**SHIFTER MQTT**

**ABSTRACT:**

In the rapidly evolving field of the Internet of Things (IoT), the integration of microcontrollers, sensors, and cloud-based communication platforms has become increasingly essential. This project demonstrates the development and implementation of a basic IoT system using an ESP32 microcontroller, an ultrasonic distance sensor, a buzzer, and MQTT for communication, specifically leveraging the Shiftr.io platform. The primary objective is to create a functional setup that measures distances using the ultrasonic sensor, publishes these measurements to an MQTT broker, and remotely controls a buzzer through MQTT messages.

The project begins with the configuration of the ESP32 to connect to a specified WiFi network, enabling internet connectivity. Upon establishing the WiFi connection, the ESP32 connects to the Shiftr.io MQTT broker using the MQTT protocol. The system subscribes to specific MQTT topics to receive commands for the buzzer and publishes sensor data to other topics for remote monitoring.

The ultrasonic sensor, connected to the ESP32, continuously measures the distance to an object. These measurements are processed and published to the /ultrasonic MQTT topic at regular intervals. Simultaneously, the ESP32 subscribes to the /buzzer topic to receive commands that control the state of the buzzer. Based on the received payload (1 or 0), the buzzer is turned on or off. The current state of the buzzer is then published to the /buzzerStatus topic to provide feedback on its status.

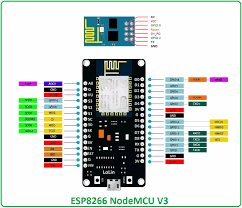
MQTT Box, a tool for monitoring and testing MQTT communications, is utilized to visualize the data being transmitted and received. This tool allows users to manually publish messages to control the buzzer and observe the distance measurements published by the ESP32.

The project showcases several core IoT concepts, including the integration of sensors and actuators with cloud-based communication platforms, remote device control, and real-time data monitoring. It highlights the practical application of MQTT in IoT systems, demonstrating how devices can communicate efficiently over the internet.

Through this project, users gain hands-on experience with setting up and configuring an MQTT broker, programming an ESP32 for WiFi and MQTT connectivity, and using MQTT to enable real-time communication between devices and the cloud. The project lays the groundwork for more complex IoT applications, such as home automation, environmental monitoring, and remote device management, illustrating the potential of IoT technologies to enhance and automate everyday tasks

**COMPONENTS:**

**ESP8266**: The ESP8266 is a low-cost Wi-Fi microcontroller chip with full TCP/IP stack and microcontroller capability. It is widely used in IoT projects due to its ability to connect to a Wi-Fi network and process data. In this project, the ESP8266 handles Wi-Fi connectivity, receives control commands, and interfaces with both the servo motor and the LCD display.



**ULTRASONIC SENSOR:** The ultrasonic sensor emits sound waves and measures the time it takes for them to bounce back, enabling precise distance calculation. Integrated with the ESP8266, it provides real-time distance data wirelessly to the Blynk app, offering remote distance monitoring on mobile devices.



**Jumper Wires**: Jumper wires are insulated wires with connectors (typically male or female pins) at each end, used to create temporary or semi-permanent connections between different components on a breadboard or between a breadboard and other devices.



**BUZZER:**The buzzer system integrated with the Arduino IoT Cloud enables remote toggling of its state, providing a seamless user experience for activating or deactivating auditory feedback. This functionality is achieved through the synchronization of the NodeMCU microcontroller with the cloud platform, facilitating real-time control and monitoring of the buzzer's operations from anywhere with an internet connection.



**CODE:**

#include <WiFi.h>

#include <MQTT.h>

const char ssid[] = "moto g54 5G\_3480";

const char pass[] = "9500697232";

const char mqttUsername[] = "new-iot"; // Change to your MQTT username

const char mqttPassword[] = "dn3WXRPJD77svMgG"; // Change to your MQTT password

WiFiClient net;

MQTTClient client;

const int buzzerPin = 26; // Change to your actual buzzer pin number

const int trigPin = 2; // Ultrasonic sensor trigger pin

const int echoPin = 4; // Ultrasonic sensor echo pin

unsigned long lastMillis = 0;

void connectWiFi() {

Serial.print("Connecting to WiFi...");

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

Serial.print(".");

delay(1000);

}

Serial.println(" Connected to WiFi!");

}

void connectMQTT() {

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

while (!client.connect("arduino", mqttUsername, mqttPassword)) {

Serial.print(".");

delay(1000);

}

Serial.println(" Connected to MQTT!");

client.subscribe("/buzzer/control");

}

void messageReceived(String &topic, String &payload) {

Serial.println("Incoming: " + topic + " - " + payload);

if (topic == "/buzzer/control") {

if (payload == "1") {

digitalWrite(buzzerPin, HIGH); // Turn on buzzer

client.publish("/buzzer/status", "ON"); // Publish buzzer status as 1 (ON)

} else if (payload == "0") {

digitalWrite(buzzerPin, LOW); // Turn off buzzer

client.publish("/buzzer/status", "OFF"); // Publish buzzer status as 0 (OFF)

}

}

}

float measureDistance() {

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

unsigned long duration = pulseIn(echoPin, HIGH);

float distance = (duration \* 0.0343) / 2; // Speed of sound is 343m/s, divide by 2 for one-way distance

return distance;

}

void setup() {

Serial.begin(115200);

WiFi.begin(ssid, pass);

pinMode(buzzerPin, OUTPUT); // Set buzzer pin as output

pinMode(trigPin, OUTPUT); // Set ultrasonic sensor trigger pin as output

pinMode(echoPin, INPUT); // Set ultrasonic sensor echo pin as input

connectWiFi();

client.begin("new-iot.cloud.shiftr.io", net);

client.onMessage(messageReceived);

connectMQTT();

}

void loop() {

client.loop();

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

connectWiFi();

}

if (!client.connected()) {

connectMQTT();

}

// Measure distance and publish

float distance = measureDistance();

Serial.print("Distance: ");

Serial.print(distance);

Serial.println(" cm");

client.publish("/ultrasonic", String(distance));

// Wait for a moment before taking next measurement

delay(1000);

}



**CODE EXPALNATION**

#### Libraries and Credentials

* **Includes Necessary Libraries for WiFi and MQTT:** The project begins by including the necessary libraries for WiFi and MQTT communication. These libraries enable the ESP32 to connect to a WiFi network and communicate with the MQTT broker.
* **Stores WiFi and MQTT Credentials:** The WiFi SSID and password, along with MQTT broker username and password, are stored in variables. This allows the ESP32 to connect to the specified WiFi network and authenticate with the MQTT broker.

#### WiFi and MQTT Setup

* **Connecting to WiFi:** A function is defined to handle the connection to the WiFi network. It continuously attempts to connect until a connection is established, providing feedback via the serial monitor.
* **Connecting to MQTT Broker:** Another function handles the connection to the MQTT broker. It also continuously attempts to connect until successful and subscribes to a specific MQTT topic to control the buzzer.

#### MQTT Callback

* **Handling Received MQTT Messages:** A callback function is defined to handle messages received on the subscribed MQTT topic. When a message is received, the function checks the topic and payload. If the message is for the buzzer control topic and contains "1", the buzzer is turned on. If the message contains "0", the buzzer is turned off. The buzzer's status is then published to a separate MQTT topic.

#### Ultrasonic Distance Measurement

* **Measuring Distance Using the Ultrasonic Sensor:** A function is defined to measure the distance using the ultrasonic sensor. It triggers the sensor to send out a pulse and measures the time it takes for the echo to return. This time is used to calculate the distance to an object in centimeters.

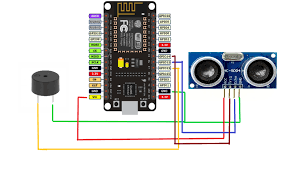
#### Setup Function

* **Initialization of Serial Communication, WiFi, MQTT Connections, and Pin Modes:** In the setup function, the serial communication is initialized for debugging purposes. The ESP32 attempts to connect to the specified WiFi network and MQTT broker. The pin modes for the buzzer and ultrasonic sensor are set up to ensure proper operation.

#### Main Loop

* **Maintaining WiFi and MQTT Connections and Publishing Distance Measurements:** The main loop continuously checks and maintains the WiFi and MQTT connections. If either connection is lost, it attempts to reconnect. The loop also measures the distance using the ultrasonic sensor at regular intervals and publishes the measured distance to a specific MQTT topic. This ensures that the distance data is continuously sent to the MQTT broker.

**CIRCUIT DIAGRAM:**

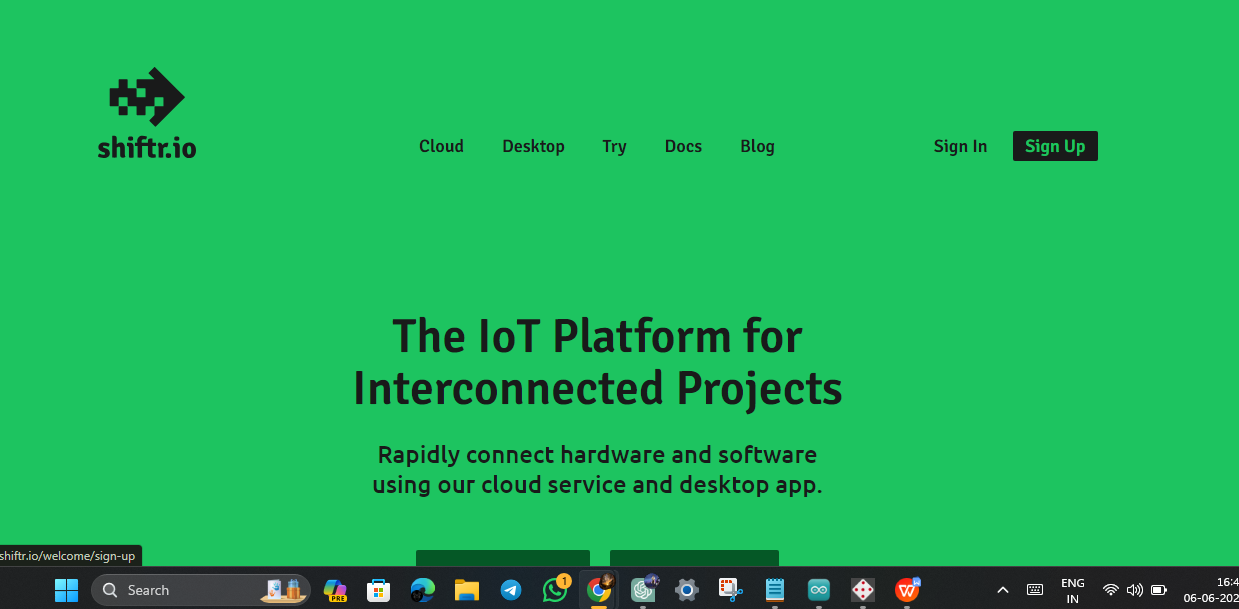


**STEPS TO CREATE A PROJECT IN ARDUINO IOT CLOUD**

**Step 1: Set Up Shiftr.io**

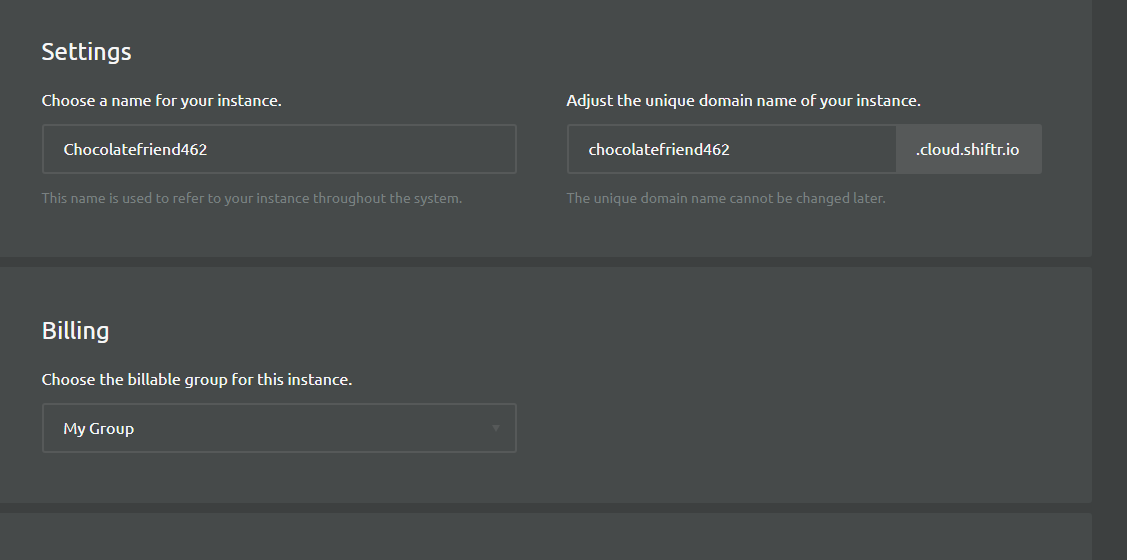
**Create an Account:**

* Visit Shiftr.io and create an account.



**Create a New Space:**

* After logging in, create a new space. This space will be your MQTT broker.



**Get Connection Details:**

* Note the broker URL, username, and password provided by Shiftr.io. You will need these for your ESP32 and MQTT Box setup.

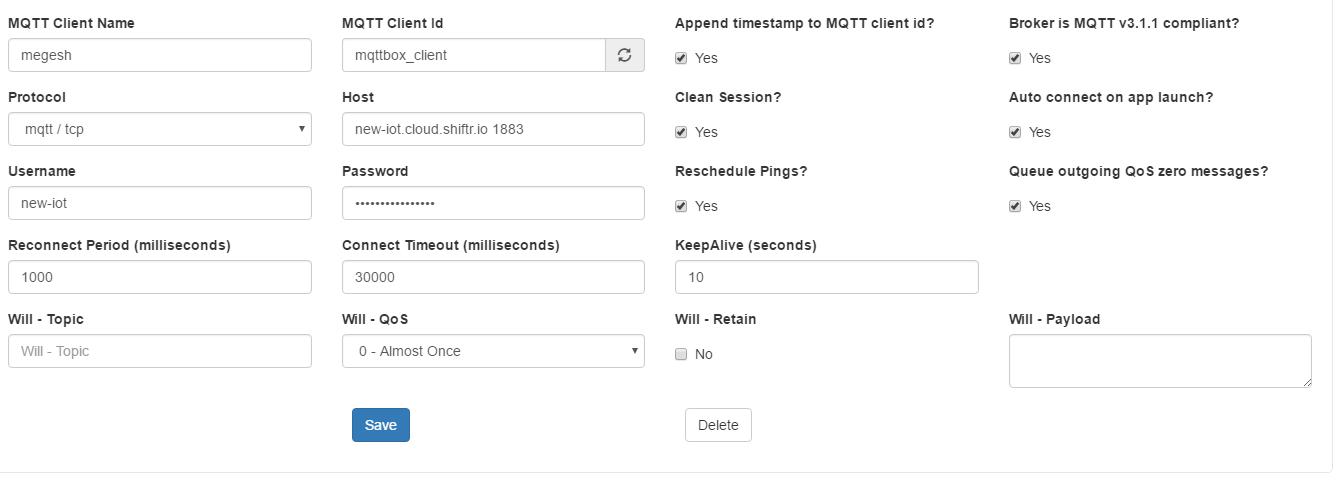
**Step 2: Set Up MQTT Box**

**Install MQTT Box:**

* Download and install MQTT Box from MQTT Box.

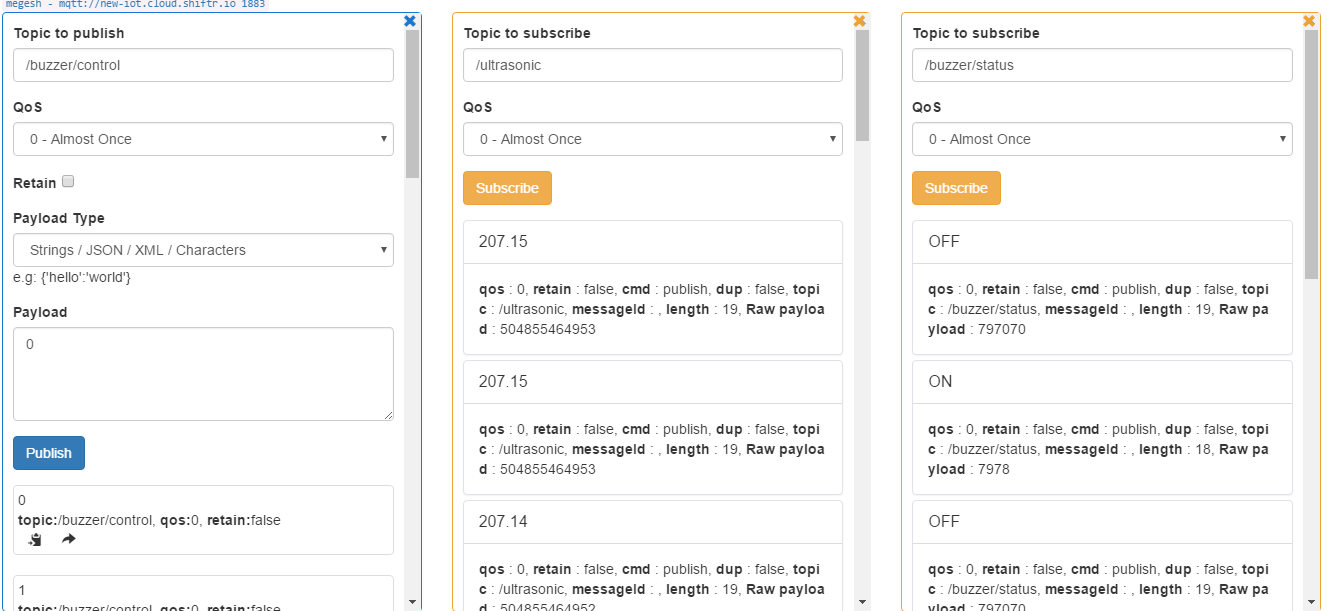
**Create a New Connection:**

* Open MQTT Box and create a new connection.
* Enter the broker URL, username, and password you got from Shiftr.io.



**Subscribe to Topics:**

* Subscribe to the following topics to monitor and control the buzzer:
* /buzzer for controlling the buzzer
* /ultrasonic for receiving distance measurements
* /buzzerStatus for buzzer status updates



**Step 3: Connect Hardware Components**

**Ultrasonic Sensor:**

* Connect the VCC pin to 5V on the ESP32.
* Connect the GND pin to the ground.
* Connect the Trig pin to a digital pin (e.g., D4).
* Connect the Echo pin to another digital pin (e.g., D5).

**Buzzer:**

* Connect the positive terminal to a digital pin (e.g., D26).
* Connect the negative terminal to the ground.
* Step 4: Write the Arduino Code

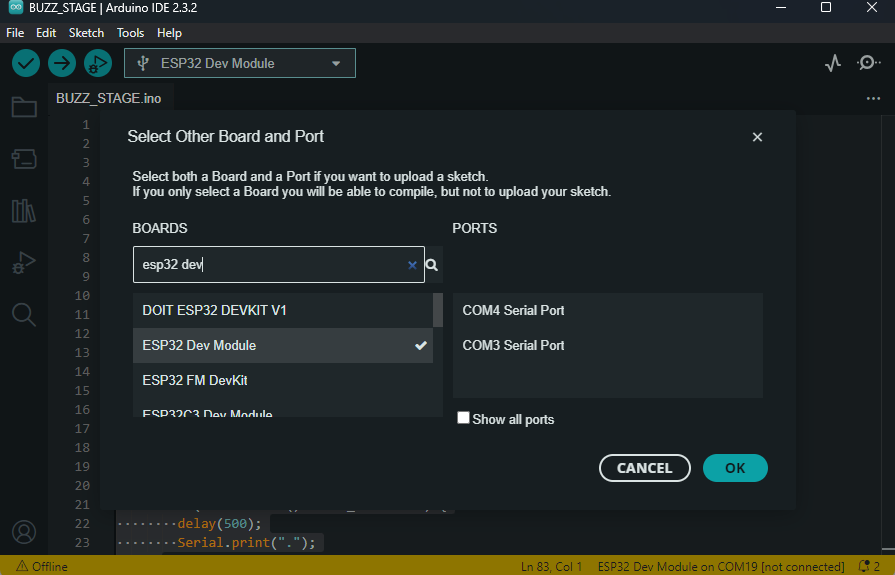
**Libraries:**

* Ensure you have the necessary libraries installed:
* WiFi
* MQTT

**Step 5: Upload Code to ESP32**

**Arduino IDE Setup:**

* Ensure you have the ESP32 board support installed in the Arduino IDE.
* Select the correct board and port from the Tools menu.



**Upload the Code:**

* Connect your ESP32 to the computer via USB.
* Upload the code.

**Step 6: Monitor and Control via MQTT Box**

**Monitor Distance**

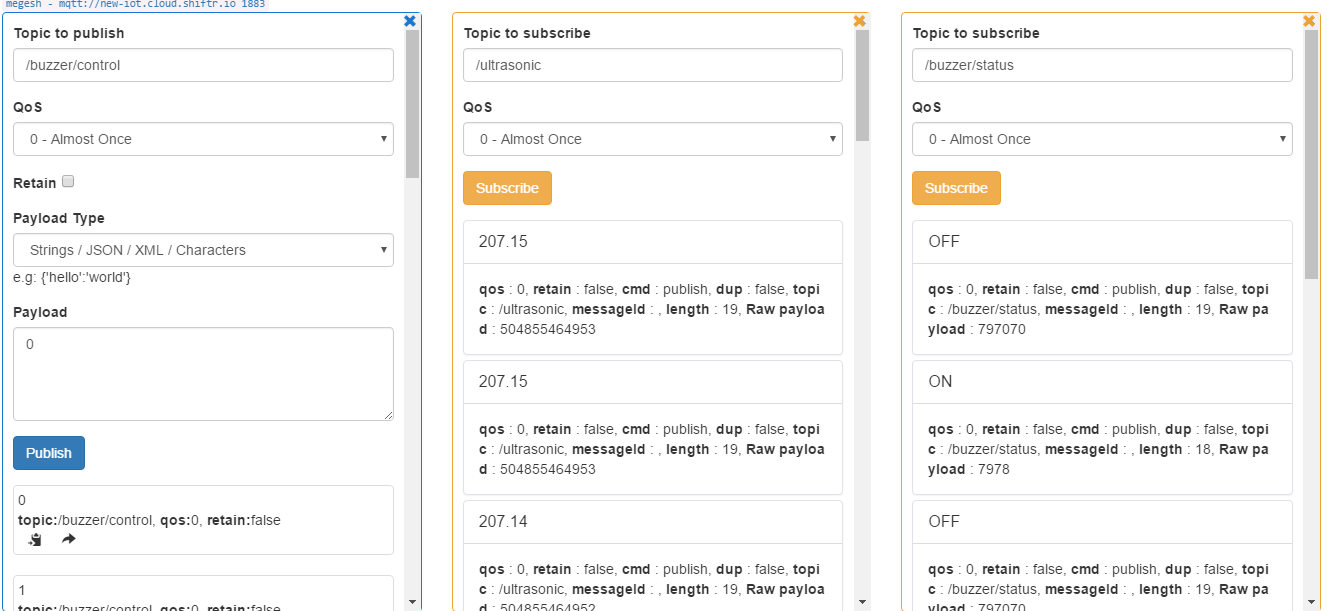
* In MQTT Box, check the messages published to the /ultrasonic topic to see the distance measurements from the ultrasonic sensor.

**Control the Buzzer:**

* Publish "1" to the /buzzer topic to turn the buzzer on.
* Publish "0" to the /buzzer topic to turn the buzzer off.

**Monitor Buzzer Status:**

* Subscribe to the /buzzerStatus topic to receive updates on the buzzer's status (ON or OFF).



**CONCLUSION:**

By following the steps outlined in this guide, you have successfully created an IoT project that integrates the ESP32 microcontroller, an ultrasonic sensor, and a buzzer with Shiftr.io and MQTT Box. This project demonstrates several key concepts and skills:

**WiFi and MQTT Integration:**

* You learned how to connect the ESP32 to a WiFi network and an MQTT broker using the Shiftr.io platform.
* This enables the ESP32 to communicate with other devices and services over the internet, a foundational capability for IoT projects.

**Sensor Data Collection and Publishing:**

* The ultrasonic sensor measures distance, and these measurements are published to the /ultrasonic MQTT topic.
* This illustrates how sensor data can be collected and transmitted in real-time to a central server or broker.

**Remote Actuation:**

* By subscribing to the /buzzer MQTT topic, the ESP32 can receive commands to turn the buzzer on and off.
* This shows how remote control of devices can be implemented using MQTT, allowing for automation and remote management of hardware.

**Monitoring and Feedback:**

* The buzzer's status is published to the /buzzerStatus topic, providing feedback on its current state.
* This ensures that the system's state can be monitored remotely, enhancing reliability and control.

**MQTT Box Usage:**

* You used MQTT Box to monitor messages and control the buzzer, demonstrating a practical tool for testing and debugging MQTT communications