JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

Project based learning report

IoT Communication with MQTT Using Raspberry Pi and ESP8266

Submitted To - Ritesh Kumar Sharma Sir

Submitted By -

Priyansh Yadav (22102075)
Bhaskar Jha (22102078)

Introduction

This project demonstrates the implementation of a lightweight, real-time IoT communication system using the MQTT (Message Queuing Telemetry Transport) protocol. It utilizes a Raspberry Pi as an MQTT broker and an ESP8266 microcontroller as a client, forming a foundational setup for efficient machine-to-machine (M2M) communication. The primary goal is to establish and simulate reliable topic-based messaging between devices over a local network.

The Raspberry Pi, running a full MQTT broker (e.g., Mosquitto), manages all message routing, while the ESP8266 acts as a publisher or subscriber depending on the use case. Topics were created and monitored using both CLI tools and a mobile MQTT client app, allowing for intuitive control and real-time data observation. Additionally, RealVNC was used for remote desktop access to the Raspberry Pi, enabling seamless management and testing from different locations.

Through this setup, I was able to successfully publish and subscribe to topics, visualize message transfers, and simulate IoT scenarios such as sensor data streaming or device control. This project serves as a hands-on introduction to MQTT architecture, and lays the groundwork for future development of smart automation systems, remote monitoring, and IoT-based applications.

Working of MQTT

1. MQTT Architecture:

MQTT follows a publish-subscribe messaging model. It's designed for lightweight, low-bandwidth, and reliable communication between devices, making it ideal for IoT.

It has three core components:

- Broker:
  The central server that receives messages from clients and routes them to other subscribed clients. In your case, this is the Raspberry Pi running the Mosquitto broker.

- Clients:
  Devices (like ESP8266, mobile apps, or other microcontrollers) that connect to the broker to publish or subscribe to topics. Your ESP8266 and mobile MQTT app are MQTT clients.

- Topics:
  Channels or addressable names to which clients publish or subscribe (e.g., "home/led" or "sensor/temperature"). Topics are case-sensitive and hierarchical (e.g., home/room1/light).

2. Communication Flow:

1. Broker Setup (Raspberry Pi):
   You install and run an MQTT broker like Mosquitto. It listens on a port (default is 1883) and handles all client connections.

2. Client Connection (ESP8266):
   - The ESP8266 connects to the broker using its IP address and port.
   - It maintains a persistent TCP connection to send or receive data.

3. Topic Subscription (Mobile App / ESP8266):
   - A client subscribes to a topic (e.g., home/led).
   - The broker keeps track of all clients and their subscribed topics.

4. Publishing a Message:
   - A client (e.g., the mobile app) publishes a message to a topic (home/led = ON).
   - The broker receives this message and checks which clients are subscribed to that topic.

5. Message Distribution:
   - The broker forwards the message to all subscribed clients.
   - For example, the ESP8266 subscribed to home/led will receive the "ON" command and can act on it (e.g., turn on an LED).

3. Key Features of MQTT:

- Lightweight Protocol:
  Small packet sizes, perfect for constrained devices like ESP8266.

- Quality of Service (QoS) Levels:
  MQTT allows three QoS levels to manage reliability:

  - 0 – At most once (fire and forget)

  - 1 – At least once (acknowledged delivery)

  - 2 – Exactly once (guaranteed, but slowest)

- Last Will and Testament (LWT):
  If a client disconnects unexpectedly, the broker can publish a predefined message to a topic.

- Retained Messages:
  A message can be retained on a topic so that new subscribers immediately receive the last known value.

- Persistent Sessions:
  Clients can maintain session data (like subscriptions) across reconnects.

Why Use MQTT for IoT Projects?

- Efficient bandwidth usage

- Low power consumption

- Reliable delivery

- Scalable for thousands of devices

In Our Project:

- Broker: Raspberry Pi running Mosquitto
- Clients: ESP8266 (possibly running PubSubClient), Mobile App (like MQTT Dash or MQTT Explorer)
- Tools: CLI for topic testing, RealVNC for remote broker control
- Simulation: Example - Mobile app publishes Message to test/topic, ESP8266 receives message.

Steps to Setup a MQTT Network

Set Up Raspberry Pi as MQTT Broker

1. Install Raspberry Pi OS

- Download Raspberry Pi Imager from: https://www.raspberrypi.com/software/
- Insert the microSD card into your PC.
- Use Raspberry Pi Imager to install Raspberry Pi OS Lite (for headless) or Full (with GUI).
- Optionally configure:
    - Enable SSH
    - Set Wi-Fi SSID and password
- Flash the OS onto the microSD card.
- Insert the microSD card into Raspberry Pi and power it up.

2. Access Raspberry Pi (CLI or GUI)

- Use:
    - HDMI + Keyboard for direct access

OR SSH via Terminal:

- ssh pi@<raspberry_pi_ip_address>
- OR RealVNC if you installed the desktop version.

3. Update and Upgrade Raspberry Pi

- sudo apt update
- sudo apt upgrade -y

4. Install MQTT Broker (Mosquitto)

- sudo apt install mosquitto mosquitto-clients -y
- sudo systemctl enable mosquitto

5. Test Mosquitto Locally (Optional)

Open two terminals:

- Terminal 1 (Subscriber):
  mosquitto_sub -h localhost -t test/topic

- Terminal 2 (Publisher):
  mosquitto_pub -h localhost -t test/topic -m "Hello from Pi!"

You should see the message on the subscriber terminal.

Set Up ESP8266 as MQTT Client

6. Install Arduino IDE and ESP8266 Board Support

- Download Arduino IDE from https://www.arduino.cc/en/software
- In Arduino IDE:
    - Go to File → Preferences

In *Additional Board URLs*, add:

- http://arduino.esp8266.com/stable/package_esp8266com_index.json
- Go to Tools → Board Manager, search for ESP8266, and install it.

7. Install MQTT Library

- In Arduino IDE: Sketch → Include Library → Manage Libraries
- Search and install: PubSubClient by Nick O'Leary

8. Connect ESP8266 to MQTT Broker

Upload a simple sketch:

cpp
CopyEdit

```cpp
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

// WiFi credentials
const char* ssid = "pc";
const char* password = "vtvg2883";

// MQTT broker details
const char* mqtt_server = "192.168.163.149";            //"192.168.29.27";
// Replace with your broker's IP address
const int mqtt_port = 1883; // Default port
const char* mqtt_topic = "test/topic"; // Topic to subscribe to
```

```
WiFiClient espClient;
PubSubClient client(espClient);

// Callback function to handle incoming MQTT messages
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived on topic: ");
  Serial.println(topic);
  Serial.print("Message: ");

  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();
}

void setup_wifi() {
  delay(10);
  pinMode(D0, OUTPUT);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void reconnect() {
  // Loop until the ESP8266 is connected to the MQTT broker
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("ESP8266Client")) {
      Serial.println("connected");
      client.subscribe(mqtt_topic);
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
```

```
      Serial.println(" trying again in 5 seconds");
      delay(5000);
    }
  }
}

void setup() {
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
}
```

- Use Serial Monitor to see connection status.
- This sets up the ESP8266 to connect to your Raspberry Pi's MQTT broker and subscribe to a topic like test/topic.

Connect with MQTT Mobile App

- Install MQTT Dash, MQTT Explorer, or IoT MQTT Panel
- Add a new broker with:
    - IP: Raspberry Pi's local IP
    - Port: 1883
- Publish messages to topics like test/topic
- Observe ESP8266 reacting

Pictures

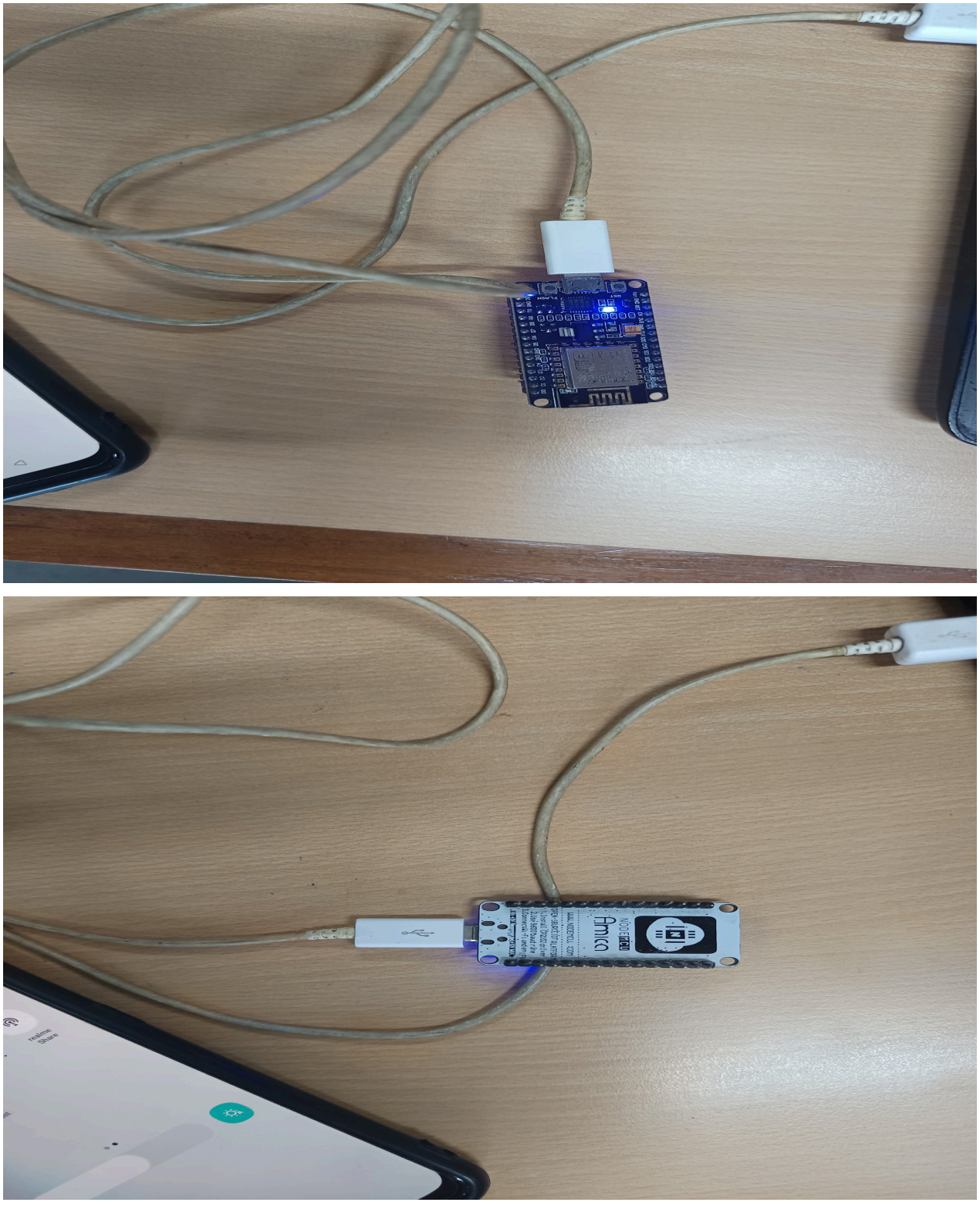

Figure 1.1 , 1.2 - ESP8266 Module

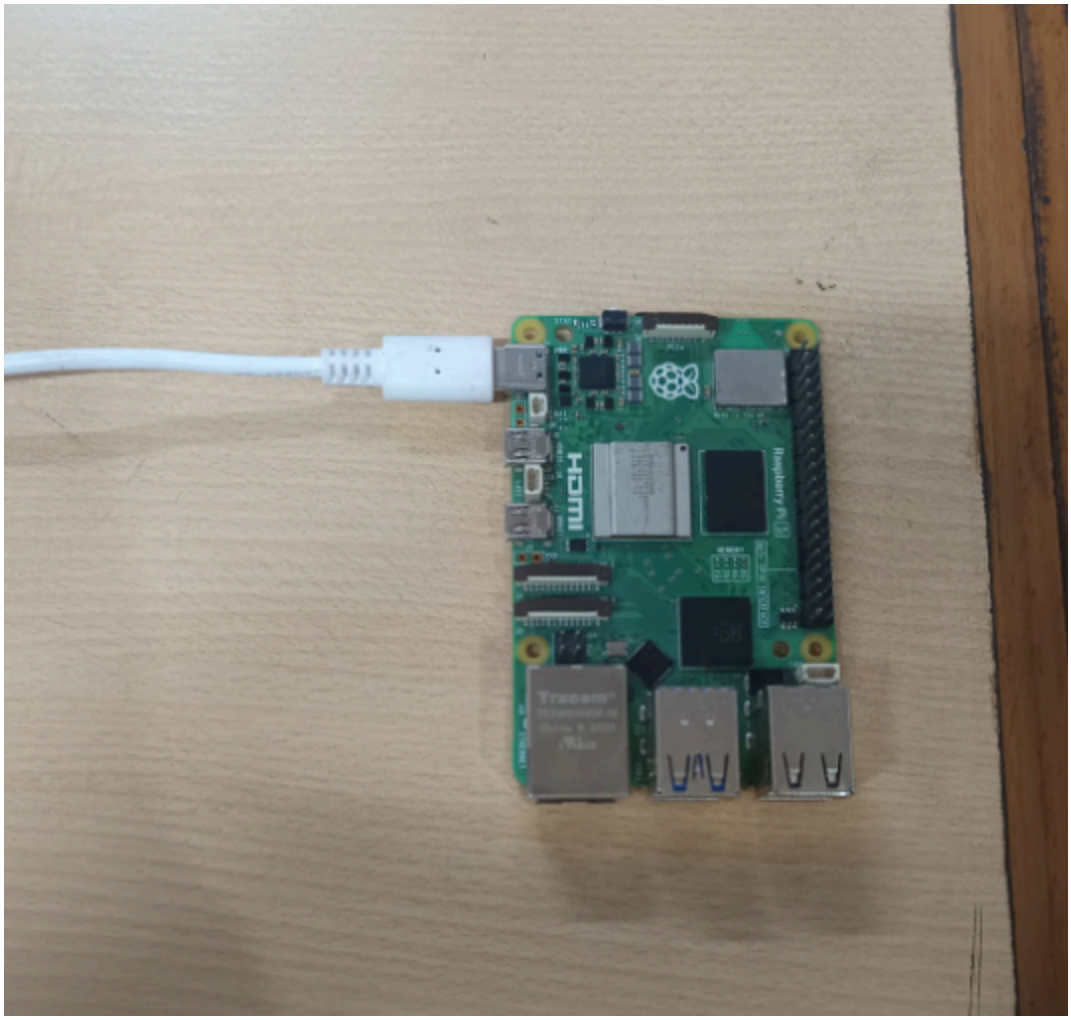

Figure 2 - Raspberry Pi

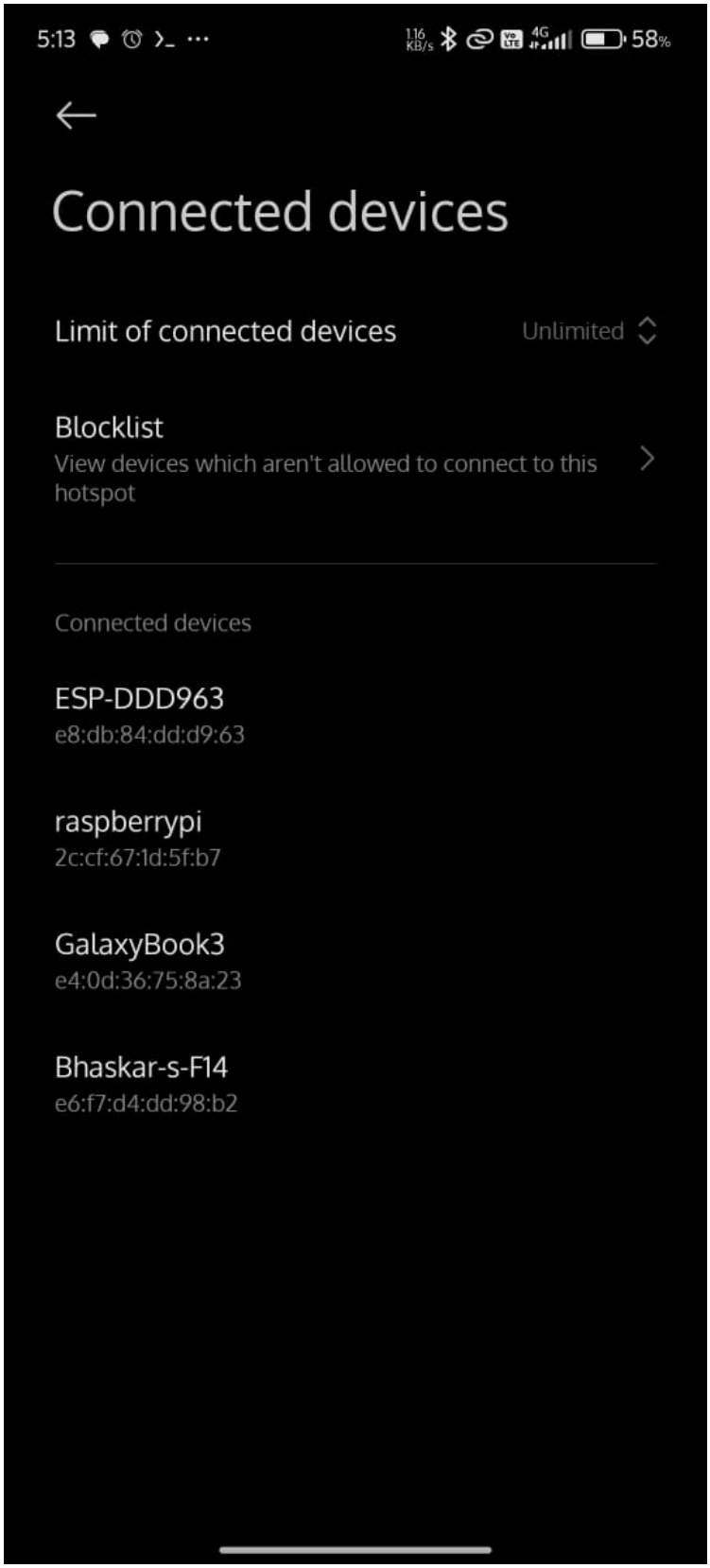

Figure 3 - Mobile Hotspot Connected Devices

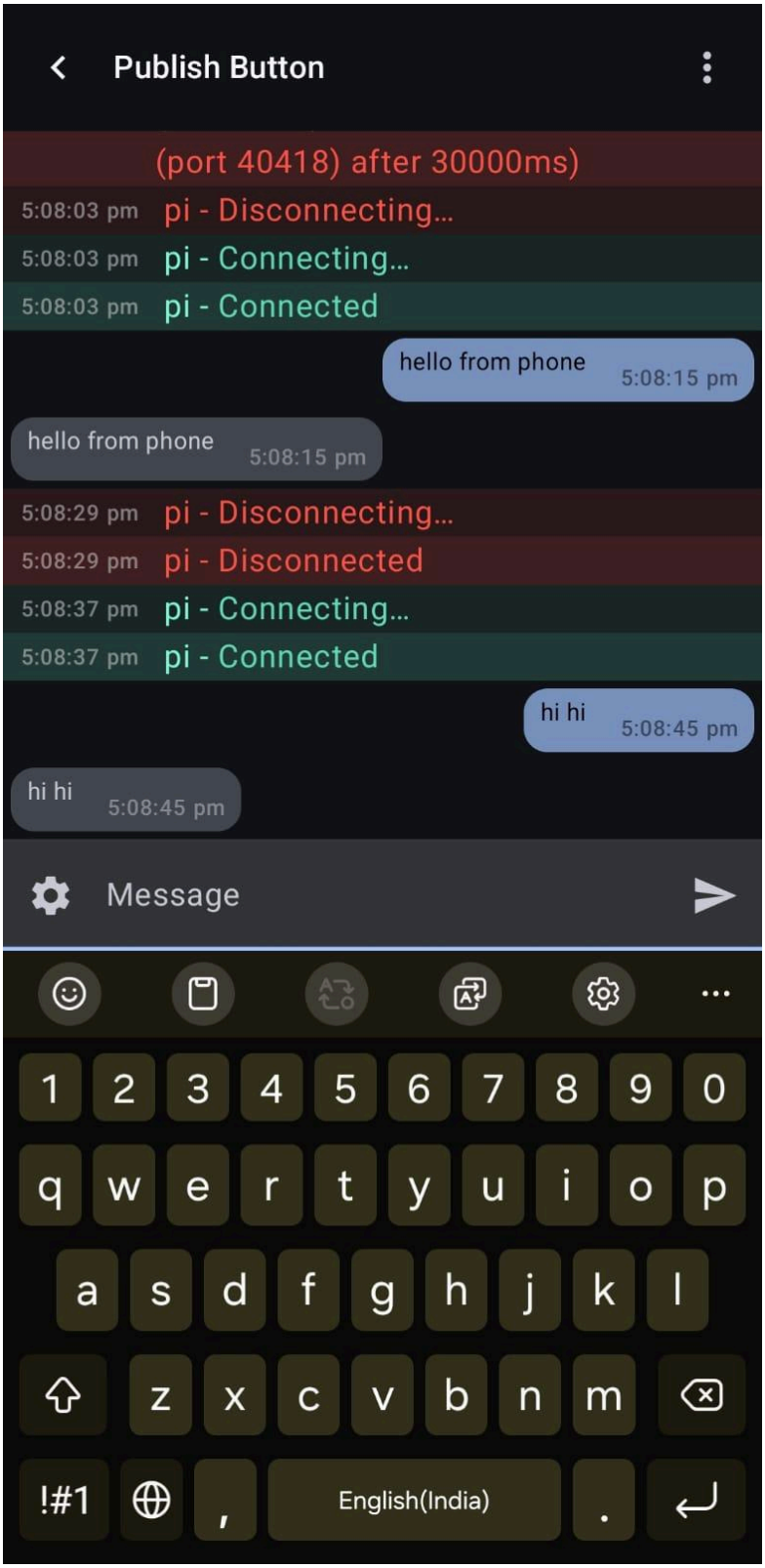

Figure 4 - MQTT Dashboard App

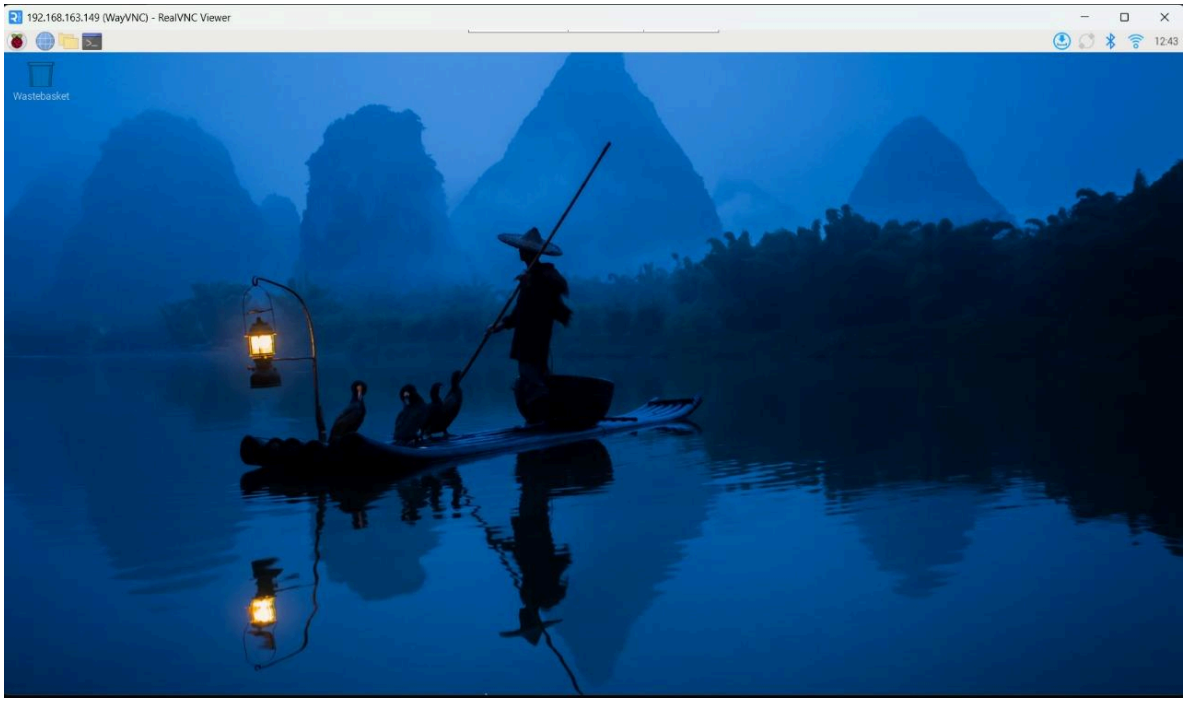

Figure 5 - RealVNC View of Raspberry OS

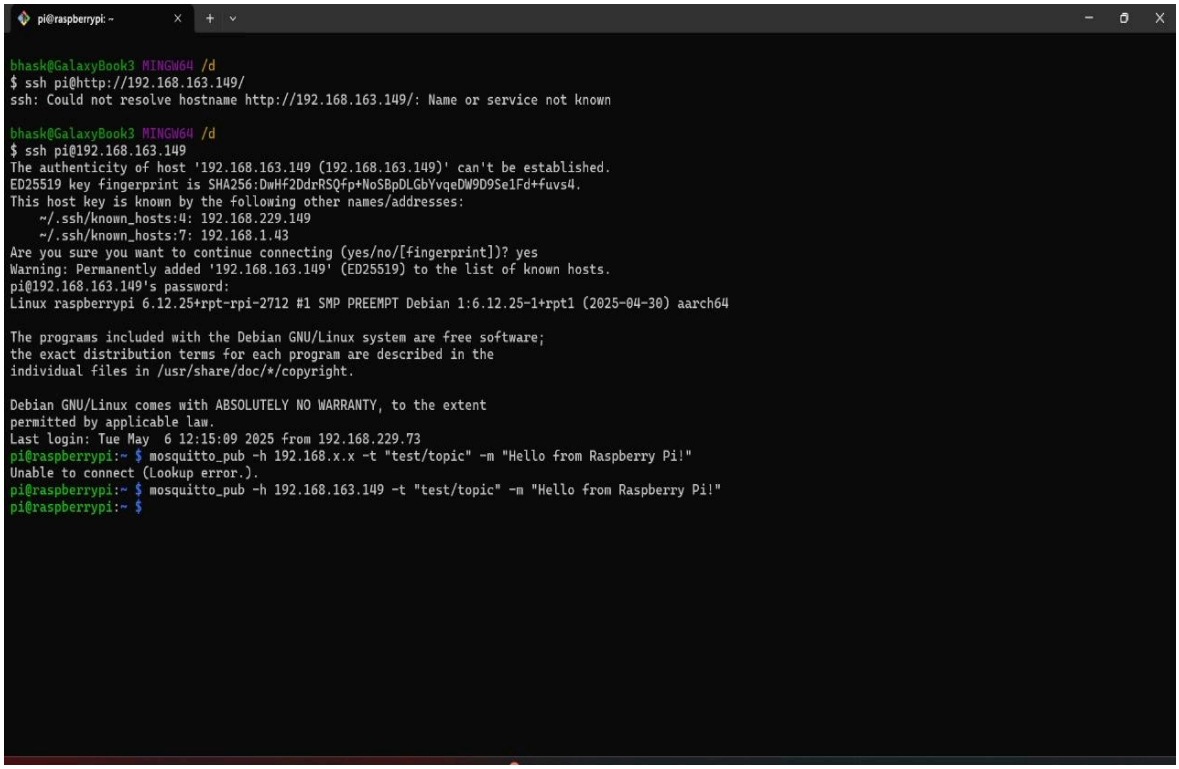

Figure 6 - Raspberry Pi Terminal

```
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Attempting MQTT connection...connected
Attempting MQTT connection...connected.............
WiFi connected
IP address:
192.168.163.96
Attempting MQTT connection...connected
Message arrived on topic: test/topic
Message: Hello from Raspberry Pi!
Message arrived on topic: test/topic
Message: hello from phone
Message arrived on topic: test/topic
Message: hi hi
```

Figure 7 - Serial Monitor of ESP8266