 ***University of Engineering***

***& Technology Lahore***

**Coal Final Project Report**

**Project Name: “**Environmental Light Intensity Meter”

**Group Members:**

Affan Ahmad(2022-CS-178)

Areej Fatima(2022-CS-177)

Ramsha Shabbir(2022-CS-205)

Hamna Kamran(2022-CS-210)

**Submitted to:**

Sir Tehseen

**Description:**

The Environmental Light Intensity Meter incorporates a strategic selection of components to achieve its functionality. The heart of the system is an Arduino Uno microcontroller, serving as the central processing unit. To enhance connectivity and enable IOT capabilities, an ESP32 module is integrated, allowing for remote monitoring and data transmission.

The project utilizes a Light Dependent Resistor (LDR) as the primary sensor for accurate light intensity measurements. The data is then visually presented through a 16x2 LCD display, providing a clear and concise readout. LEDs and resistors are employed in the circuit to indicate operational status and aid in calibration, ensuring the reliability of the meter under varying environmental conditions.

**Methodology:**

**Project Planning:**

Objective: Accurate Light Measurement: Achieve a high level of precision in measuring environmental light intensity using the Light Dependent Resistor (LDR) sensor.

Real-Time Display: Develop a responsive real-time display on the 16x2 LCD screen for immediate feedback on current light intensity levels.

Remote Monitoring: Enable remote monitoring through ESP32 integration, allowing users to access and analyze light intensity data over the internet.

User-Friendly Interface: Design an intuitive interface for easy interpretation of light intensity readings, catering to users with varying technical backgrounds.

**Resource Allocation:**

**Hardware Components:**

Arduino Uno (Main microcontroller).

ESP32 module (Internet connectivity).

Light Dependent Resistor (LDR) sensor.

16x2 LCD screen.

LEDs and resistors.

**Programming Tools:**

Arduino IDE for microcontroller programming.

ESP-IDF for ESP32 programming.

**Prototyping and Testing:**

Breadboard, jumper wires, and soldering tools. Multimeter for circuit testing.

**Component Selection:**

Choose the Arduino Uno as the main microcontroller for processing data. Incorporate the ESP32 for internet connectivity and remote monitoring capabilities. Select a Light Dependent Resistor (LDR) to measure light intensity accurately. Include a 16x2 LCD screen for displaying real-time data. Integrate LEDs and resistors for status indicators and calibration support.

**Assembly and Wiring:**

Assemble the hardware components on a prototyping board. Connect the Arduino Uno, ESP32, LDR, LCD, LEDs, and resistors according to the circuit design. Double-check wiring to avoid short circuits and ensure proper functioning.

**Programming:**

Write assembly language code for the Arduino Uno to read data from the LDR. Develop code to process and display the light intensity on the 16x2 LCD. Incorporate code for LED indicators and calibration procedures. Program the ESP32 for internet connectivity and remote data transmission.

**Testing:**

Conduct initial tests to verify the accuracy of light intensity measurements. Check the functionality of the LCD display, LEDs, and calibration features. Perform integration tests to ensure proper communication between components.Test the ESP32's connectivity for remote monitoring.

**Working of Components:** **Arduino Uno:**

**Role:** The Arduino Uno serves as the central microcontroller, responsible for processing data from the Light Dependent Resistor (LDR) and controlling the display and indicator components.

**Working**: The Arduino Uno reads analog signals from the LDR, processes the data using assembly language code, and sends commands to other components for display and feedback.

**ESP32:**

**Role:** The ESP32 provides internet connectivity, enabling remote monitoring of light intensity data.

**Working:** Programmed with appropriate firmware, the ESP32 connects to a Wi-Fi network, allowing users to access real-time light intensity information from a remote location through a web interface or other means.

**Light Dependent Resistor (LDR):**

**Role:** The LDR is the sensor responsible for detecting ambient light intensity.

**Working:** The resistance of the LDR changes with varying light levels. The Arduino Uno reads the analog voltage across the LDR to determine the light intensity, converting this information into a readable format for display.

**LEDs:**

**Role:** LEDs provide visual indicators for the operational status of the Environmental Light Intensity Meter.

**Working:** LEDs are programmed to light up or blink in specific patterns to convey information, such as power-on status, successful calibration, or connectivity with the internet.

**16x2 LCD:**

**Role:** The 16x2 LCD serves as the display interface, presenting real-time light intensity readings to the user.

**Working:** The Arduino Uno sends the processed light intensity data to the LCD, where it is displayed in a clear and readable format. Users can instantly observe the light levels in their environment.

**Resistors:**

**Role:** Resistors are used in the circuit for various purposes, such as current limiting and calibration.

**Working:** In the context of LEDs, resistors are often employed to control the current flowing through the LEDs, preventing damage. Additionally, resistors may be used in calibration circuits to fine-tune the response of the LDR.

**Code:**

; This example program displays different types of data on a 16x2 LCD.

;

; Data Pin Connections for 16x2 LCD

; [LCD pins] [Arduino UNO Pins]

; RS --------------- 8 (PB0)

; E --------------- 9 (PB1)

; D4 --------------- 4 (PD4)

; D5 --------------- 5 (PD5)

; D6 --------------- 6 (PD6)

; D7 --------------- 7 (PD7)

; A --------------- 13 (PB5) ; Anode pin of LCD Backlight LED

.include "m328pdef.inc"

.include "delay\_Macro.inc"

.include "1602\_LCD\_Macros.inc"

.include "UART\_Macros.inc"

.cseg

.org 0x0000

SBI DDRB, PB3 ; Set PB3 pin for Output to LED

CBI PORTB, PB3 ; LED OFF

SBI DDRB, PB4 ; Set PB4 pin for Output to LED

CBI PORTB, PB4 ; LED OFF

.def A = r16 ; Rename or attach a label to the register

.def AH = r17

; ADC config

LDI A, 0b11000111 ; [ADEN ADSC ADATE ADIF ADIE ADIE ADPS2 ADPS1 ADPS0]

STS ADCSRA, A

LDI A, 0b01100000 ; [REFS1 REFS0 ADLAR – MUX3 MUX2 MUX1 MUX0]

STS ADMUX, A ; Select ADC0 (PC0) pin

SBI PORTC, PC0 ; Enable Pull-up Resistor

Serial\_begin

LCD\_init ; Initialize the 16x2 LCD

LCD\_backlight\_OFF ; Turn Off the LCD Backlight delay 1000

LCD\_backlight\_ON ; Turn On the LCD Backlight delay 300

; Initialize variables for strings and lengths

LDI ZL, LOW (2 \* dark\_string)

LDI ZH, HIGH (2 \* dark\_string)

LDI R20, DarkString\_len LCD\_send\_a\_string delay 500

loop:

LDS A, ADCSRA ; Start Analog to Digital Conversion

ORI A, (1<<ADSC)

STS ADCSRA, A

wait:

LDS A, ADCSRA ; Wait for ADC conversion to complete sbrc A, ADSC

rjmp wait

LDS A, ADCL ; Must Read ADCL before ADCH LDS AH, ADCH

delay 100 ; Delay 100ms

cpi AH, 200 ; Compare LDR reading with the desired threshold value (e.g., 200) brlo LED\_OFF ; Jump if AH < 200

SBI PORTB, PB4 ; LED ON CBI PORTB, PB3 ; LED OFF rjmp LED\_ON\_End ; Jump to skip LED\_OFF condition

LED\_OFF:

CBI PORTB, PB4 ; LED OFF

SBI PORTB, PB3 ; LED ON

LED\_ON\_End:

; LCD Configuration

LCD\_send\_a\_command 0x80

delay 20

; Scroll text on LCD

LCD\_send\_a\_command 0x14

LCD\_send\_a\_command 0x14

LCD\_send\_a\_command 0x14

LCD\_send\_a\_command 0x14

LCD\_send\_a\_command 0x14

LCD\_send\_a\_command 0x14

LCD\_send\_a\_command 0x14

LCD\_send\_a\_command 0x14

LCD\_send\_a\_command 0x14

; Send data to UART and LCD

Serial\_writeReg AH ; Send register data to UART delay 50

LCD\_send\_a\_register AH ; Send register data to LCD delay 50

; Display spaces on LCD

LDI ZL, LOW (2 \* spaces)

LDI ZH, HIGH (2 \* spaces)

LDI R20, SpacesString\_len LCD\_send\_a\_string delay 50

rjmp loop

dark\_string: .db "Darkness:", 0 spaces: .db " ", 0

; Calculate string lengths

len: .equ DarkString\_len = (2 \* (len - dark\_string)) - 1 Spacelen: .equ SpacesString\_len = (2 \* (len - spaces)) - 1

; It is recommended to define constants (arrays, strings, etc.) at the end of the code segment

; .db directive is used to declare constants

; The length of the string must be an even number of bytes

**Code esp32**

#include <WiFi.h>

#include <PubSubClient.h>

#include <Firebase\_ESP\_Client.h>

#include "addons/TokenHelper.h"

#include "addons/RTDBHelper.h"

#include "ThingSpeak.h"

// Wi-Fi credentials

const char \*ssid = "Galaxy";

const char \*password = "fzvg0533";

// MQTT configuration

const char \*mqttServer = "broker.hivemq.com";

const int mqttPort = 1883;

const char \*mqttClientId = "2022-CS-178";

const char \*outputTopic = "esp32/output";

WiFiClient espClient;

// ThingSpeak configuration

unsigned long myChannelNumber = 2391804;

const char \*myWriteAPIKey = "T7K0BZL98517507L";

String myStatus = "";

// Firebase configuration

FirebaseData fbdo;

FirebaseAuth auth;

FirebaseConfig config;

bool signupOK = false;

static unsigned long lastMillis = 0;

PubSubClient client(espClient);

void setup()

{

Serial.begin(115200);

Serial2.begin(9600);

WifiSetup();

// configure cloud services

fireBaseSetup();

ThingSpeak.begin(espClient);

ThingSpeak.setStatus(myStatus);

// Configure MQTT

client.setServer(mqttServer, mqttPort);

client.setCallback(callBack);

connectToMQTT();

}

void WifiSetup()

{

// Connect to Wi-Fi

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

// keep retrying until successful connection

while (WiFi.status() != WL\_CONNECTED)

{

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

}

void fireBaseSetup()

{

// configure firebase

config.api\_key = "AIzaSyBUr\_U4sLQYtNtjkn\_j0LPPXv0IVaCSu08";

config.database\_url = "https://ldr-mqtt-esp-default-rtdb.asia-southeast1.firebasedatabase.app/";

// try to connect

if (Firebase.signUp(&config, &auth, "", ""))

{

Serial.println("ok");

signupOK = true;

}

else

{

Serial.printf("%s\n", config.signer.signupError.message.c\_str());

}

config.token\_status\_callback = tokenStatusCallback;

Firebase.begin(&config, &auth);

Firebase.reconnectWiFi(true);

}

void callBack(char \*inputTopic, byte \*message, unsigned int length)

{

// Callback function for MQTT messages

Serial.print("Message arrived on topic: ");

Serial.print(inputTopic);

Serial.print(". Message: ");

String messageTemp;

for (int i = 0; i < length; i++)

{

Serial.print((char)message[i]);

messageTemp += (char)message[i];

}

Serial.println();

}

void loop()

{

int data = -1; //-1 for data=not read

// reconnect to wifi if disconnected

if (WiFi.status() != WL\_CONNECTED)

{

WifiSetup();

}

// Handle MQTT events

if (client.connected() == false)

{

connectToMQTT();

}

client.loop();

// read data from arduino

if (Serial2.available() > 0)

{

data = Serial2.read();

Serial.println(data);

// clear the buffer in case there are too many messages

while (Serial2.available())

{

Serial2.read();

}

}

// Publish a message every 5 seconds

if (millis() - lastMillis > 1000 && data != -1)

{

lastMillis = millis();

std::string str = std::to\_string(data);

std::string msg = "Light Intensity: " + str;

const char \*message = str.c\_str();

publishMessage(message);

Serial.println(message);

// publish data to cloud

ThingSpeakPublishData(1, data);

firebasePublishData("test/int", data);

}

delay(100);

}

void ThingSpeakPublishData(int field, int data)

{

// Publish data to ThingSpeak

ThingSpeak.setField(1, data);

int httpCode = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);

if (httpCode == 200)

{

Serial.println("Channel update successful.");

}

else

{

Serial.println("Problem updating channel. HTTP error code " + String(httpCode));

}

}

void firebasePublishData(String address, int data)

{

// Publish data to Firebase

if (Firebase.ready() && signupOK)

{

if (Firebase.RTDB.setInt(&fbdo, address, data))

{

Serial.println("PASSED");

Serial.println("PATH: " + fbdo.dataPath());

Serial.println("TYPE: " + fbdo.dataType());

}

else

{

Serial.println("FAILED");

Serial.println("REASON: " + fbdo.errorReason());

}

}

}

void connectToMQTT()

{

while (!client.connected())

{

Serial.println("Connecting to MQTT...");

if (client.connect(mqttClientId))

{

Serial.println("Connected to MQTT");

}

else

{

Serial.print("Failed with state ");

Serial.print(client.state());

delay(2000);

}

}

}

void publishMessage(const char \*message)

{

if (client.connected())

{

client.publish(outputTopic, message);

Serial.println("Message Published");

}

}

**Documentation:**

**16x2 LCD Module Code Documentation**

**Overview:**

This AVR assembly code demonstrates how to use a 16x2 LCD with an Arduino UNO. It displays various types of information on the LCD and controls an LED based on analog sensor readings. Additionally, the code includes UART communication for sending data to ESP.

**Pin Configuration:**

* LCD Pins to Arduino UNO Pins:
* RS --------------- 8 (PB0)
* E --------------- 9 (PB1)
* D4 --------------- 4 (PD4)
* D5 --------------- 5 (PD5)
* D6 --------------- 6 (PD6)
* D7 --------------- 7 (PD7)
* A --------------- 13 (PB5) ; Anode pin of LCD Backlight LED

**Included Files:**

* m328pdef.inc: This file defines the ATmega328P microcontroller.
* delay\_Macro.inc: Contains macros for implementing delays.
* 1602\_LCD\_Macros.inc: Includes macros for controlling a 16x2 LCD.
* UART\_Macros.inc: Encompasses macros relevant to UART communication.

**Global Constants:**

* LED Pins: PB3 (LED1) and PB4 (LED2).
* ADC Configuration: Holds registers and configurations for analog-to-digital conversion (ADC).
* String Constants: "Darkness:" and " " (three spaces).

**Main Functions:**

**setup():**

* Initializes LED pins and ensures LEDs are turned off initially.
* Configures ADC for precise analog readings.
* Establishes serial communication.
* Initializes and configures the 16x2 LCD.

**loop():**

* Reads analog sensor data (from LDR) through ADC.- Manages LEDs based on the sensor readings.
* Sends light intensity values to the serial port.

**Additional Functions:** delay(): - A macro designed for introducing delays in milliseconds.

**Documentation:**

**ESP32 IoT Project Documentation**

**Overview:**

This code is designed for an ESP32 microcontroller to enable IoT functionalities. It integrates with Wi-Fi, MQTT, ThingSpeak, and Firebase services to facilitate communication and data storage. The primary goal is to read data from an Arduino device connected via Serial2 and publish it to both ThingSpeak and Firebase, while also providing real-time updates via MQTT.

**Dependencies:**

WiFi.h: Library for connecting to Wi-Fi networks.

PubSubClient.h: MQTT library for ESP32.

Firebase\_ESP\_Client.h: Library for interfacing with Firebase Realtime Database.

ThingSpeak.h: Library for sending data to ThingSpeak IoT platform.

Configuration

**Wi-Fi Credentials:**

SSID: Galaxy

Password: fzvg0533

MQTT Configuration

**Broker:** broker.hivemq.com

Port: 1883

Client ID: 2022-CS-178

Output Topic: esp32/output

**ThingSpeak Configuration:**

Channel Number: 2391804

Write API Key: T7K0BZL98517507L

**Firebase Configuration:**

API Key: AIzaSyBUr\_U4sLQYtNtjkn\_j0LPPXv0IVaCSu08

Database URL: <https://ldr-mqtt-esp-default-rtdb.asia-southeast1.firebasedatabase.app/>

**Setup**

Initialize serial communication at a baud rate of 115200.

Begin Serial2 communication at 9600 baud for Arduino data transfer.

Configure Wi-Fi using the provided credentials.

Setup and configure Firebase and ThingSpeak services.

Configure MQTT, set the callback function, and establish a connection.

**Main Functions**

WifiSetup()

Connects to the specified Wi-Fi network.

Prints the IP address upon successful connection.

**fireBaseSetup()**

Configures Firebase with the provided API key and database URL.

Attempts to sign up and prints status. Sets token status callback and initializes Firebase.

callBack(char \*inputTopic, byte \*message, unsigned int length) Callback function for MQTT messages.

Prints the arrived message and topic.

**loop()**

Continuously checks and handles Wi-Fi, MQTT, and Serial2 data.

Publishes data to ThingSpeak and Firebase every second.

ThingSpeakPublishData(int field, int data)

Publishes data to ThingSpeak for the specified field.

firebasePublishData(string address, int data)

Publishes data to Firebase at the given address.

**connectToMQTT()**

Attempts to connect to the MQTT broker. publishMessage(const char \*message)

Publishes a message to the specified MQTT topic.

**Data Flow:**

Connect to Wi-Fi.

Setup Firebase and ThingSpeak.

Configure MQTT and establish a connection.

Continuously check for Serial2 data.

If data is available, publish it to ThingSpeak, Firebase, and MQTT every second.

Repeat the process.

**Notes:**

The system uses an MQTT broker for real-time communication.

Data is read from an Arduino device connected via Serial2.

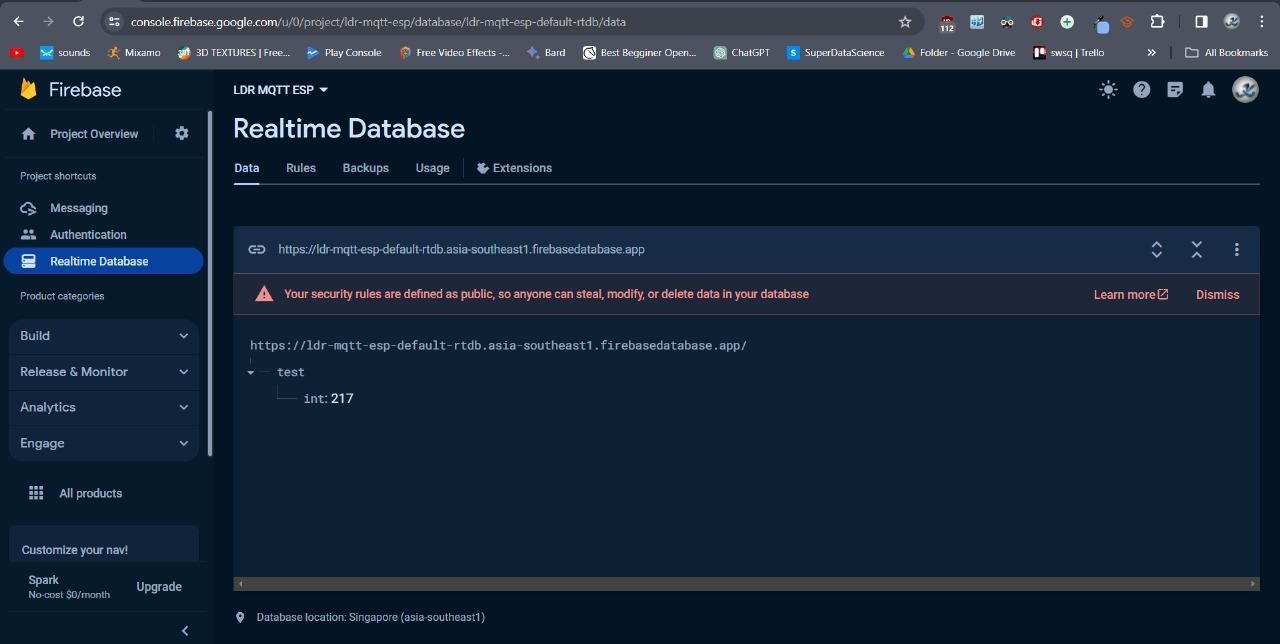
Published data includes light intensity information.

Real-time updates are provided through MQTT.

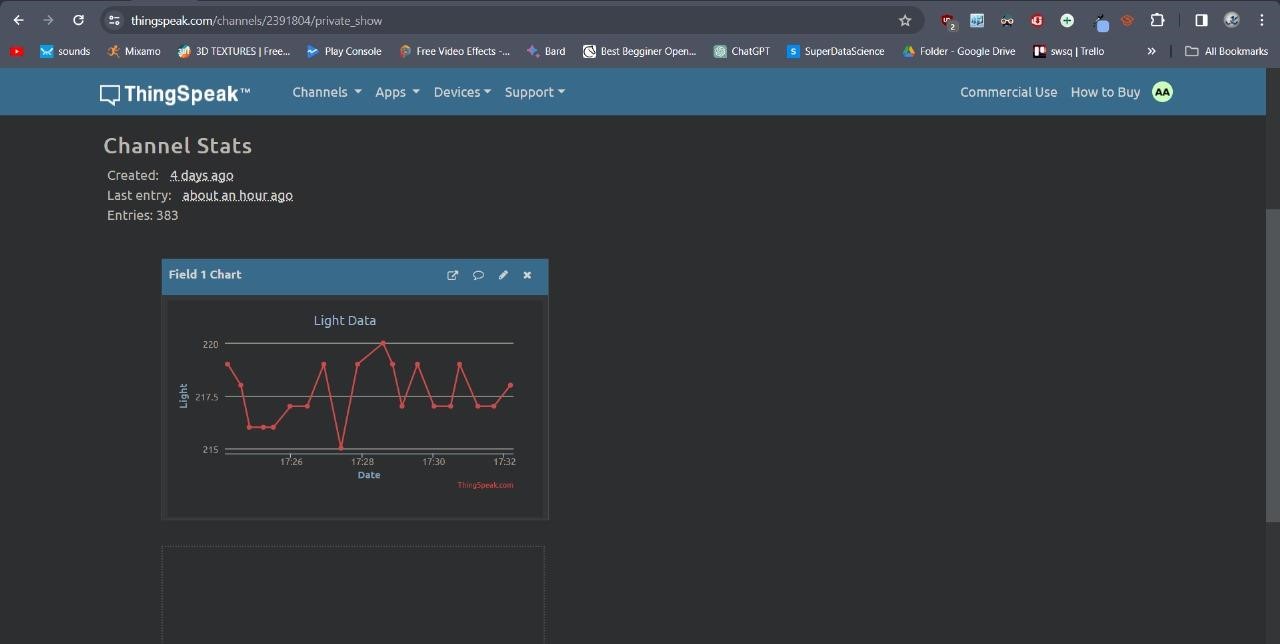
Data is stored in both ThingSpeak and Firebase for historical reference.

Wi-Fi reconnection is attempted if the connection is lost.

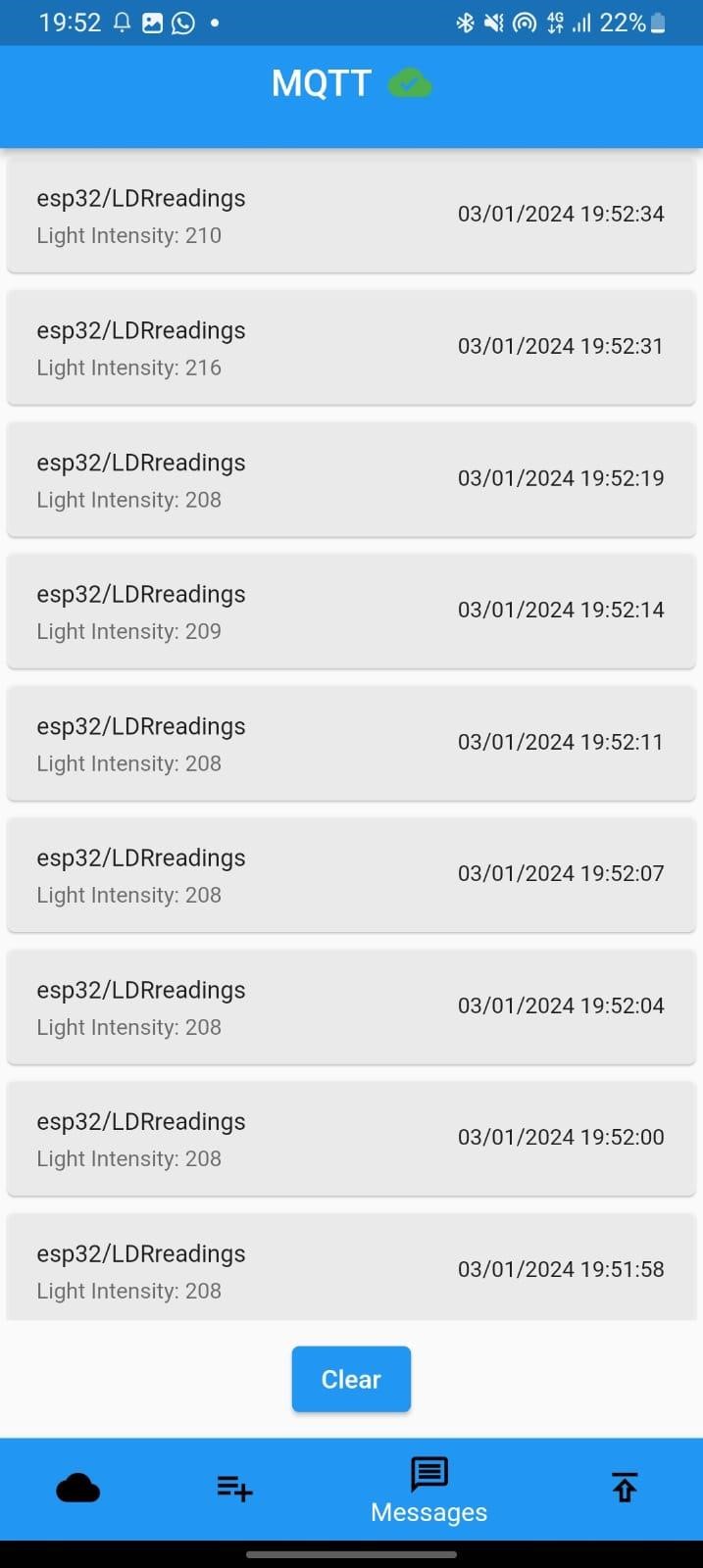
**Firebase RealTime DataBase:**



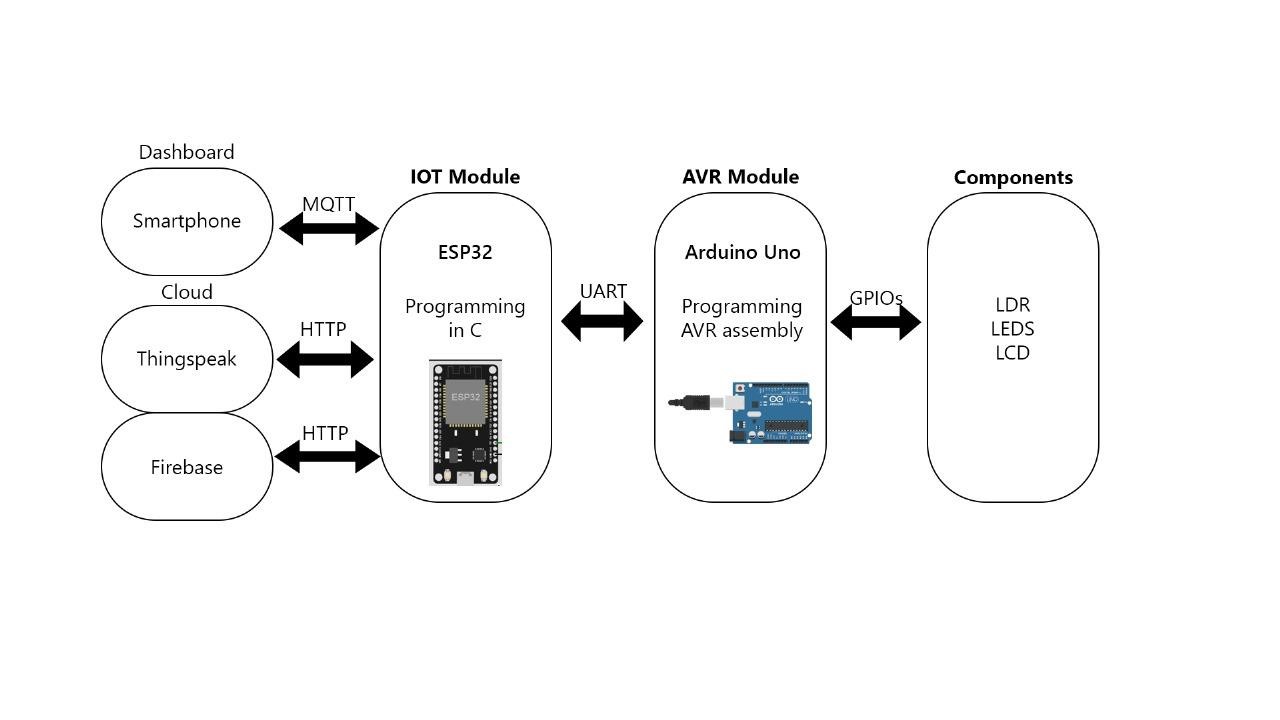
**ThingSpeak:**



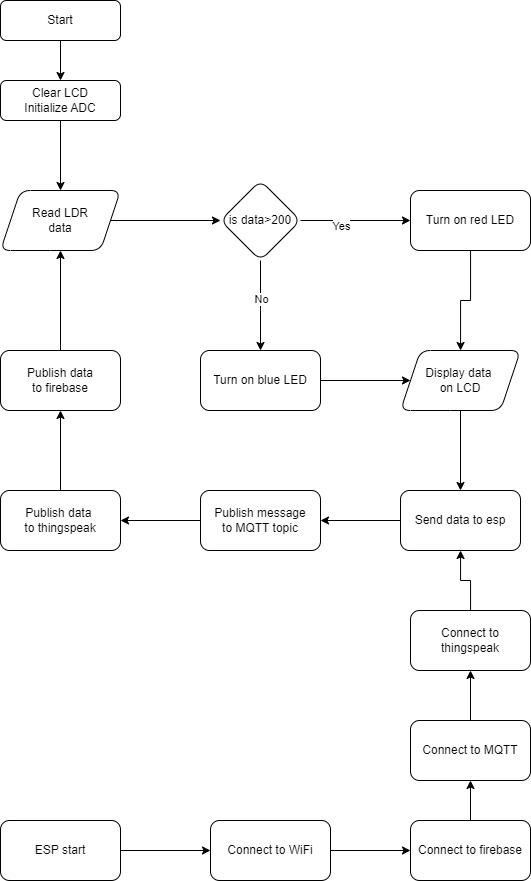
**MQTT dashboard:**



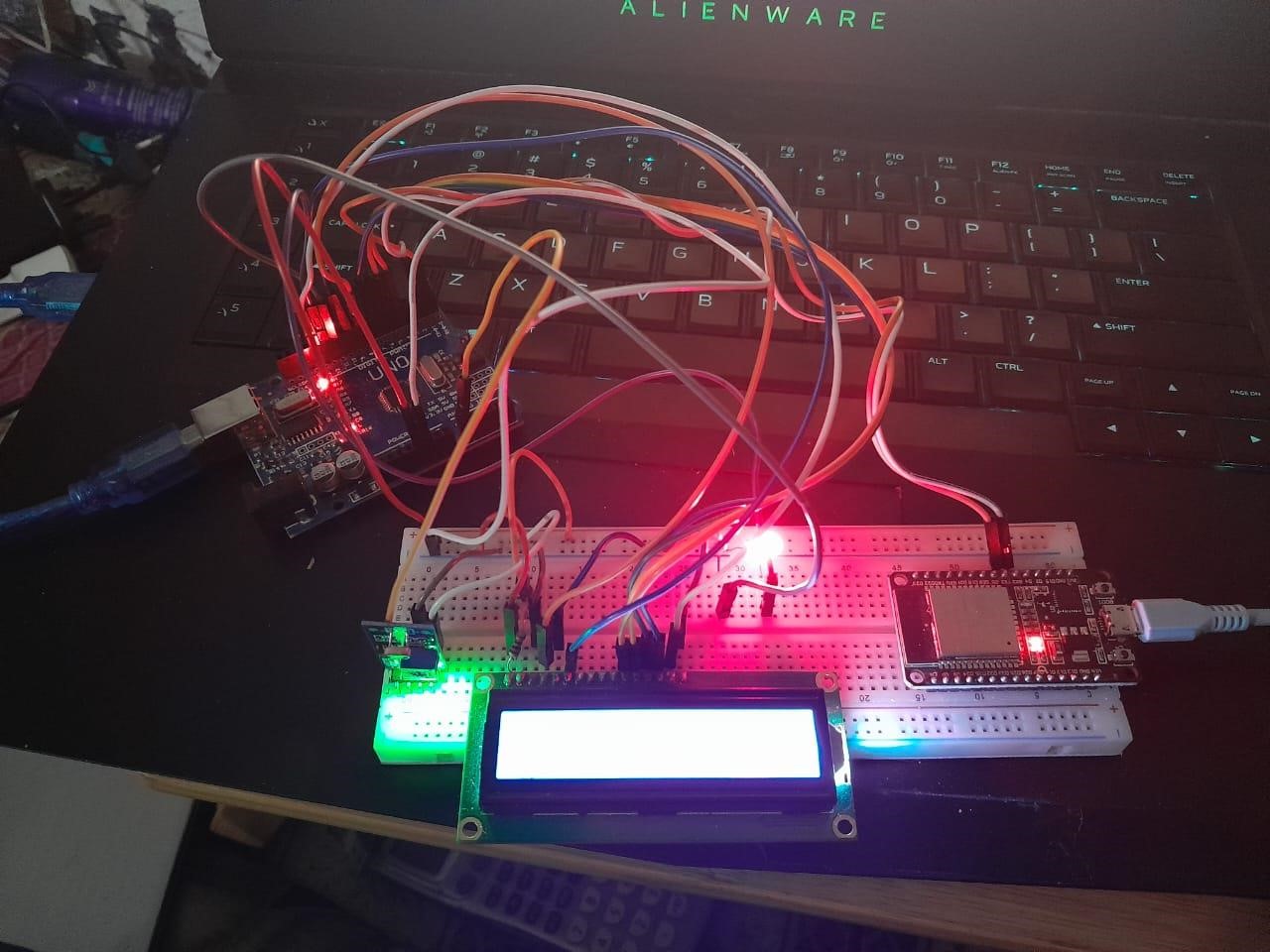
**Data Base Diagram:**

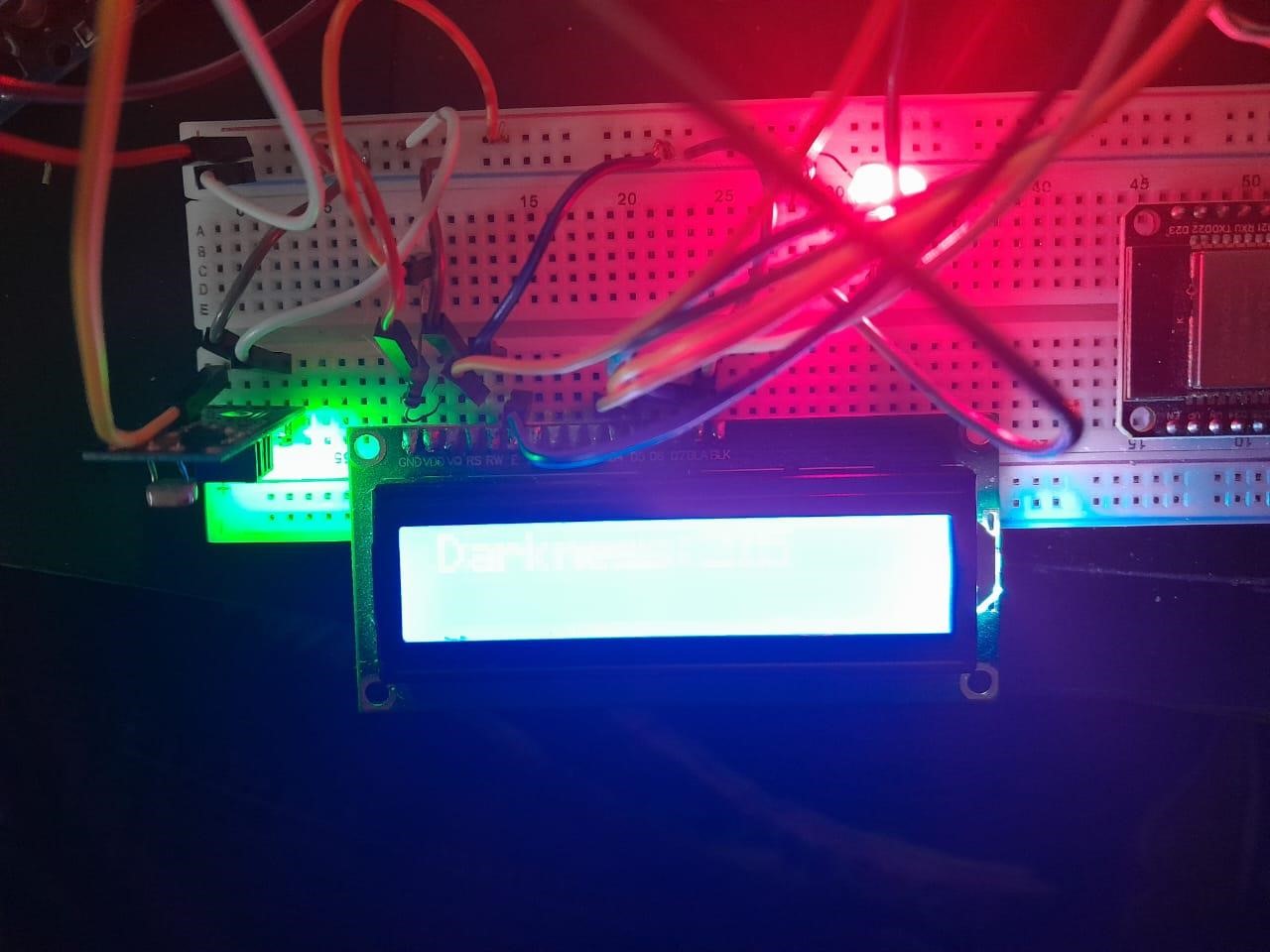
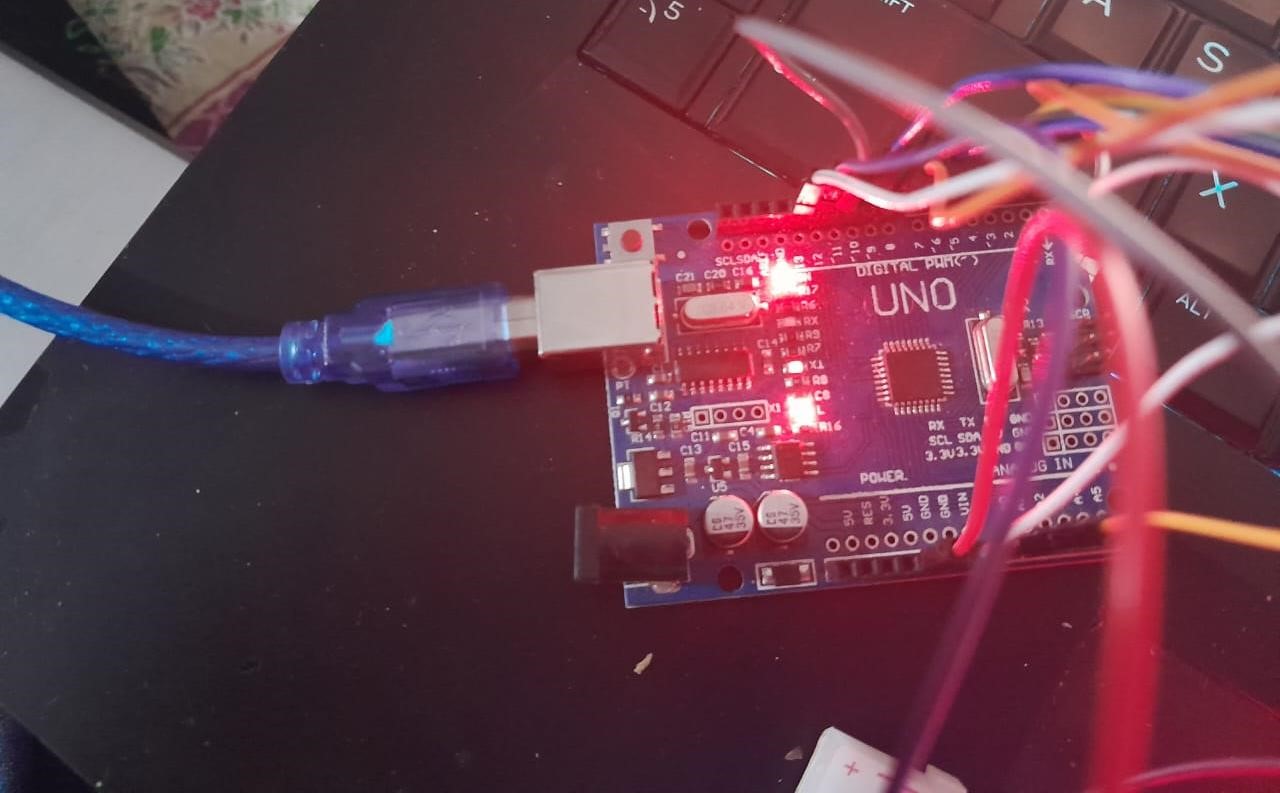


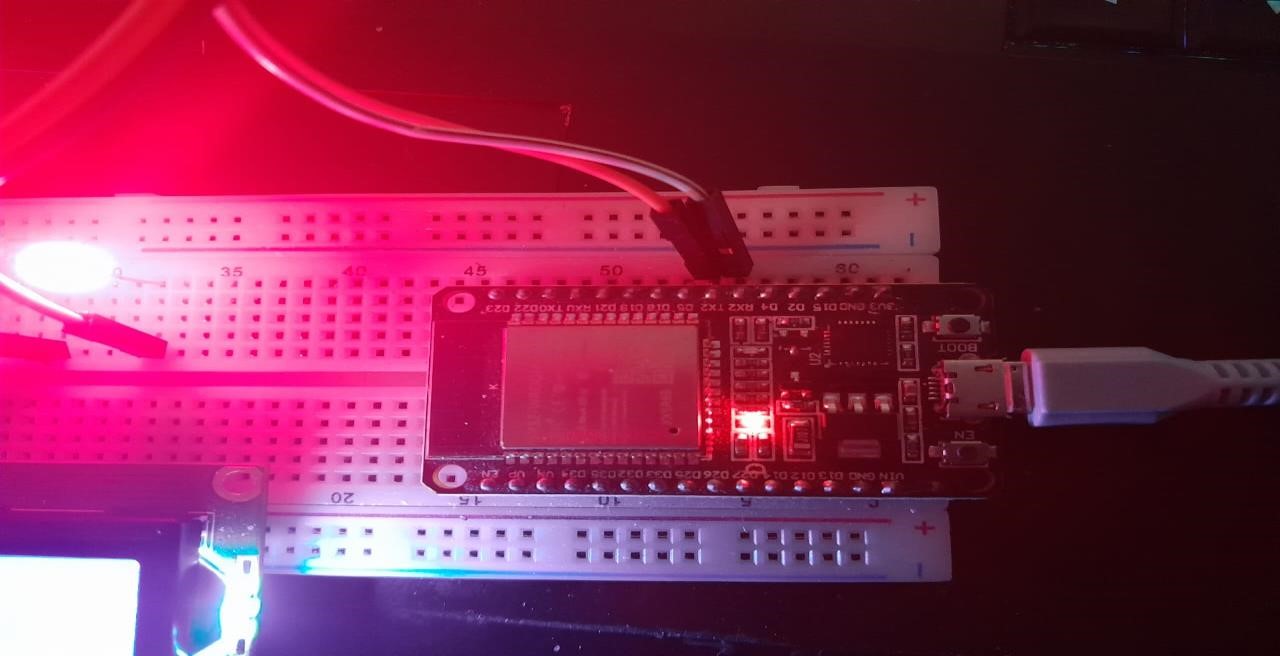
**Flow Chart:**



**Our Project:**









**References:**

[**https://github.com/TehseenHasan/AVR\_Assembly\_Example\_Codes**](https://github.com/TehseenHasan/AVR_Assembly_Example_Codes_for_Atmega328p)

[**\_for\_Atmega328p**](https://github.com/TehseenHasan/AVR_Assembly_Example_Codes_for_Atmega328p)

[**https://randomnerdtutorials.com/esp32-thingspeak-publisharduino/**](https://randomnerdtutorials.com/esp32-thingspeak-publish-arduino/)

[**https://randomnerdtutorials.com/esp32-firebase-realtime-database/**](https://randomnerdtutorials.com/esp32-firebase-realtime-database/)

**Github Link for Code:**

[**https://github.com/kakanics/CoalProject**](https://github.com/kakanics/CoalProject)

**Links for LinkedIn Vidoes:**

**Affan Ahmed’s LinkedIn Video:** [**https://www.linkedin.com/posts/affan-ahmed-142a4824b\_excited-tounveil-our-coal-project-at-department-activity7148504388913750016-**](https://www.linkedin.com/posts/affan-ahmed-142a4824b_excited-to-unveil-our-coal-project-at-department-activity-7148504388913750016-H74c?utm_source=share&utm_medium=member_desktop)

[**H74c?utm\_source=share&utm\_medium=member\_desktop**](https://www.linkedin.com/posts/affan-ahmed-142a4824b_excited-to-unveil-our-coal-project-at-department-activity-7148504388913750016-H74c?utm_source=share&utm_medium=member_desktop)

**Hamna Kamran’s LinkedIn Video:** [**https://www.linkedin.com/posts/hamna-kamran-102546269\_excitedto-unveil-our-coal-project-at-department-activity-**](https://www.linkedin.com/posts/hamna-kamran-102546269_excited-to-unveil-our-coal-project-at-department-activity-7148511752794673153-wa9A?utm_source=share&utm_medium=member_desktop)

[**7148511752794673153-**](https://www.linkedin.com/posts/hamna-kamran-102546269_excited-to-unveil-our-coal-project-at-department-activity-7148511752794673153-wa9A?utm_source=share&utm_medium=member_desktop)

[**wa9A?utm\_source=share&utm\_medium=member\_desktop**](https://www.linkedin.com/posts/hamna-kamran-102546269_excited-to-unveil-our-coal-project-at-department-activity-7148511752794673153-wa9A?utm_source=share&utm_medium=member_desktop)

**Areej’s LinkedIn Video:** [**https://www.linkedin.com/posts/areej-fatima-16a023270\_excited-tounveil-our-coal-project-at-department-activity7148510723718000640-**](https://www.linkedin.com/posts/areej-fatima-16a023270_excited-to-unveil-our-coal-project-at-department-activity-7148510723718000640-LMet?utm_source=share&utm_medium=member_desktop)

[**LMet?utm\_source=share&utm\_medium=member\_desktop**](https://www.linkedin.com/posts/areej-fatima-16a023270_excited-to-unveil-our-coal-project-at-department-activity-7148510723718000640-LMet?utm_source=share&utm_medium=member_desktop) **Ramsha’s LinkedIn Video:**

[**https://www.linkedin.com/posts/ramsha-shabbir-98352a270\_excitedto-unveil-our-coal-project-at-department-activity-**](https://www.linkedin.com/posts/ramsha-shabbir-98352a270_excited-to-unveil-our-coal-project-at-department-activity-7148512771054125056-vyTs?utm_source=share&utm_medium=member_desktop)

[**7148512771054125056-**](https://www.linkedin.com/posts/ramsha-shabbir-98352a270_excited-to-unveil-our-coal-project-at-department-activity-7148512771054125056-vyTs?utm_source=share&utm_medium=member_desktop)

[**vyTs?utm\_source=share&utm\_medium=member\_desktop**](https://www.linkedin.com/posts/ramsha-shabbir-98352a270_excited-to-unveil-our-coal-project-at-department-activity-7148512771054125056-vyTs?utm_source=share&utm_medium=member_desktop)

**Links for Youtube Vidoes:**

**Hamna:**

[**https://youtu.be/yPxkkRDECVs**](https://youtu.be/yPxkkRDECVs)

**Affan:**

[**https://youtu.be/uHyAF5ZTCTY**](https://youtu.be/uHyAF5ZTCTY)

**Areej:** [**https://youtu.be/aLWOkBJxvwc?si=lh4ssqzcKq7nHY9-**](https://youtu.be/aLWOkBJxvwc?si=lh4ssqzcKq7nHY9-)

**Ramsha:**

[**https://youtu.be/Eee-iAFEx3E?si=atq-YO-wxlFft\_5c**](https://youtu.be/Eee-iAFEx3E?si=atq-YO-wxlFft_5c)