h** – Built-in with the ESP32 core.
4.

Code:

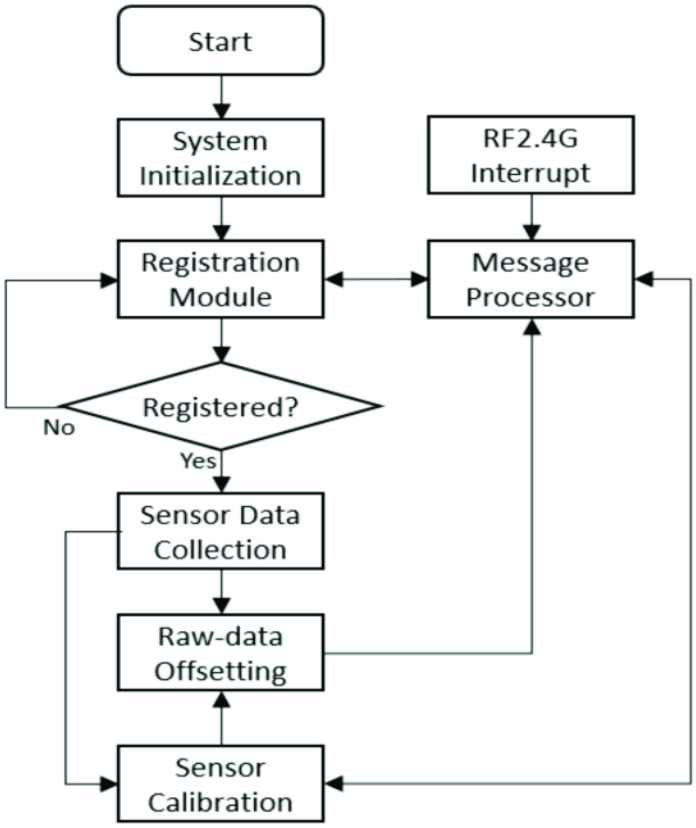
```
// Define analog pin connected to sensor
const int lightSensorPin = 34; // GPIO34 is an ADC pin on ESP32

void setup() {
  Serial.begin(115200);  // Start Serial Monitor
  analogReadResolution(12); // Optional: Set ADC resolution (ESP32 default is 12 bits)
}

void loop() {
  int lightValue = analogRead(lightSensorPin); // Read analog value (0 - 4095)
  Serial.print("Light Level: ");
  Serial.println(lightValue);
  delay(500); // Wait half a second
}
```

Schematic:



:

Flowchart:



Motion Sensor Module User Manual

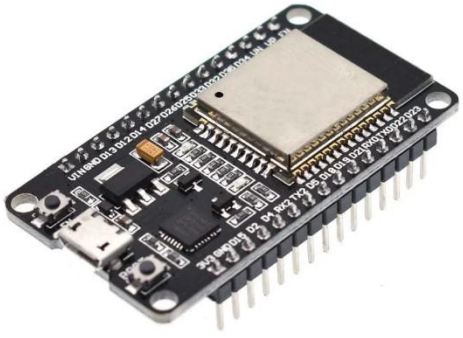
Section: ESP32 Setup Guide

Hardware Requirements

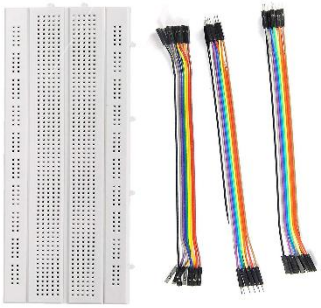
PIR (Passive Infrared) Motion Sensor: The most common type of motion sensor.

- **HC-SR501**: A widely used and affordable PIR sensor module that typically includes a lens, IR sensor, and supporting circuitry.

- **ESP32** Development Board



- **Breadboard & Jumper Wires**



- **USB Cable** (for ESP32 programming and power)
- **Computer with Arduino IDE or Platform IO**

Wiring Diagram: Light Sensor Module to ESP32

Light Sensor Pin	Connects To (ESP32)	Description
GND	GND	Ground
VCC	3.3V	Power supply
OUT	GPIO34 (or any ADC pin)	Analog signal to ESP32

3. Code Setup (ESP32-Specific)

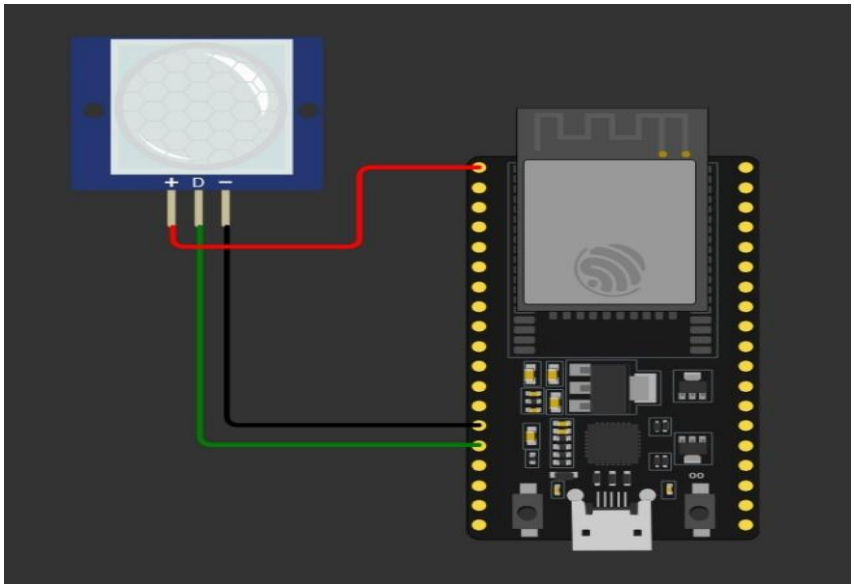
3.1 Required Libraries

No external library required for basic analog input from the light sensor.
(Optional) `LiquidCrystal_I2C` – if you want to display light levels on an LCD.
Arduino.h – Built-in with the ESP32 core.

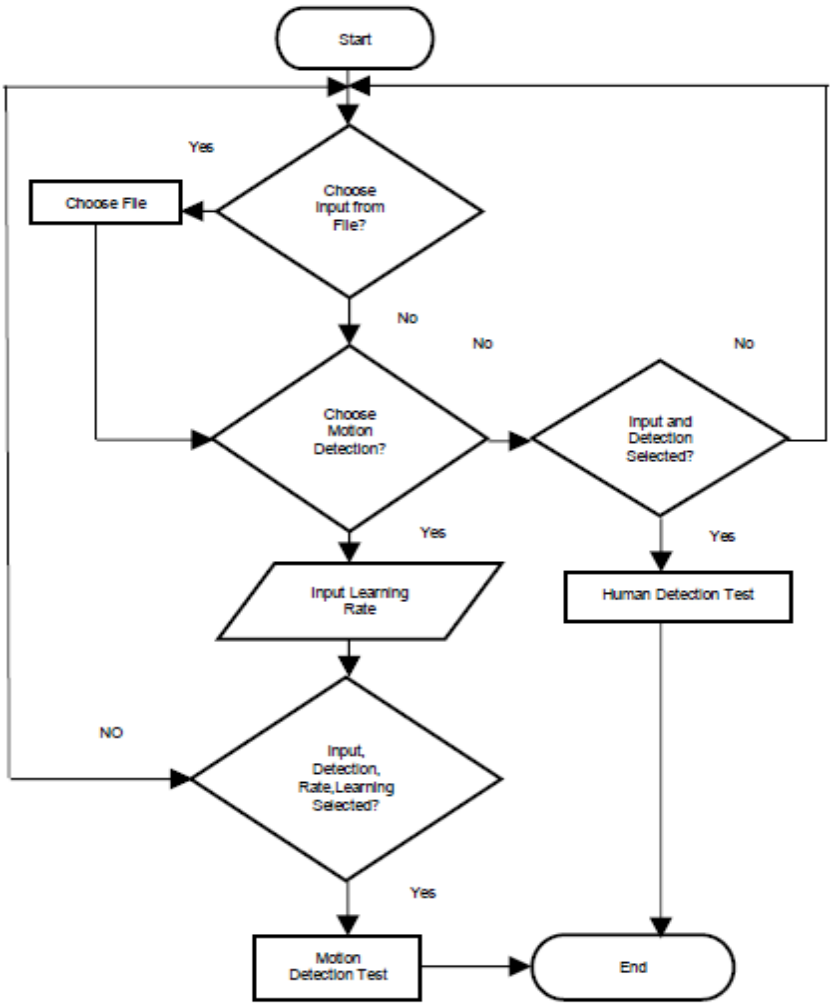
Code:

```
const int PIN_TO_SENSOR = 19; // GPIO19 pin connected to OUTPUT pin of sensor
int pinStateCurrent   = LOW;  // current state of pin
int pinStatePrevious  = LOW;  // previous state of pin

void setup() {
  Serial.begin(9600);           // initialize serial
  pinMode(PIN_TO_SENSOR, INPUT); // set ESP32 pin to input mode to read value from OUTPUT pin of sensor
}

void loop() {
  pinStatePrevious = pinStateCurrent; // store old state
  pinStateCurrent = digitalRead(PIN_TO_SENSOR);   // read new state

  if (pinStatePrevious == LOW && pinStateCurrent == HIGH) {   // pin state change: LOW -> HIGH
    Serial.println("Motion detected!");
    // TODO: turn on alarm, light or activate a device ... here
  }
  else
  if (pinStatePrevious == HIGH && pinStateCurrent == LOW) {   // pin state change: HIGH -> LOW
    Serial.println("Motion stopped!");
    // TODO: turn off alarm, light or deactivate a device ... here
  }
```

Schematic:



Flowchart:


Start
Yes
Choose File
Choose Input from File?
No
No
No
Choose Motion Detection?
Input and Detection Selected?
Yes
Yes
Input Learning Rate
Human Detection Test
NO
Input, Detection, Rate,Learning Selected?
Yes
Motion Detection Test
End