Environment Light Intensity Meter

Session: 2022 – 2026

Submitted by:

Mian Muhammad Ashhad 2022-CS-114

Muhammad Saad Akmal 2022-CS-148

Robass Atif 2022-CS-150

Mohammad Abdullah 2022-CS-155

Submitted to:

Sir Tehseen-ul-Hassan Shah

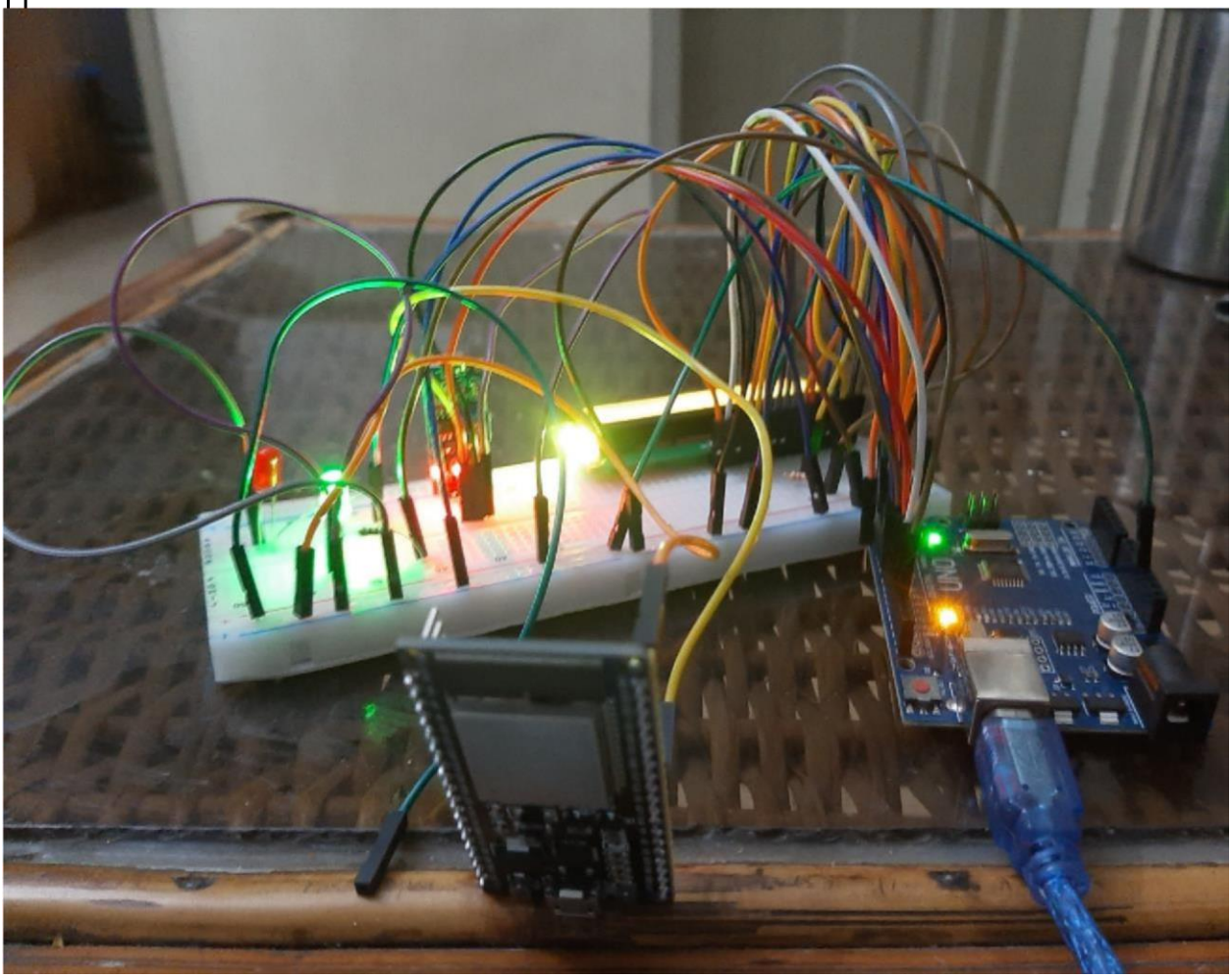
Department of Computer Science

University of Engineering and Technology Lahore

Pakistan

Contents

Project Photo



Description

The Environment Light Intensity Meter project utilizes a light sensor connected to a microcontroller (e.g., Arduino or Raspberry Pi) for measuring ambient light conditions. The collected light intensity data is then transmitted to three platforms:

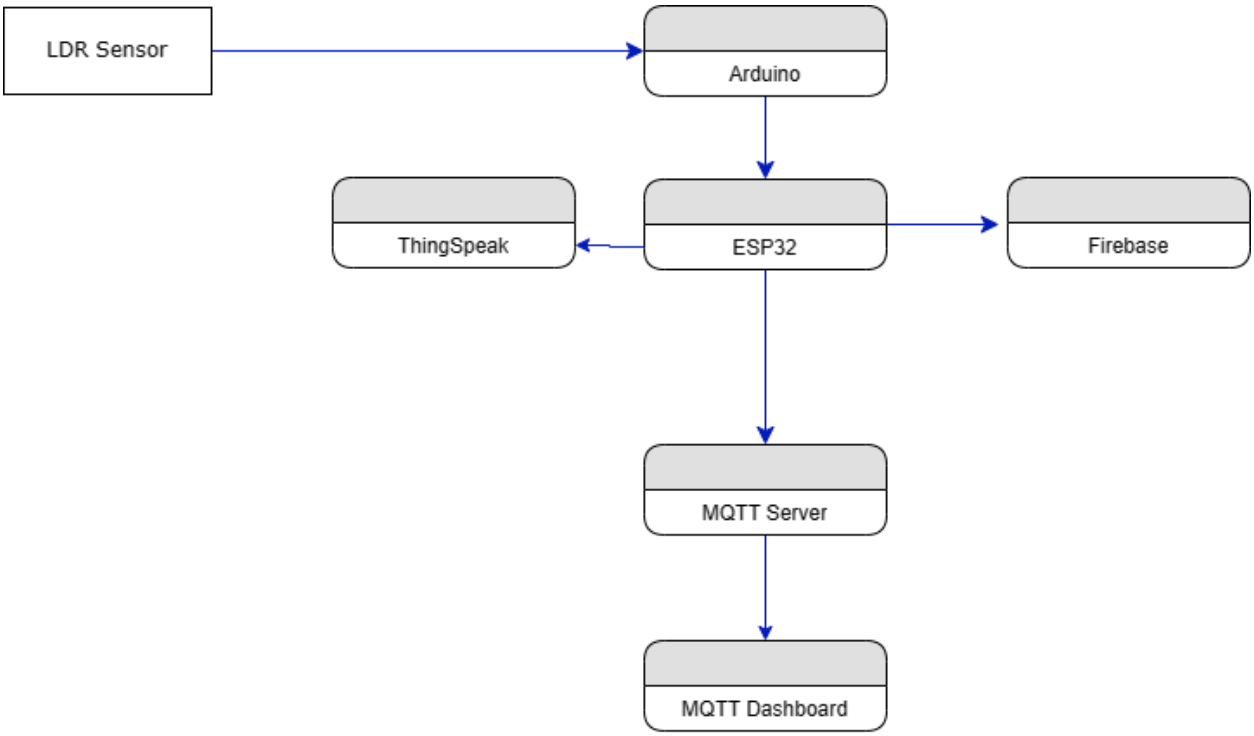
- MQTT (Message Queuing Telemetry Transport): Enables real-time communication for receiving live light intensity updates.
- Firebase: Stores historical light intensity data securely in a cloud database for long-term analysis.
- ThingSpeak: Provides a platform for visualizing and analyzing light intensity trends through charts and graphs

This integrated system ensures a comprehensive approach to environmental light monitoring, offering realtime insights and historical data analysis through different communication and storage channels.

Methodology Used

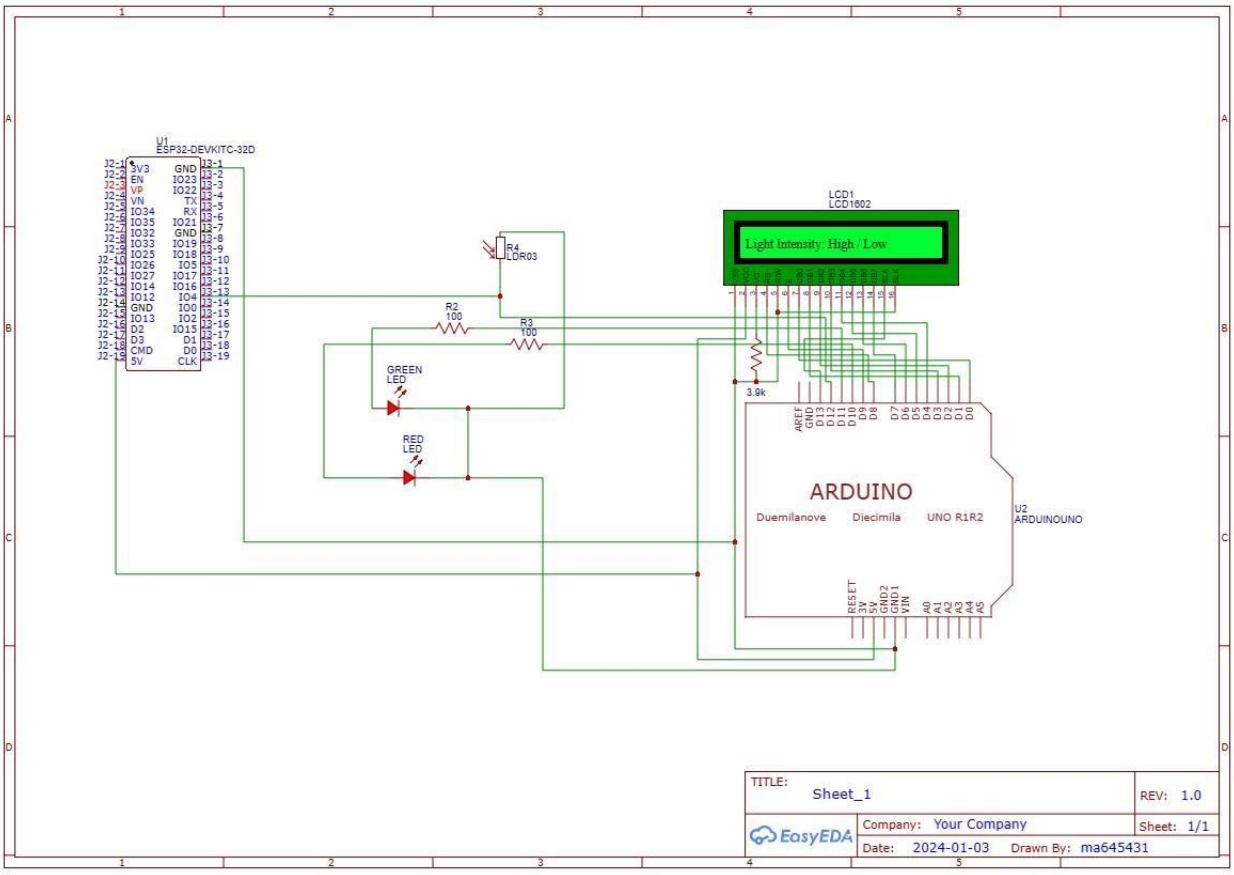
AVR Module(ESP32)

Data Flow Diagram



ESP32, MQTT, and ThingSpeak for comprehensive environmental monitoring. Arduino and ESP32 microcontrollers serve as the project's backbone, interfacing with a light sensor to measure

Circuit Diagram



Explanation

The Environment Light Intensity Meter project employs an integrated system utilizing Arduino, ambient light conditions. The Arduino provides a versatile and user-friendly platform, while the ESP32 enhances connectivity with its advanced features, ensuring efficient data transmission.

The collected light intensity data undergoes a dual transmission process. First, the MQTT (Message Queuing Telemetry Transport) protocol enables real-time communication, facilitating the prompt reception of live light intensity updates. This aspect is crucial for addressing immediate changes in environmental lighting conditions.

Simultaneously, the data is transmitted to ThingSpeak, a platform that excels in visualization and analysis. ThingSpeak transforms the raw data into intuitive charts and graphs, allowing users to easily comprehend and interpret light intensity trends over time. This visual representation enhances the accessibility and usability of the data, making it valuable for both experts and enthusiasts interested in studying environmental variations.

The combined use of Arduino and ESP32, along with the integration of MQTT and ThingSpeak, results in a robust and versatile environmental monitoring system. This approach ensures real-time

insights and facilitates in-depth historical analysis through different communication and visualization platforms.

AVR module code

```
.include "m328pdef.inc"

.include "delay.inc"

.include "1602_LCD.inc"

.cseg

.org 0x00


; Setting pins to Output for LCD

sbi DDRD,PD0 ; D0 pin of LCD sbi

DDRD,PD1 ; D1 sbi


DDRD,PD2 ; D2 sbi DDRD,PD3 ;

D3 sbi DDRD,PD4 ; D4 sbi

DDRD,PD5 ; D5 sbi DDRD,PD6 ;

D6 sbi DDRD,PD7 ; D7


;Setting LCD Mode selection pins

sbi DDRB,PB0 ; RS sbi DDRB,PB1 ;

E pin of LCD ;Setting LCD Backlight

pin sbi DDRB,PB5 ; BLA pin of LCD


; LCD Init

LCD_send_a_command 0x01 ; sending all clear command

LCD_send_a_command 0x38 ; set LCD mode to 16*2 line LCD

LCD_send_a_command 0x0C ; screen ON sbi PORTB,PB5
```

```
SBI DDRB, PB3    ; PB3 set as OUTPUT Pin
CBI PORTB, PB3   ; LED OFF

SBI DDRB, PB2    ; PB2 set as OUTPUT Pin
CBI PORTB, PB2   ; LED OFF

CBI DDRB, PB4    ; PB4 set as INPUT pin
SBI PORTB, PB4   ; Enable internal pull-up resistor

loop:
; check if push button is pressed
SBIS PINB, PB4   ; if not pressed, skip next line if the PINB reg. bit# 4 is 1 rjmp
L1

; PB4 is pressed
CBI PORTB, PB3   ; LED OFF SBI PORTB,
PB2   ; LED ON

rjmp RED_ON_LCD
delay 500 rjmp loop

L1:
; PB4 is not pressed
SBI PORTB, PB3   ; LED ON CBI
```

```
PORTB, PB2   ; LED OFF rjmp

GREEN_ON_LCD
rjmp loop


RED_ON_LCD:
LCD_send_a_character 0x4C ; 'L'

LCD_send_a_character 0x49 ; 'I'

LCD_send_a_character 0x47 ; 'G' LCD_send_a_character

0x48 ; 'H'


LCD_send_a_character 0x54 ; 'T'

LCD_send_a_character 0x20 ; ' ' (space)

LCD_send_a_character 0x49 ; 'I'

LCD_send_a_character 0x4E ; 'N'

LCD_send_a_character 0x54 ; 'T'

LCD_send_a_character 0x45 ; 'E'

LCD_send_a_character 0x4E ; 'N'

LCD_send_a_character 0x53 ; 'S'

LCD_send_a_character 0x49 ; 'I'

LCD_send_a_character 0x54 ; 'T'

LCD_send_a_character 0x59 ; 'Y'

LCD_send_a_command 0xC0 ; move curser to next line

LCD_send_a_character 0x20 ; ' ' (space)

LCD_send_a_character 0x20 ; ' ' (space)

LCD_send_a_character 0x20 ; ' ' (space)

LCD_send_a_character 0x20 ; ' ' (space)
```

```
LCD_send_a_character 0x20 ; ' ' (space)

LCD_send_a_character 0x20 ; ' ' (space) LCD_send_a_character

0x20 ; ' ' (space)

LCD_send_a_character 0x4C ; 'L'

LCD_send_a_character 0x4F ; 'A' LCD_send_a_character

0x57 ; 'B'  reti


GREEN_ON_LCD:

LCD_send_a_character 0x4C ; 'L'

LCD_send_a_character 0x49 ; 'I'

LCD_send_a_character 0x47 ; 'G' LCD_send_a_character

0x48 ; 'H'

LCD_send_a_character 0x54 ; 'T'

LCD_send_a_character 0x20 ; ' ' (space)

LCD_send_a_character 0x49 ; 'I'

LCD_send_a_character 0x4E ; 'N'

LCD_send_a_character 0x54 ; 'T'

LCD_send_a_character 0x45 ; 'E'

LCD_send_a_character 0x4E ; 'N'

LCD_send_a_character 0x53 ; 'S'

LCD_send_a_character 0x49 ; 'I'

LCD_send_a_character 0x54 ; 'T'

LCD_send_a_character 0x59 ; 'Y'

LCD_send_a_command 0xC0 ; move curser to next line
```

LCD_send_a_character 0x20 ; ' ' (space)

LCD_send_a_character 0x20 ; ' ' (space)

LCD_send_a_character 0x20 ; ' ' (space) LCD_send_a_character 0x20 ; ' ' (space)

LCD_send_a_character 0x20 ; ' ' (space) LCD_send_a_character 0x20

; ' ' (space)

LCD_send_a_character 0x48 ; 'H'

LCD_send_a_character 0x49 ; 'I'

LCD_send_a_character 0x47 ; 'G' LCD_send_a_character

0x48 ; 'H' ret

IoT module code

```
#include <WiFi.h>

#include <PubSubClient.h>

#include <ESP32Firebase.h>

#include "ThingSpeak.h"


WiFiClient wifiClient;

PubSubClient client;  // Pass the WiFi client to PubSubClient constructor

Firebase firebase("https://pelagic-cocoa-382522-default-rtdb.asia-southeast1.firebasedatabase.app/");


unsigned long myChannelNumber = 2; const char * myWriteAPIKey

= "CO6KKYSWDKXSQW11";
```

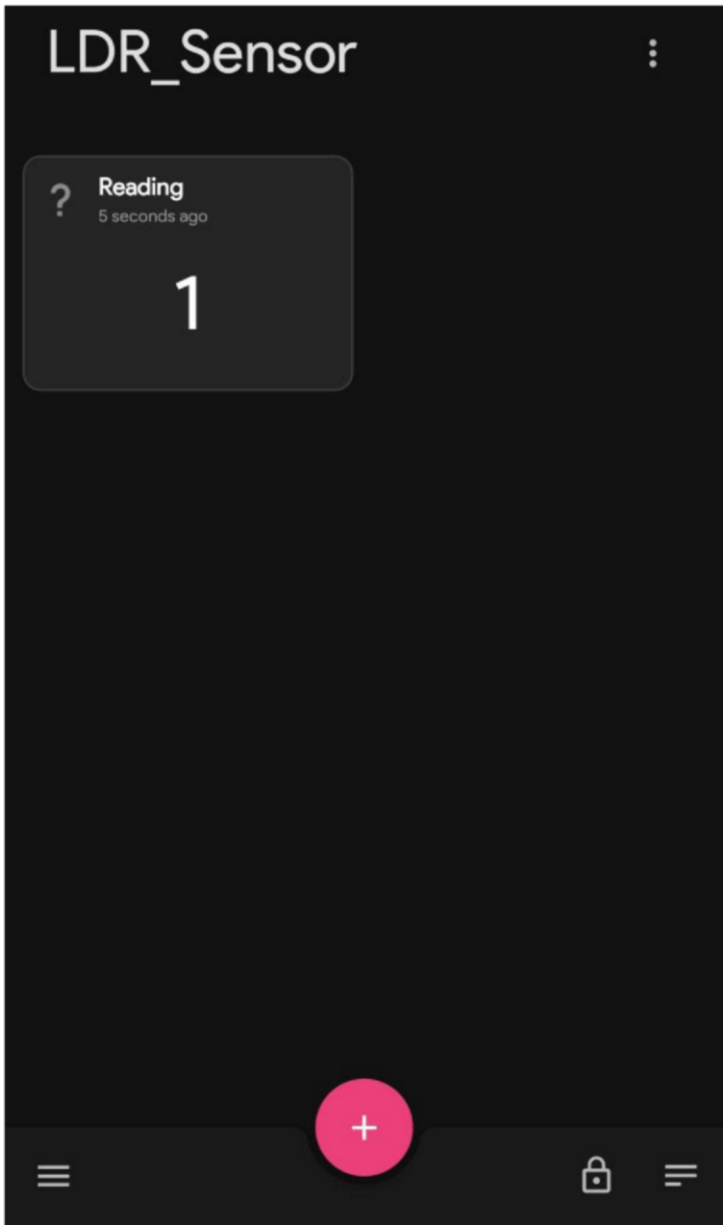
```
const int LDRPin = 4; const char
*ssid = "SAAD"; const char
*password = "atzn2722";

void setup() {   Serial.begin(9600);
pinMode(LDRPin, INPUT);


 /*WiFi.mode(WIFI_STA);
WiFi.disconnect();   delay(1000);*/


 // Connect to WiFi   WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {

   delay(1000);
   Serial.println("Connecting to WiFi...");
 }
 Serial.println("Connected to WiFi");

 // Connect to MQTT broker
 //connectToMQTT();

 ThingSpeak.begin(wifiClient);
}

void loop() {
```

```
 // put your main code here, to run repeatedly:
 /*if (!client.connected()) {
   // Reconnect to MQTT broker if connection is lost     connectToMQTT();
 }*/

 int ldrValue = digitalRead(LDRPin);
 Serial.print("LDR Value: ");
 Serial.println(ldrValue);

 // Convert ldrValue to char array
char buffer[10];       itoa(ldrValue,
buffer, 10);


 // Publish LDR value to MQTT
 /*if (client.publish("ashhad", buffer)) {
   Serial.println("Message sent successfully");
 } else {
   Serial.println("Failed to send message");
 }*/
 sendToTS(ldrValue);   firebase.pushInt("Intensity",
ldrValue);   Serial.println("Value sent to Firebase");
delay(1000);

}
```

```
void sendToTS(int ldrValue) {   int x = ThingSpeak.writeField(myChannelNumber,
1, ldrValue, myWriteAPIKey);
if(x == 200)

 {
  Serial.println("Channel update successful.");
 }
 else
 {
  Serial.println("Problem updating channel. HTTP error code " + String(x));
 }
}

void connectToMQTT() {

 client.setClient(wifiClient);    client.setServer("test.mosquitto.org",1883);
while
(!client.connected()) {
  Serial.println("Attempting MQTT connection...");    if
(client.connect("b14d6440-8a03-46b3-963a-d3384e54e6d8")) {
    Serial.println("Connected to MQTT broker");
  } else {
    Serial.print("MQTT connection failed, rc=");
    Serial.print(client.state());      Serial.println("
Retrying in 1 second...");      delay(1000);
```

```
  }
 }
}
```

MQTT dashboard Screenshot



Project video Link

- https://youtu.be/JrU5knfI4E2

- https://www.linkedin.com/feed/update/urn:li:activity:7148576675193143296/

Github Link

https://github.com/Ashhad-Mazhar/AVR-IoT-Project

Reference

- https://github.com/Rupakpoddar/ESP32Firebase
- https://github.com/hideakitai/MQTTPubSubClient