**Custom Text Display on 1602 LCD**

****

Session: 2022-2026

**Submitted by:**

Muskan Altaf (2022-CS-116)

Rida Batool (2022-CS-118)

Rushba Irshad (2022-CS-141)

Fatima Shahid (2022-CS-142)

**Submitted to:**

Prof. Tehseen

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

**Contents**

[**Project Photo** 3](#_Toc155266103)

[**Project Description** 4](#_Toc155266104)

[**Methodology Used** 4](#_Toc155266105)

[**Data Flow Diagram (DFD)** 6](#_Toc155266106)

[**Data Flow and Components:** 6](#_Toc155266107)

[**Detailed Flow-chart** 8](#_Toc155266108)

[**Detailed Explanation:** 9](#_Toc155266109)

[**Circuit Diagram** 10](#_Toc155266110)

[**Circuit Connections:** 11](#_Toc155266111)

[**Detailed Explanation of the Working of each Component** 12](#_Toc155266112)

[**AVR Module Code:** 15](#_Toc155266113)

[**IoT Module Code** 16](#_Toc155266114)

[**Code Documentation** 18](#_Toc155266115)

[**AVR Module:** 18](#_Toc155266116)

[**ESP8266 Module:** 20](#_Toc155266117)

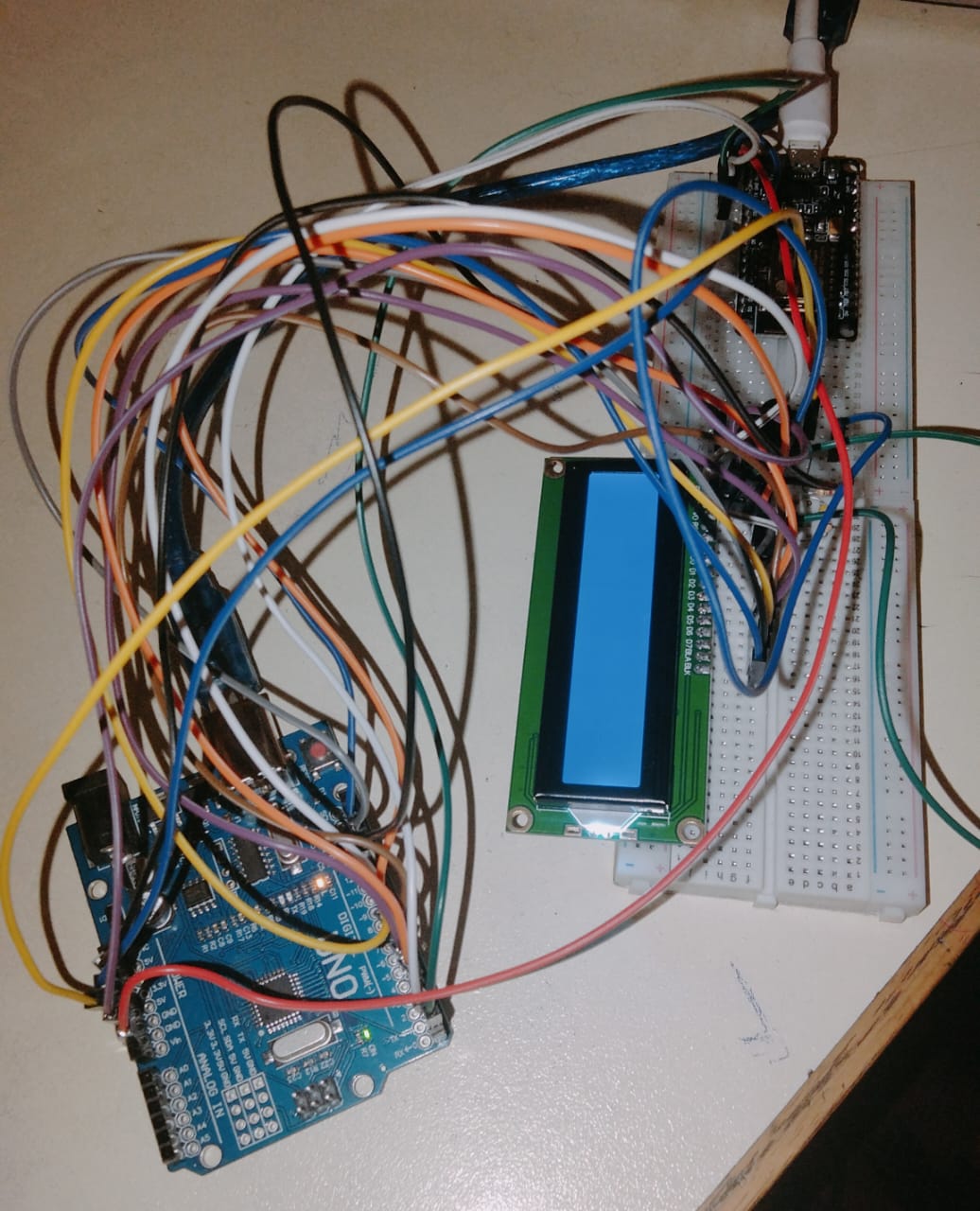
[**MQTT Dash app’s Dashboard Screenshots** 23](#_Toc155266118)

[**Project Video links of YouTube and LinkedIn** 24](#_Toc155266119)

[**GitHub Link** 24](#_Toc155266120)

[**References** 24](#_Toc155266121)

# **Project Photo**

****

# **Project Description**

This project involves creating a system that displays a countdown on a 16x2 LCD using an Arduino UNO and also allows users to input custom text from a smartphone, sending it to the Arduino UNO via ESP8266 through UART communication, and displaying it on the same LCD. Additionally, sensor data is collected and sent to Firebase Realtime Database through the ESP8266.

**Objective:**

The main objectives of this project are as follows:

* Develop an assembly program for the Arduino UNO to control a 16x2 LCD, displaying a countdown from 9 to 0 upon the press of a push button.
* Implement communication between an ESP8266 and a smartphone using MQTT for custom text input.
* Send custom text from the smartphone to the Arduino UNO through UART communication and display it on the 16x2 LCD.
* Collect data from sensors connected to the ESP8266 and send this data to the Firebase Realtime Database.

**Components Required:**

1. Arduino UNO with USB Cable - 1
2. ESP8266 Development Board with Micro USB Cable - 1
3. Breadboard - 1
4. Pin-to-Pin Jumper Wires - 20
5. Push Button 12x12mm - 2
6. 16x2 LCD - 1
7. Male Header Pins Strip for LCD - 1
8. 3.9 Kilo-ohm resistance – 2

# **Methodology Used**

**1. Assembling the Hardware:**

* Connect the 16x2 LCD to the Arduino UNO using jumper wires and a male header strip.
* Connect two push buttons to the Arduino UNO. One button will initiate the countdown, and the other will trigger the sending of custom text.
* Connect the ESP8266 to the Arduino UNO using UART communication. Connect RX of ESP8266 to TX of Arduino and vice versa.
* Connect the necessary resistors for the push buttons to prevent floating values.

**2. Writing Assembly Code for AVR Module (Arduino UNO):**

* Write an assembly program to control the 16x2 LCD. Initialize the LCD and write functions to display the countdown and received text.
* Use interrupts to detect the button presses. One button starts the countdown, and the other initiates the sending of custom text to the Arduino UNO.
* Implement the countdown logic in assembly, decrementing from 9 to 0, and displaying each count on the LCD.
* Create a function to receive custom text from the ESP8266 and display it on the LCD.

**3. Writing Code for IoT Module (ESP8266):**

* Set up the ESP8266 to connect to the MQTT broker and wait for incoming messages from the smartphone.
* Use the MQTT Dash app on the smartphone to send custom text messages to the ESP8266.
* Upon receiving the custom text, send it to the Arduino UNO using UART communication.
* Implement code to read sensor values connected to the ESP8266 and send them to the Firebase Realtime Database.

**4. Integrating Firebase Realtime Database:**

* Set up a Firebase account and create a new project.
* Obtain the necessary credentials (API key, project ID, etc.) to connect the ESP8266 to the Firebase Realtime Database.
* Modify the ESP8266 code to include Firebase functionality and send sensor data to the database.

**5. Testing and Troubleshooting:**

* Test the countdown functionality by pressing the designated button on the Arduino UNO.
* Test the custom text display by sending messages from the smartphone using the MQTT Dash app.
* Verify that sensor data is successfully sent to the Firebase Realtime Database.

**6. Results:**

* Document the connections, pin configurations, and code logic.
* Create a user manual for future reference.
* Ensure all components are securely connected and fix any issues encountered during testing.

# **Data Flow Diagram (DFD)**

Smart Phone

Custom Text (MQTT)

ESP8266

Custom Text (UART)

Arduino UNO

Countdown Logic

16x2 LCD

Display Data

Firebase Realtime DB

## **Data Flow and Components:**

1. **Smartphone:**
   * Sends custom text to the ESP8266 using MQTT (external entity).
2. **ESP8266:**
   * Receives custom text from the smartphone using MQTT.
   * Sends the custom text to the Arduino UNO using UART.
   * Collects sensor data.
3. **Arduino UNO:**
   * Receives custom text from the ESP8266 using UART.
   * Executes countdown logic and controls the 16x2 LCD.
4. **16x2 LCD:**
   * Displays countdown and received custom text.
5. **Firebase Realtime Database:**
   * Receives and stores sensor data from the ESP8266.

**Processes:**

* **Countdown Logic (Inside Arduino UNO):**
  + Controls the countdown logic based on button input.
  + Communicates with the 16x2 LCD to display the countdown.
* **Display Data (Inside 16x2 LCD):**
  + Receives data from the Arduino UNO.
  + Displays countdown and received custom text on the LCD.

**Notes:**

* External entities, such as the smartphone and Firebase Realtime Database, are represented outside the system boundary.
* Specific technologies and protocols (MQTT, UART) are highlighted to indicate the nature of data communication.
* Processes are described within key components to show specific functionalities.

# **Detailed Flow-chart**

Initialization and Setup

(Configuring Pins, Libraries, etc)

Smartphone Interaction

(MQTT Dash App)

Receive Custom Text on ESP8266 (MQTT Communication)

Send Custom Text to Arduino UNO (UART Communication)

Countdown Logic on Arduino UNO

(Button Press Detection)

(Update 16x2 LCD Display)

Display Custom Text on 16x2 LCD

Collect Sensor Data on ESP8266 (Read Sensor Values and Store Data)

Send Sensor Data to Firebase DB (Firebase Integration)

## **Detailed Explanation:**

1. **Initialization and Setup:**
   * Configure pins, libraries, and other settings on both the ESP8266 and Arduino UNO.
2. **Smartphone Interaction:**
   * User inputs custom text on the smartphone using the MQTT Dash app.
3. **Receive Custom Text on ESP8266:**
   * ESP8266 receives custom text via MQTT communication.
4. **Send Custom Text to Arduino UNO:**
   * ESP8266 sends the received custom text to the Arduino UNO using UART communication.
5. **Countdown Logic on Arduino UNO:**
   * The Arduino UNO detects button presses to initiate countdown logic.
   * Updates the 16x2 LCD display with the countdown values.
6. **Display Custom Text on 16x2 LCD:**
   * The Arduino UNO displays the received custom text on the 16x2 LCD.
7. **Collect Sensor Data on ESP8266:**
   * Read sensor values connected to the ESP32.
8. **Send Data to Firebase DB:**
   * ESP8266 sends the collected data to the Firebase Realtime Database for storage.
9. **End of Program:**
   * The flowchart concludes at the end of the program.

**Notes:**

* Each step represents a specific action or decision point in the project.
* The flowchart provides a visual representation of the sequence of operations in the "Custom Text Display on 1602 LCD" project.
* Depending on the complexity of the actual code, additional decision points or loops might be present. This flowchart provides a high-level overview.

# **Circuit Diagram**

Smartphone

(MQTT Dash)

MQTT Broker

Arduino UNO

ESP8266

Custom Text (UART)

Custom Text (MQTT)

Arduino UNO

16x2 LCD

Arduino UNO

ESP8266

Arduino UNO

ESP8266

Smartphone

Custom Text (UART)

Custom Text (MQTT)

Smartphone

Custom Text (UART)

Custom Text (MQTT)

MQTT Broker

Custom Text (UART)

Custom Text (MQTT)

Custom Text (UART)

Custom Text (MQTT)

ESP8266

ESP8266

Arduino UNO

Firebase Realtime Database

## **Circuit Connections:**

1. **Arduino UNO and 16x2 LCD:**
   * Connect the 16x2 LCD to the Arduino UNO using jumper wires.
   * Utilize a male header strip to connect the LCD to the breadboard for a stable connection.
2. **ESP8266** **and Arduino UNO:**
   * Connect the ESP8266 and Arduino UNO using UART communication.
   * Ensure proper grounding between the ESP8266 and Arduino UNO.
3. **Push Buttons:**
   * Connect push buttons to the Arduino UNO for countdown initiation and custom text triggering.
   * Use 3.9 Kilo-ohm resistors in series with the push buttons for stable voltage levels.
4. **Breadboard and Jumper Wires:**
   * Connect components on the breadboard using jumper wires.
   * Organize the layout to ensure a clear and manageable circuit.

# **Detailed Explanation of the Working of each Component**

**1. Arduino UNO with USB Cable:**

* **Description:**
  + The Arduino UNO is a microcontroller board based on the ATmega328P. It provides a simple and versatile platform for electronics projects.
  + Equipped with digital and analogue pins, it's commonly used for interfacing with various sensors, displays, and other peripherals.
  + The USB cable allows for programming the Arduino UNO and provides power.
* **Role in the Project:**
  + The Arduino UNO serves as the central processing unit for the project, controlling the 16x2 LCD, managing the countdown, and receiving custom text from the ESP8266.

**2. ESP8266** **Development Board with Micro USB Cable:**

* **Description**:
  + The ESP8266 development board is a powerful and versatile microcontroller based on the ESP8266 module.
  + It features built-in Wi-Fi and Bluetooth capabilities, making it suitable for IoT applications.
  + The Micro USB cable is used for programming and power supply.
* **Role in the Project:**
  + The ESP8266 facilitates communication with the smartphone using MQTT, sends custom text to the Arduino UNO via UART, and collects sensor data for transmission to the Firebase Realtime Database.

**3. Breadboard:**

* **Description:**
  + A breadboard is a crucial tool for prototyping electronic circuits without soldering.
  + It consists of a grid of holes, allowing easy insertion and connection of electronic components.
* **Role in the Project:**
  + The breadboard provides a platform for temporary connections between components, aiding in the assembly and testing of the circuit.

**4. Pin-to-Pin Jumper Wires:**

* **Description**:
  + Jumper wires are flexible wires with connectors at both ends, allowing easy connections between components on a breadboard.
  + They come in various lengths and colours for organization.
* **Role in the Project:**
  + Jumper wires are used to establish electrical connections between the Arduino UNO, ESP8266, LCD, push buttons, and other components on the breadboard.

**5. Push Button 12x12mm:**

* **Description**:
  + Push buttons are simple mechanical switches used to control electrical circuits by making or breaking the connection when pressed.
  + The 12x12mm size is a common dimension for these switches.
* **Role in the Project:**
  + Two push buttons are employed in the project. One initiates the countdown on the Arduino UNO, and the other triggers the sending of custom text from the ESP8266.

**6. 16x2 LCD:**

* **Description:**
  + The 16x2 LCD (16 characters per line, 2 lines) is a standard alphanumeric display.
  + It consists of a liquid crystal display and is commonly used for text-based output in electronics projects.
* **Role in the Project:**
  + The 16x2 LCD displays the countdown initiated by the Arduino UNO and shows custom text received from the ESP8266.

**7. Male Header Pins Strip for LCD:**

* **Description:**
  + Male header pins are metal pins attached to a plastic base. They can be inserted into female headers or soldered to a PCB.
  + They provide a convenient way to connect components like the LCD to a breadboard.
* **Role in the Project:**
  + The male header pins strip is used to connect the 16x2 LCD to the breadboard, facilitating a secure and reliable connection.

**8. Kilo-ohm Resistance:**

* **Description:**
  + A 3.9 Kilo-ohm resistor has a resistance of 3.9 kilo-ohms (3900 ohms).
  + It limits current in the circuit, especially when connected to push buttons to avoid floating values.
* **Role in the Project:**
  + Two 3.9 Kilo-ohm resistors are used in conjunction with push buttons to ensure stable and defined voltage levels, preventing undesired behaviour when the buttons are not pressed.

# **AVR Module Code:**

.include "m328pdef.inc"

.include "delay.inc"

.include "lcd\_Macros.inc"

.include "UART\_Macros.inc"

.dseg

.org SRAM\_START

countdown: .byte 1

.cseg

.org 0x0000

LCD\_init ; initilize the 16x2 LCD

LCD\_backlight\_OFF

delay 500

LCD\_backlight\_ON

Serial\_begin ; Initialize UART Communication

; Initialize countdown value

LDI r16, 9

STS countdown, r16

loop:

countdown\_loop:

; Display countdown value on LCD

LCD\_clear ; Clear the LCD

LCD\_home ; Move cursor to the beginning of the first line

LDS r16, countdown

LCD\_send\_a\_register r16

; Decrement countdown

LDS r16, countdown ; Load value from memory into register

DEC r16 ; Decrement the register

STS countdown, r16 ; Store the updated value back to memory

; Delay for 1 second

delay 1000

; Check if countdown is complete

LDS r16, countdown ; Load value from memory into register

CPI r16, 0 ; Compare the register with immediate value 0

BREQ countdown\_complete

RJMP countdown\_loop

countdown\_complete:

LDI r16, 0

LCD\_send\_a\_command 0x01 ; clear the LCD

delay 500

receive\_text:

Serial\_read

CPI r16, 0

BREQ loop2

rjmp display\_text

loop2:

rjmp countdown\_complete

display\_text:

LCD\_send\_a\_reg\_ASCII r16

RJMP receive\_text ; Continue to receive more text

; Add an endless loop at the end if needed

endless\_loop:

RJMP endless\_loop

# **IoT Module Code**

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

#include <Firebase\_ESP\_Client.h>

// Provide the token generation process info.

#include <addons/TokenHelper.h>

// Provide the RTDB payload printing info and other helper functions.

#include <addons/RTDBHelper.h>

const char\* ssid = "OPPO A15s";

const char\* password = "ridabatool09";

const char\* mqtt\_server = "broker.hivemq.com";

const int mqtt\_port = 1883;

/\* 2. Define the API Key \*/

#define API\_KEY "AIzaSyDnqeFgtIIRkZRltT05U2aWOqXyRFKOwXg"

/\* 3. Define the RTDB URL \*/

#define DATABASE\_URL "projectcoal-d7a68-default-rtdb.firebaseio.com/"

// Define Firebase Data object

FirebaseData fbdo;

FirebaseAuth auth;

FirebaseConfig config;

bool signupOK = false;

WiFiClient espClient;

PubSubClient client(espClient);

void setup\_wifi()

{

  delay(10);

  WiFi.mode (WIFI\_STA);

  WiFi.begin (ssid, password);

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

  {

    delay(500);

  }

  /\* Assign the api key (required) \*/

  config.api\_key = API\_KEY;

  /\* Assign the RTDB URL (required) \*/

  config.database\_url = DATABASE\_URL;

  /\* Sign up \*/

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

  {

    signupOK = true;

  }

  else

  {

    delay(50);

  }

  /\* Assign the callback function for the long running token generation task \*/

  config.token\_status\_callback = tokenStatusCallback; //see addons/TokenHelper.h

  Firebase.begin(&config, &auth);

  Firebase.reconnectWiFi (true);

}

void callback(char\* topic, byte\* payload, unsigned int length)

{

  // Print received MQTT message to UART

  for (int i = 0; i < length; i++)

  {

    Serial.write(payload[i]);

    delay(750);

  }

  // Convert payload to String

  String mqttPayload = String((char\*)payload);

  // Send the received data to Firebase

  if (Firebase.ready() && signupOK)

  {

    // Write the MQTT payload to the database path test/mqtt\_data

    if (Firebase.RTDB.setString(&fbdo, "test/mqtt\_data", mqttPayload))

   {

    }

    else

    {

      delay(50);

    }

  }

}

void reconnect()

{

  while (!client.connected())

  {

    String clientId = "ESP8266Client-";

    clientId += String(random(0xffff), HEX);

    if (client.connect(clientId.c\_str()))

    {

      client.subscribe("2022-cs-118");

    }

    else

    {

      delay(5000);

    }

  }

}

void setup()

{

  Serial.begin(9600);

  setup\_wifi();

  client.setServer(mqtt\_server, mqtt\_port);

  client.setCallback(callback);

}

void loop()

{

  if (!client.connected())

  {

    reconnect();

  }

  client.loop();

}

**Code Documentation**  
Here's a basic documentation for both the AVR module code and the ESP8266 module code:

## **AVR Module:**

**Purpose:**

This AVR module code is designed to control a 16x2 LCD display and perform a countdown from 9 to 0. After the countdown is complete, it reads characters from a serial interface (UART) and displays them on the LCD.

**Components:**

* AVR microcontroller (ATmega328P)
* 16x2 LCD Display
* UART Communication Module

**Code Structure:**

**Initialization:**

* + Initialize LCD, turn off the backlight, delay, and then turn on the backlight.
  + Initialize UART communication.

LCD\_init

LCD\_backlight\_OFF

delay 500

LCD\_backlight\_ON

Serial\_begin

**Countdown Loop:**

* Initialize a countdown variable.
* Display the countdown value on the LCD.
* Decrement the countdown and repeat until the countdown reaches 0.

; Initialize countdown value

LDI r16, 9

STS countdown, r16

loop:

countdown\_loop:

; Display countdown value on LCD

LCD\_clear ; Clear the LCD

LCD\_home ; Move cursor to the beginning of the first line

LDS r16, countdown

LCD\_send\_a\_register r16

; Decrement countdown

LDS r16, countdown ; Load value from memory into register

DEC r16 ; Decrement the register

STS countdown, r16 ; Store the updated value back to memory

; Delay for 1 second

delay 1000

; Check if countdown is complete

LDS r16, countdown ; Load value from memory into register

CPI r16, 0 ; Compare the register with immediate value 0

BREQ countdown\_complete

RJMP countdown\_loop

countdown\_complete:

LDI r16, 0

LCD\_send\_a\_command 0x01 ; clear the LCD

delay 500

**Display Received Text:**

* Read characters from UART and display them on the LCD.

display\_text:

LCD\_send\_a\_reg\_ASCII r16

RJMP receive\_text

**Endless Loop:**

* Optionally, include an endless loop to keep the program running.

endless\_loop:

RJMP endless\_loop

## **ESP8266 Module:**

**Purpose:**

This ESP8266 module code connects to a WiFi network, subscribes to an MQTT topic, and forwards received messages to Firebase Realtime Database.

**Components:**

* ESP8266 microcontroller
* WiFi Module
* MQTT Client
* Firebase Realtime Database

**Code Structure:**

1. **WiFi Setup:**
   * Connect to a WiFi network and configure Firebase authentication.

void setup\_wifi()

{

  delay(10);

  WiFi.mode(WIFI\_STA);

  WiFi.begin(ssid, password);

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

  {

    delay(500);

  }

  /\* Assign the api key (required) \*/

  config.api\_key = API\_KEY;

  /\* Assign the RTDB URL (required) \*/

  config.database\_url = DATABASE\_URL;

  /\* Sign up \*/

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

  {

    signupOK = true;

  }

  else

  {

    delay(50);

  }

  /\* Assign the callback function for the long running token generation task \*/

  config.token\_status\_callback = tokenStatusCallback; //see addons/TokenHelper.h

  Firebase.begin(&config, &auth);

  Firebase.reconnectWiFi(true);

}

**MQTT Callback:**

* Define a callback function to handle received MQTT messages.
* Convert the payload to a string and send it to Firebase.

void callback(char\* topic, byte\* payload, unsigned int length)

{

  // Print received MQTT message to UART

  for (int i = 0; i < length; i++)

  {

    Serial.write(payload[i]);

    delay(750);

  }

  // Convert payload to String

  String mqttPayload = String((char\*)payload);

  // Send the received data to Firebase

  if (Firebase.ready() && signupOK)

  {

    // Write the MQTT payload to the database path test/mqtt\_data

    if (Firebase.RTDB.setString(&fbdo, "test/mqtt\_data", mqttPayload))

    {

    }

    else

    {

      delay(50);

    }

  }

}

**Reconnect Function:**

* Reconnect to the MQTT broker in case of disconnection.

void reconnect()

{

  while (!client.connected())

  {

    String clientId = "ESP8266Client-";

    clientId += String(random(0xffff), HEX);

    if (client.connect(clientId.c\_str()))

    {

      client.subscribe("2022-cs-118");

    }

    else

    {

      delay(5000);

    }

  }

}

**Setup Function:**

* Initialize serial communication, set up WiFi, MQTT server, and callback.

void setup()

{

  Serial.begin(9600);

  setup\_wifi();

  client.setServer(mqtt\_server, mqtt\_port);

  client.setCallback(callback);

}

**Main Loop:**

* Check and maintain the connection to the MQTT broker.
* Execute the callback function on received messages.

void loop()

{

  if (!client.connected())

  {

    reconnect();

  }

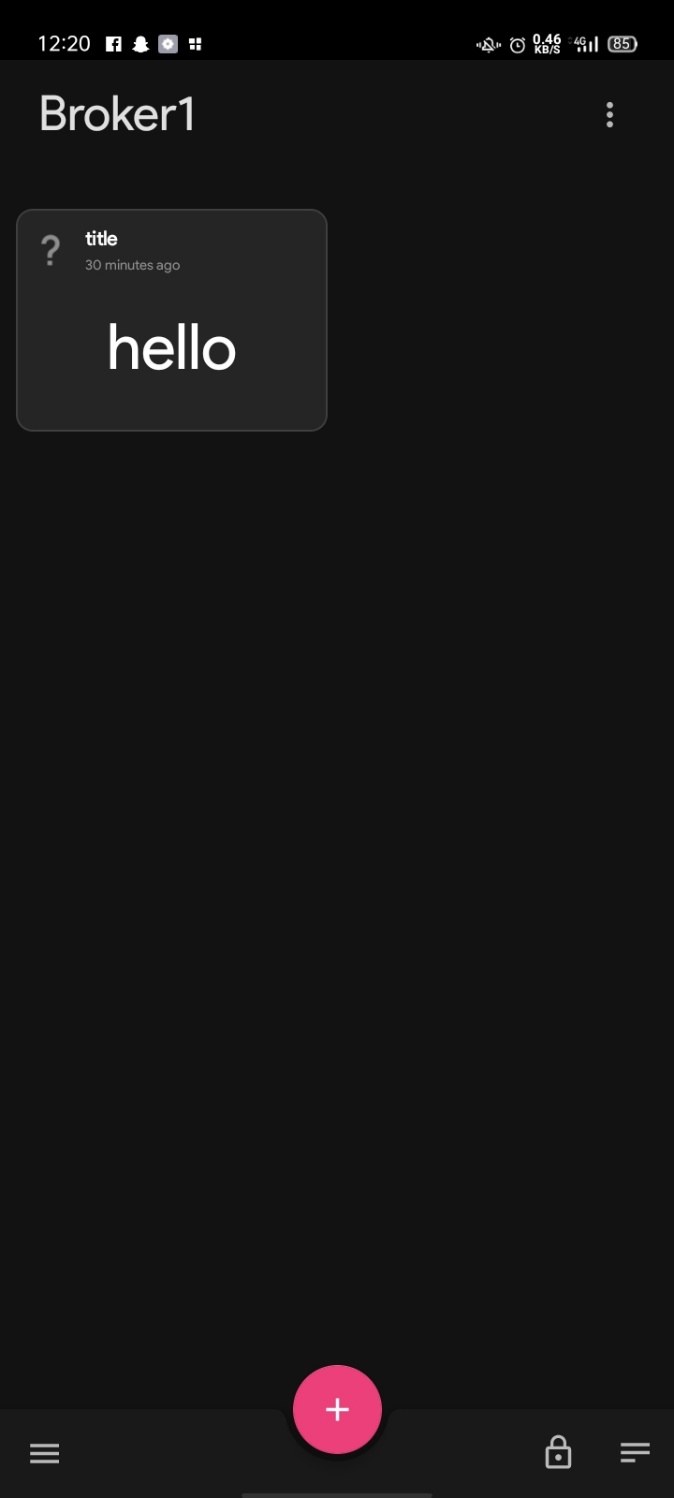
  client.loop();

}

**Additional Notes:**

* The code relies on external libraries: **ESP8266WiFi**, **PubSubClient**, and **Firebase\_ESP\_Client**.
* API Key and RTDB URL are defined for Firebase authentication.
* The code continuously loops to maintain MQTT and Firebase connections.

# **MQTT Dash app’s Dashboard Screenshots**

****

# **Project Video links of YouTube and LinkedIn**

* <https://www.linkedin.com/posts/muskan-altaf-8a6261270_arduino-esp8266-iot-activity-7148577070137204736-1bqV?utm_source=share&utm_medium=member_android>
* <https://www.linkedin.com/posts/fatima-shahid-821556243_arduino-esp8266-iot-activity-7148571022596751360-RORh?utm_source=share&utm_medium=member_android>
* <https://www.linkedin.com/posts/rida-batool-5b8270271_arduino-esp8266-iot-activity-7148576702040891392-EuRd?utm_source=combined_share_message&utm_medium=member_android>
* <https://www.linkedin.com/posts/rushba-arshad-0b692127b_arduino-diyprojects-electronics-activity-7148573169287700480-M5Vn?utm_source=share&utm_medium=member_android>
* <https://youtu.be/RE8I4y-X9K4?si=yEy5FsDJho8FOIwF>

# **GitHub Link**

* <https://github.com/Rida-7/custom_text_display_1602_LCD_coal_project>

# **References**

* <https://github.com/TehseenHasan/AVR_Assembly_Example_Codes_for_Atmega328p>
* <https://www.arduino.cc/>
* <https://chat.openai.com/>