EFY TESTED
Ashwini K. Sinha

IoT RELAY TO CONTROL APPLIANCES
Through The Internet Using Smartphone

PRADIPTA GHOSH, CHALLA NAGA SAI RAVALI

Electrical appliances such as lights, fans, air-conditioners, TVs, and water pumps are typically operated using manual switches. This system enables these appliances to be controlled over the internet using a smartphone. It combines an ESP-01 Wi-Fi module, the Switchsys MQTT broker (free public version), and the IoT MQTT panel app. The Switchsys MQTT broker implements the MQTT publish/subscribe communication model, enabling communication between the ESP-01 module and the user interface.

The IoT MQTT panel mobile application allows users to send on/off commands and monitor device status in real time from any location with internet access. Commands sent through the app are routed via the MQTT broker to the ESP-01, which processes the subscribed topic and drives the relay accordingly. This enables remote control of connected appliances from virtually anywhere. Fig. 1 shows the author's prototype, while Table 1 lists the components required to build the system.

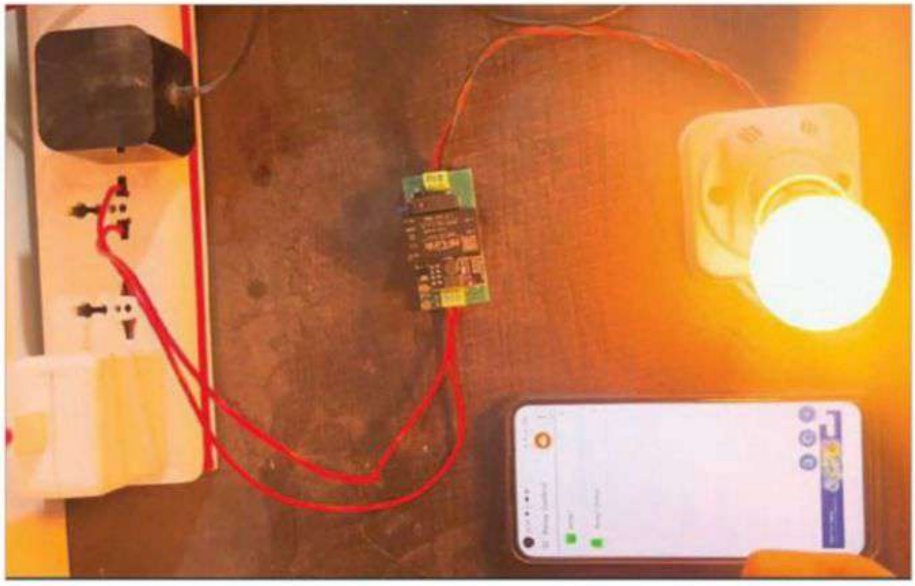

Fig. 1: Authors' prototype

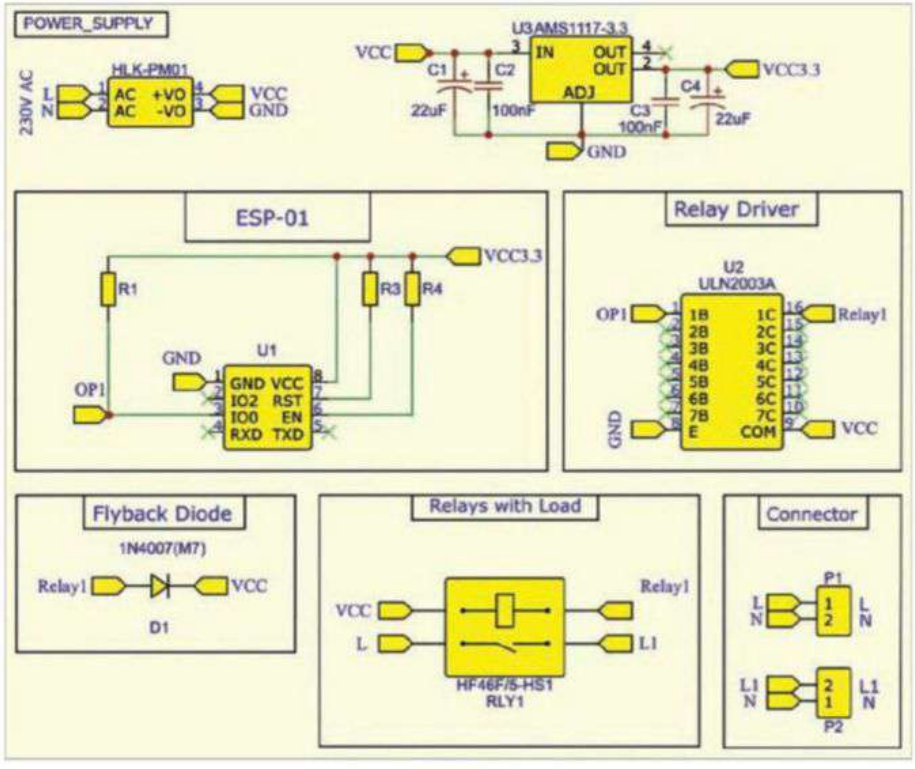

Fig. 2: Circuit diagram

TABLE 1
BILL OF MATERIALS

Components	Quantity
ESP-01 Wi-Fi module (ESP8266 SoC based)	1
8-pin ESP-01 connector	1
ESP-01 USB programmer adaptor	1
Relay driver IC ULN2003A (U2)	1
Relay HF46F/5-HS1	1
Diode 1N4007 (M7)	1
HLK-PM01 230V AC to 5V DC converter	1
10kΩ resistor (R1-R3)	3
2-pin 5.08mm pitch plug-in screw terminal block connector	2
AMS1117-3.3 voltage regulator - U3 (5V DC to 3.3V DC)	1
100nF ceramic capacitor (C2, C3)	2
22μF tantalum capacitor (C1, C4)	2

TABLE 2
MQTT PUBLISH/SUBSCRIBE DETAILS

Component	Type	Action	Topic	Payload	Function
IoT MQTT app	Publish	Relay switch on	Switchsys/relay1	1	Energises relay
IoT MQTT app	Publish	Relay switch off	Switchsys/relay1	0	De-energises relay
IoT MQTT app	Subscribe	—	Switchsys/relay1/status	1	LED indicator turns green
IoT MQTT app	Subscribe	—	Switchsys/relay1/status	0	LED indicator turns grey
ESP-01	Subscribe	—	Switchsys/relay1	1	Turns relay on
ESP-01	Subscribe	—	Switchsys/relay1	0	Turns relay off
ESP-01	Publish	Relay energised	Switchsys/relay1/status	1	Sends on status to app
ESP-01	Publish	Relay de-energised	Switchsys/relay1/status	0	Sends off status to app

```
// GPIO0 for ESP-01
// NOTE: GPIO0 MUST be HIGH at boot, otherwise ESP-01 will not start
#define Relay1 0
```

Fig. 3: Relay control pin configuration

```
// ================ WIFI CREDENTIALS ================
const char* ssid = "xxxxx";       // WiFi SSID to be updated as per user
const char* password = "xxxxx";   // WiFi Password to be updated by user

WiFi.begin(ssid, password);       // Start WiFi connection
```

Fig. 4: Wi-Fi configuration

```
// ================ SWITCHSYS MQTT CONFIG ================
const char* mqttServer = "mqtt.switchsys.in"; // MQTT Broker address

// Unique MQTT client ID (must be unique on broker)
String clientID = "Switchsys_relay1_1234";

#define sub "switchsys/relay1"          // Subscribe topic (control)
#define pub "switchsys/relay1/status"   // Publish topic (status)
```

Fig. 5: MQTT broker configuration

```
client.setServer(mqttServer, 1883);
```

Fig. 6: MQTT initialisation

```
// If payload is '1' → Relay ON
if ((char)payload[0] == '1') {
    digitalWrite(Relay1, HIGH);      // Relay ON
    client.publish(pub, "1");        // Publish ON status
}

// If payload is '0' → Relay OFF
else if ((char)payload[0] == '0') {
    digitalWrite(Relay1, LOW);       // Relay OFF
    client.publish(pub, "0");        // Publish OFF status
}
```

Fig. 7: Displays the current relay state

The system uses an ESP-01 Wi-Fi module, a relay module, the Switchsys MQTT broker, and the IoT MQTT Panel app. It becomes operational when powered on. The ESP-01 connects to the Wi-Fi network and communicates with the MQTT broker. The user sends on/off commands through the mobile app, which the ESP-01 receives via the broker. Based on the received command, the ESP-01 activates or deactivates the relay to switch the connected bulb on or off.

System design

In the IoT MQTT Panel app dashboard, there is a single switch to control the relay and an LED indicator to display the relay status.

When Relay1 switch is turned on, the app publishes payload 1 to the topic switchsys/relay1 and sends it to the ESP-01 over the internet via the Switchsys MQTT broker (broker URL: mqtt.switchsys.in, port: 1883). Since the ESP-01 is subscribed to the same topic (switchsys/relay1), it receives payload 1 and sets GPIO0 HIGH, thereby energising the relay.

After energising the relay, the ESP-01 publishes the relay status to the topic switchsys/relay1/status. This status message is then sent back to the IoT MQTT Panel app through the MQTT broker. In the app, the LED indicator is subscribed to the topic switchsys/relay1/status. Therefore, when payload 1 is received, the LED indicator turns green, indicating that the relay is on.

Similarly, when the relay switch is turned off in the IoT MQTT Panel app, payload 0 is transmitted to the ESP-01 via the topic switchsys/relay1. The ESP-01 then sets GPIO0 LOW, de-energising the relay, and publishes the updated relay status with payload 0 to the topic switchsys/relay1/status. Upon receiving this payload, the LED indicator changes to grey, indicating that the relay is off. Table 2 shows the MQTT topics and payload details used in the project.

Circuit diagram

Fig. 2 shows the circuit diagram of the system. It is built around the ESP-01 Wi-Fi module, the AMS1117-3.3 voltage regulator (5V DC to 3.3V DC), the ULN2003A relay driver IC, the HLK-PM01 230V AC to 5V DC converter module, the HF46F/5-HS1 relay, the 1N4007 diode, and a few other components.

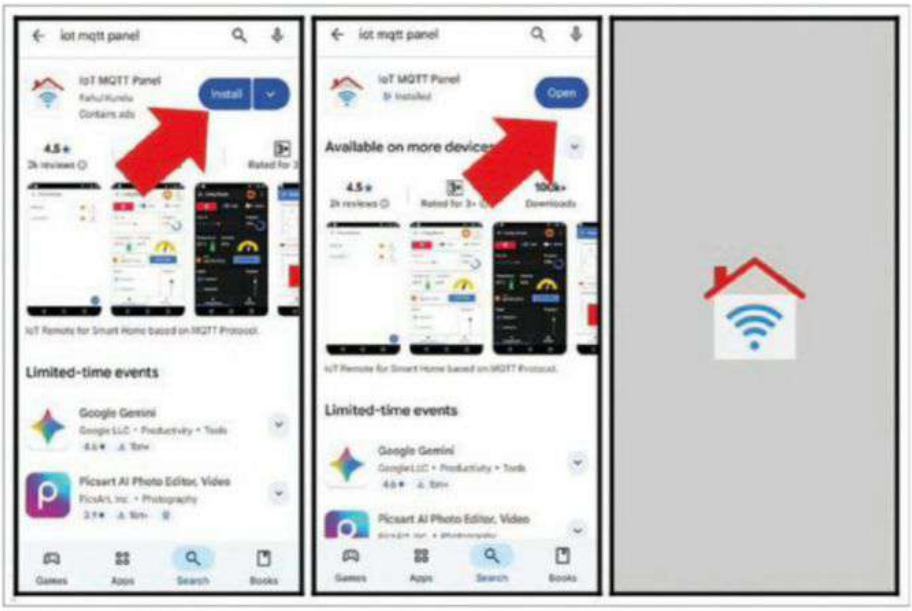

Fig. 8. Installing the IoT MQTT panel app

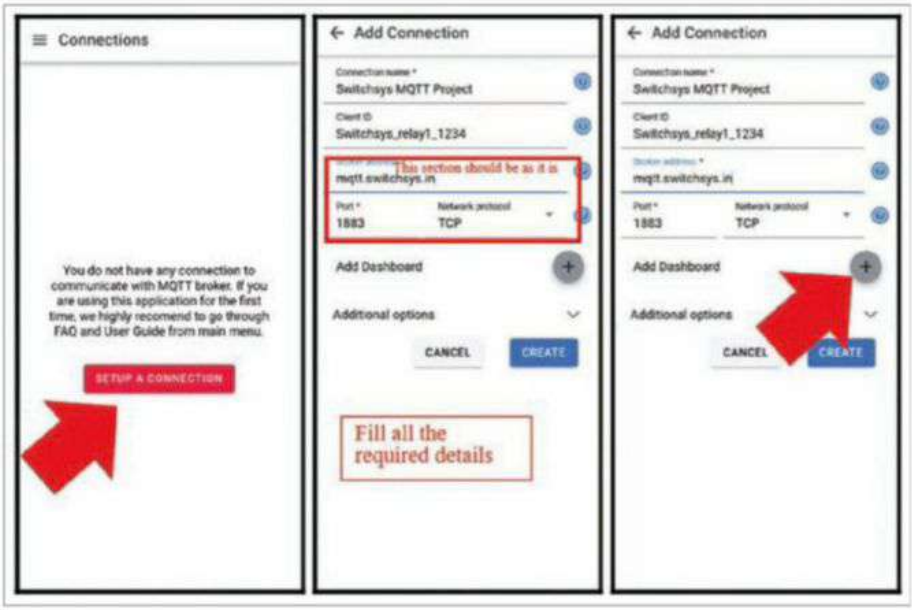

Fig. 9: MQTT configuration

EFY note. This system uses the ESP-01 module, and the GPIO pins are selected accordingly. Other ESP8266-based boards, such as NodeMCU and WeMos D1 Mini, can also be used with this system. In that case, only the ESP-01 module needs to be replaced, and the GPIO pin connections should be modified accordingly.

Software

The first step is to configure the MQTT broker used for communication between the IoT MQTT mobile application and the ESP-01 module. In this system, the Switchsys MQTT Broker, a free public MQTT broker designed for learning, testing, and IoT prototyping, is used.

The broker supports standard MQTT communication over TCP using port 1883 and does not require authentication, making it suitable for learning, testing, and evaluating IoT systems without the need for dedicated server infrastructure. The broker details used in this system are as follows:

Broker URL: mqtt.switchsys.in and Port: 1883 using non-TLS MQTT communication.

Next, the firmware must be prepared for the ESP module. In this system, the ESP-01 module is used. The ESP-01 is a compact, low-cost Wi-Fi module based on the ESP8266 System-on-Chip (SoC), which includes an integrated TCP/IP protocol stack. Due to its small size, built-in Wi-Fi capability, and ease of programming, it is widely used in standalone IoT applications. The module comes with 1MB flash memory and provides two usable GPIO pins, namely GPIO0 and GPIO2. The ESP-01 can be programmed using the Arduino IDE with the help of an ESP-01 USB Programmer Adaptor.

In this system, the ESP-01 receives commands from the IoT MQTT application through the Switchsys MQTT broker and controls the relay connected via the ULN2003A relay driver circuit. As shown in the circuit diagram in Fig. 2, GPIO0 is used as the relay control pin. The source code can be downloaded by scanning the accompanying QR code. After downloading, open the file named IoT_Smart_Relay_Switchsys_MQTT_IoT_MQTT_Panel.ino in the Arduino IDE. In the code, the relay control pin, Wi-Fi credentials, and MQTT settings must be configured according to the network and application requirements.

The relay control pin configuration used in this system is shown in Fig. 3.

Note. This system is designed using the ESP-01 module, and the GPIO pin selection is based accordingly. Other ESP8266-based boards, such as NodeMCU or WeMos D1 Mini, can also be used. In that case, the GPIO pin definitions must be modified according to the selected development board.

The Wi-Fi credentials section of the code must also be updated according to the local Wi-Fi network being used. The relevant code section for Wi-Fi configuration is shown in Fig. 4.

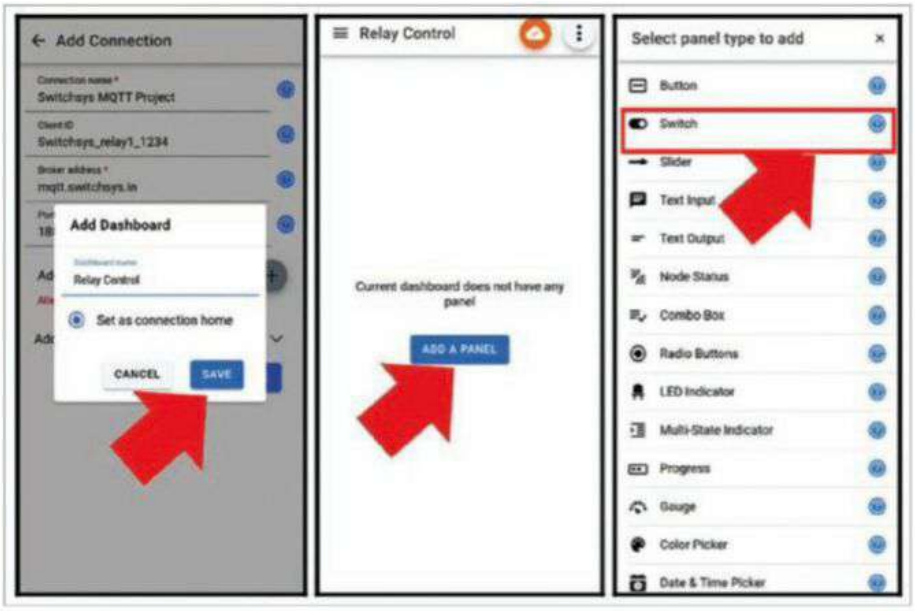

Fig. 10: MQTT dashboard preparation

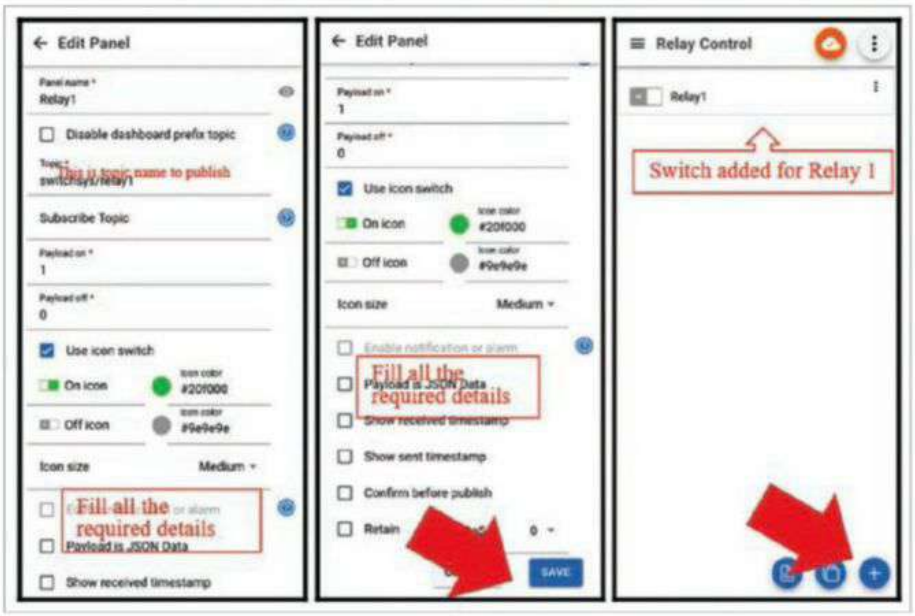

Fig. 11: MQTT panel settings

After configuring the Wi-Fi settings, the MQTT broker configuration must be updated. The MQTT server address, unique client ID, subscription topic, and publish topic are defined in the code, as shown in Fig. 5.

Unique client IDs, subscription topics, and publish topics should be used because this system relies on a public MQTT broker that may be accessed by multiple users simultaneously. Using unique topics helps prevent unwanted interference or accidental control by other users connected to the same broker.

The MQTT server connection is initialised in the code, as shown in Fig. 6.

Once the ESP-01 successfully connects to the Wi-Fi network and the MQTT broker, it continuously listens for incoming MQTT messages on the subscribed topic. When commands are received, the ESP-01 interprets the payload and controls the relay through the ULN2003A relay driver IC. If the payload contains 1, the relay is energised and switched on. If the payload contains 0, the relay is de-energised and switched off. At

Fig. 12: Relay status

the same time, the relay status is published back to the MQTT broker, enabling the IoT application to display the current relay state. See Fig. 7.

After completing all the required configurations, the code can be compiled and uploaded to the ESP-01 module using the ESP-01 USB Programmer Adaptor. For other ESP8266 development boards, such as NodeMCU or WeMos D1 Mini, a separate programmer is not required because these boards already include an onboard USB-to-serial converter for direct programming through the USB port.

IoT MQTT panel app configuration

The IoT MQTT Panel app is a highly customisable IoT dashboard application based on the MQTT protocol. It enables users to visualise and control various inputs and outputs within an IoT application. The app is available on Android and iOS. The configuration steps are shown in Figs. 8 to 12.

Fig. 13: Input and output connections for the appliance

Construction and testing

Assemble the circuit according to the circuit diagram shown in Fig. 2. The circuit includes an onboard HLK-PM01 module that converts a 230V AC mains supply to 5V DC. Therefore, appropriate precautions against electric shock must be taken while handling and testing the system.

After uploading the code to the ESP-01 and assembling the circuit, the prototype shown in Fig. 1 is ready for testing. Open the IoT MQTT panel app to control the relay through the MQTT broker. Once the ESP-01 successfully connects to the Wi-Fi network and MQTT server, the relay can be controlled directly from the application dashboard. The switch button in the app interface can be used to toggle the relay on and off remotely through MQTT communication.

When the switch in the application is turned on, the MQTT app publishes the command 1 to the subscribed MQTT topic. The ESP-01 receives this command through the Switchsys MQTT broker and energises the relay via the ULN2003A relay driver circuit. Similarly, when the switch is turned off, the app sends the command 0, and the ESP-01 de-energises the relay to turn the connected appliance off. The updated relay status is also published back to the MQTT broker so that the application dashboard can display the current relay state in real time.

After verifying proper operation through the app, the AC mains input and the electrical appliance can be connected to the relay terminals. The AC supply should be connected to the input terminal, while the electrical load or appliance should be connected to the control terminal, as shown in Fig. 13. Once connected, the appliance can be switched on and off wirelessly using the IoT MQTT panel application through the internet or a local Wi-Fi network, depending on MQTT connectivity.

Future scope

The system can be further enhanced with security features, feedback sensing, and automation scheduling, making it a practical and efficient IoT-based smart control solution. EFY

TO DOWNLOAD SOURCE CODE AND RESOURCES

Scan this QR code or go to https://sourcecode.electronicsforu.com to access and download source code and other resources for this project

Pradipta Ghosh holds an M.E. in Electronics and Instrumentation. He is passionate about electronics, MCU-based embedded systems, and IoT devices.

Challa Naga Sai Ravali holds an M.Tech degree in Communication and Signal Processing. She is an embedded hardware design engineer with experience in embedded systems, FPGA-based design, IoT solutions, and telecom product development. Her areas of interest include hardware design, wireless communication, and connected technologies.