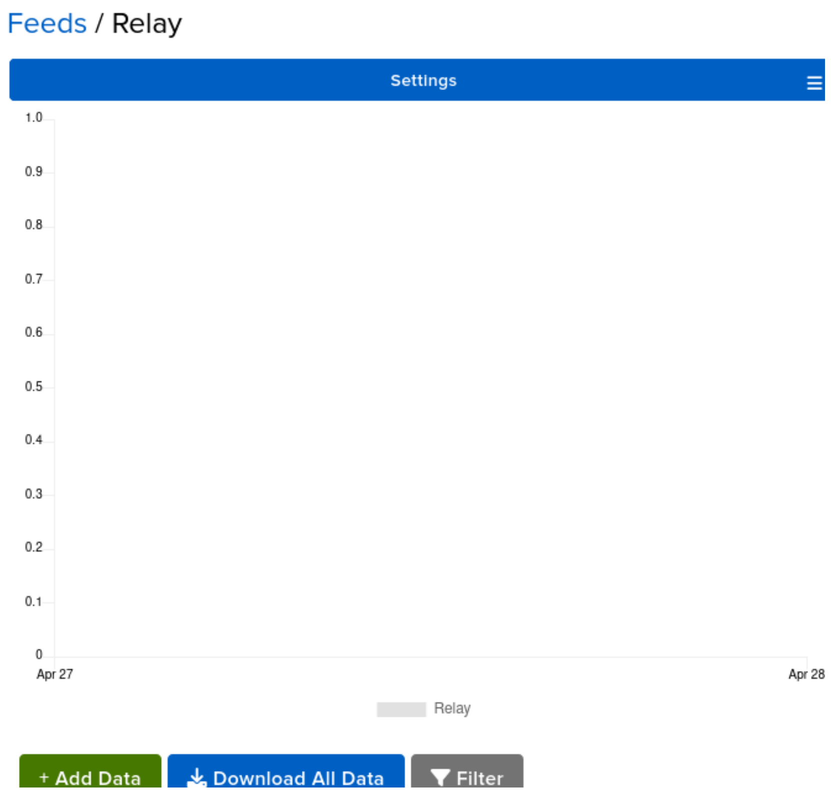
Serverless code can be triggered by many different things, including HTTP requests. You can use HTTP triggers to add a manual override to your relay control, allowing someone to turn the relay on or off from a web request.

For this assignment, you need to add two HTTP triggers to your Functions App to turn the relay on and off, reusing what you have learned from this lesson to send commands to the device.

<u>Solution</u>

In this exercise, I will use Adafruit IO with MQTT to control a 5V Relay through the Adafruit IO dashboard. (replace Azure IoT Hub)

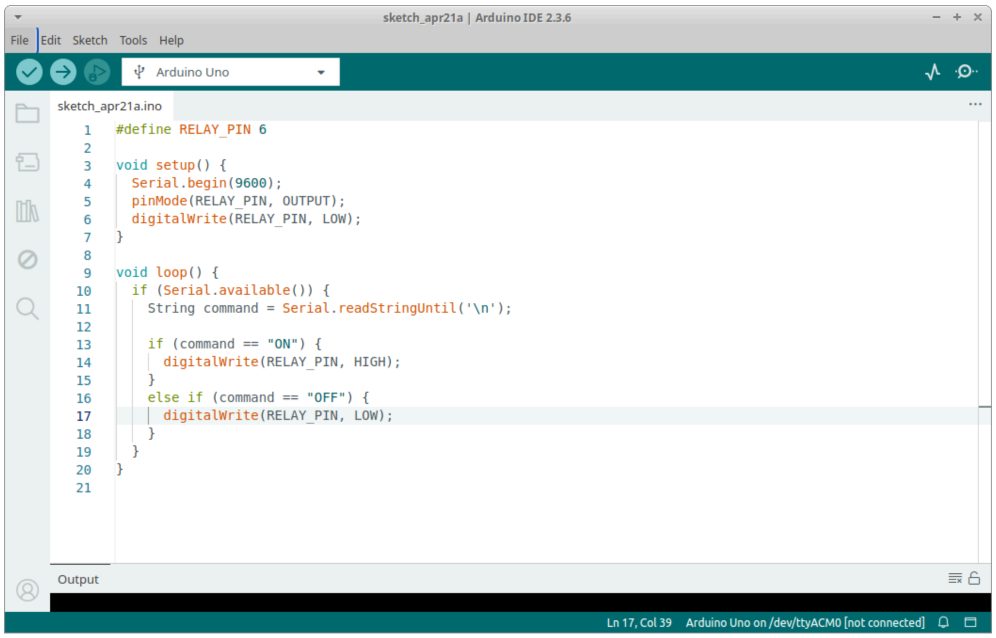
1.Create the Feed on adafruit IO to connect to 5V Relay via Arduino Uno



This feed is used to connect Adafruit and the signal emitted from the 5V Relay. After creating Feed, we will have a path to write in the code later, the path will be in the form: `nguyenlamminhhoa/feeds/relay`

2.Upload code to Arduino Uno

To make the 5v relay work and receive On/Off signals from Adafruit via MQTT, we have to upload code to the main Arduino board.

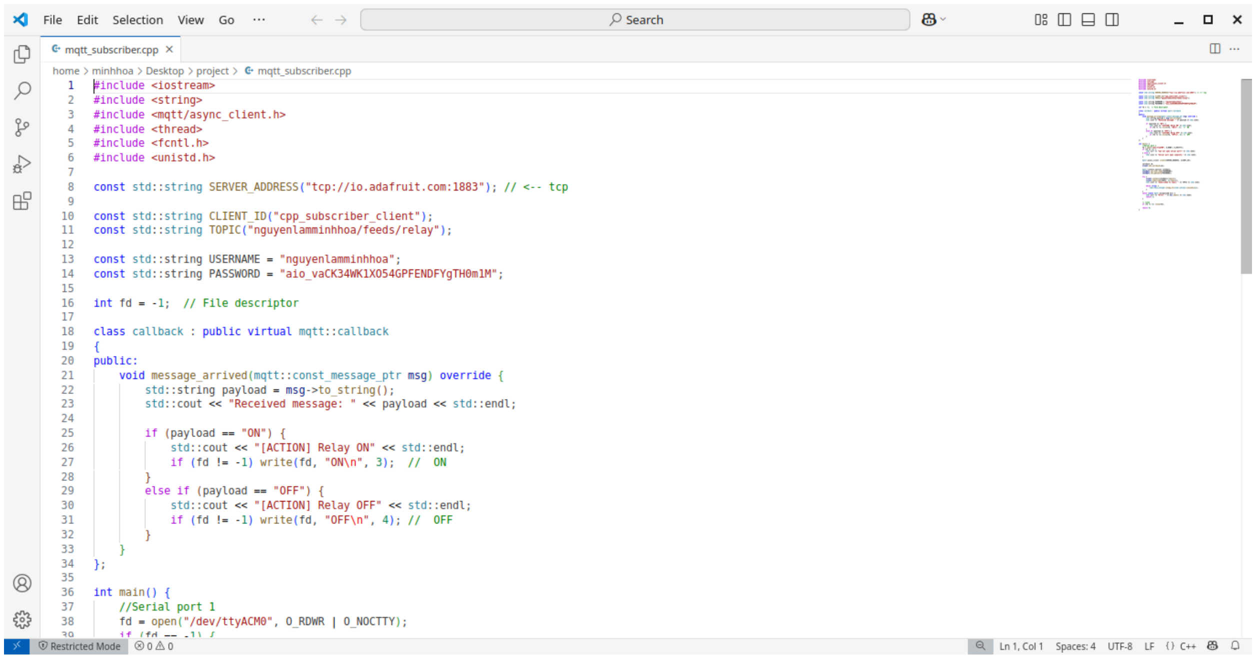

sketch_apr21a | Arduino IDE 2.3.6
File Edit Sketch Tools Help
Arduino Uno
sketch_apr21a.ino
Output
Ln 17, Col 39 Arduino Uno on /dev/ttyACM0 [not connected]

```
#define RELAY_PIN 6

void setup() {
  Serial.begin(9600);
  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, LOW);
}

void loop() {
  if (Serial.available()) {
    String command = Serial.readStringUntil('\n');

    if (command == "ON") {
      digitalWrite(RELAY_PIN, HIGH);
    }
    else if (command == "OFF") {
      digitalWrite(RELAY_PIN, LOW);
    }
  }
}
```

In the above code, the Relay signal pin is connected to D6 of Arduino, + pin for 5V and - pin for GND. After completing the connection, we will upload the code to Arduino Uno to make the connection.

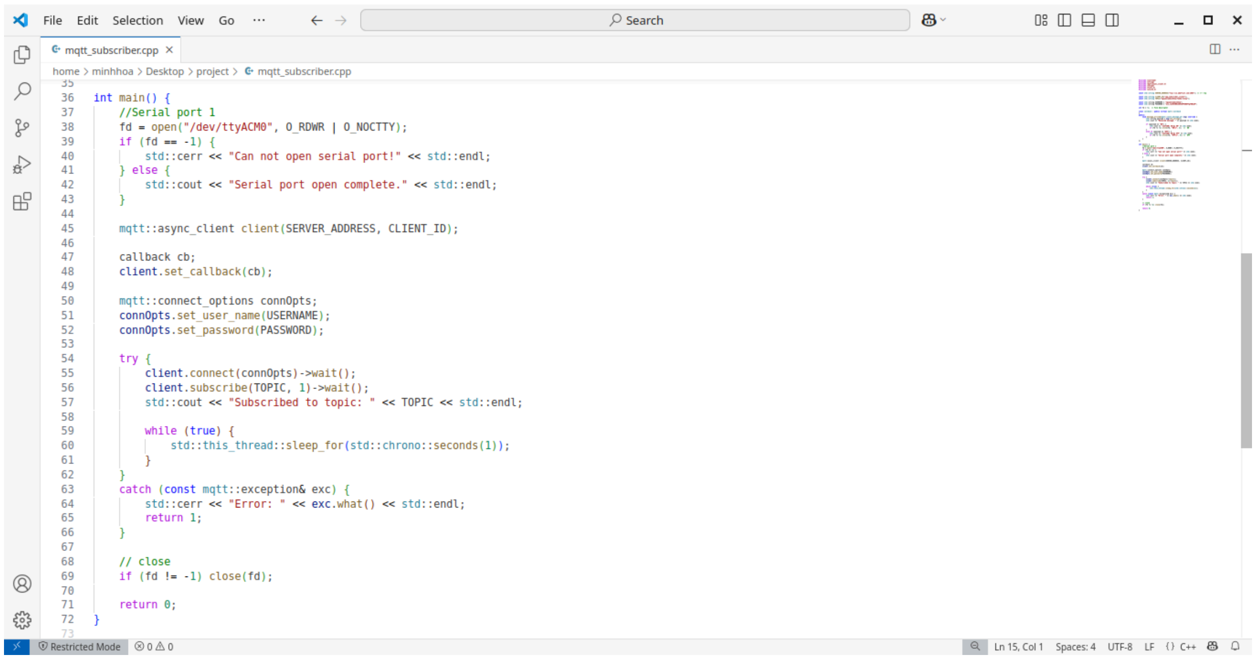
3.Write code to connect a device consisting of an Arduino Uno and a 5V relay to Adafruit via MQTT.

To be able to communicate between IoT devices and Adafruit IO, we need to use code to define and install libraries.


File Edit Selection View Go ···
Search
mqtt_subscriber.cpp
home > minhhoa > Desktop > project > mqtt_subscriber.cpp
Restricted Mode
0 0
Ln 1, Col 1 Spaces: 4 UTF-8 LF C++

```
#include <iostream>
#include <string>
#include <mqtt/async_client.h>
#include <thread>
#include <fcntl.h>
#include <unistd.h>

const std::string SERVER_ADDRESS("tcp://io.adafruit.com:1883"); // <-- tcp

const std::string CLIENT_ID("cpp_subscriber_client");
const std::string TOPIC("nguyenlamminhhoa/feeds/relay");

const std::string USERNAME = "nguyenlamminhhoa";
const std::string PASSWORD = "aio_vaCK34WK1XO54GPFENDFYgTH0m1M";

int fd = -1;  // File descriptor

class callback : public virtual mqtt::callback
{
public:
    void message_arrived(mqtt::const_message_ptr msg) override {
        std::string payload = msg->to_string();
        std::cout << "Received message: " << payload << std::endl;

        if (payload == "ON") {
            std::cout << "[ACTION] Relay ON" << std::endl;
            if (fd != -1) write(fd, "ON\n", 3);  //  ON
        }
        else if (payload == "OFF") {
            std::cout << "[ACTION] Relay OFF" << std::endl;
            if (fd != -1) write(fd, "OFF\n", 4); //  OFF
        }
    }
};

int main() {
    //Serial port 1
    fd = open("/dev/ttyACM0", O_RDWR | O_NOCTTY);
    if (fd == -1) {
```

First, we declare the libraries needed for the connection process. Then, we add the "username" and "API key" to officially connect to the correct account we are working with.

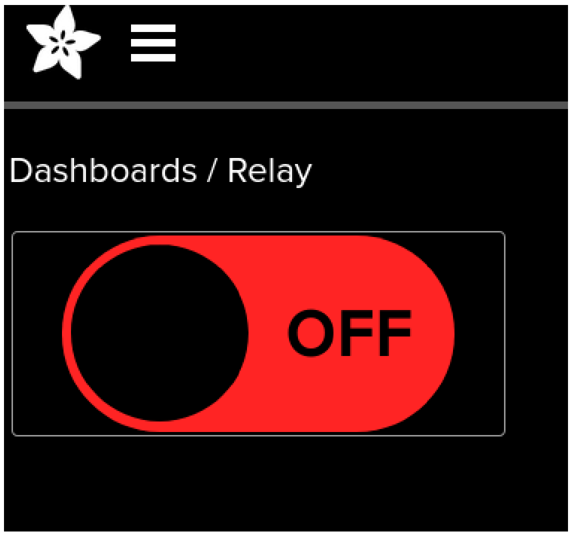

File Edit Selection View Go ···
Search
mqtt_subscriber.cpp
home > minhhoa > Desktop > project > mqtt_subscriber.cpp
Restricted Mode ⊗ 0 ⚠ 0
Ln 15, Col 1 Spaces: 4 UTF-8 LF {} C++

```cpp
int main() {
    //Serial port 1
    fd = open("/dev/ttyACM0", O_RDWR | O_NOCTTY);
    if (fd == -1) {
        std::cerr << "Can not open serial port!" << std::endl;
    } else {
        std::cout << "Serial port open complete." << std::endl;
    }

    mqtt::async_client client(SERVER_ADDRESS, CLIENT_ID);

    callback cb;
    client.set_callback(cb);

    mqtt::connect_options connOpts;
    connOpts.set_user_name(USERNAME);
    connOpts.set_password(PASSWORD);

    try {
        client.connect(connOpts)->wait();
        client.subscribe(TOPIC, 1)->wait();
        std::cout << "Subscribed to topic: " << TOPIC << std::endl;

        while (true) {
            std::this_thread::sleep_for(std::chrono::seconds(1));
        }
    }
    catch (const mqtt::exception& exc) {
        std::cerr << "Error: " << exc.what() << std::endl;
        return 1;
    }

    // close
    if (fd != -1) close(fd);

    return 0;
}
```

The rest of the code is where the connections are defined. In the main() function, the USB connection from the Arduino Uno to the PC is established. Then, the connection from the PC via MQTT to the Adafruit IO is defined, and the status and connection process is shown here.

4. Set up an on/off control for Adafruit.

To be able to control Relay via MQTT, a button is required on the control panel.

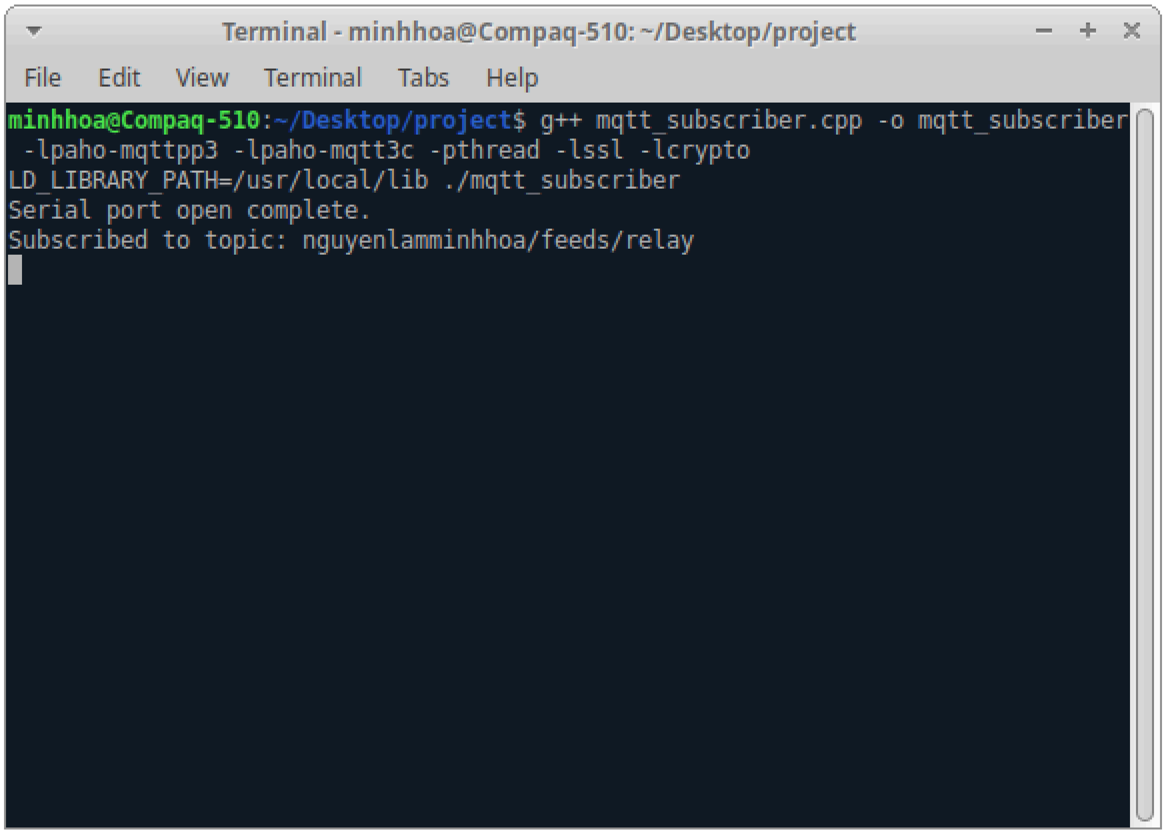


The control panel is simply designed with a lever showing ON and OFF status. This ON/OFF button is connected to the "Relay" Feed and from the "Relay" Feed via MQTT and code to reach the Arduino Uno and control the 5V Relay.

5. Launch and results

To be able to launch, we need to start Terminal and use the following line of code:

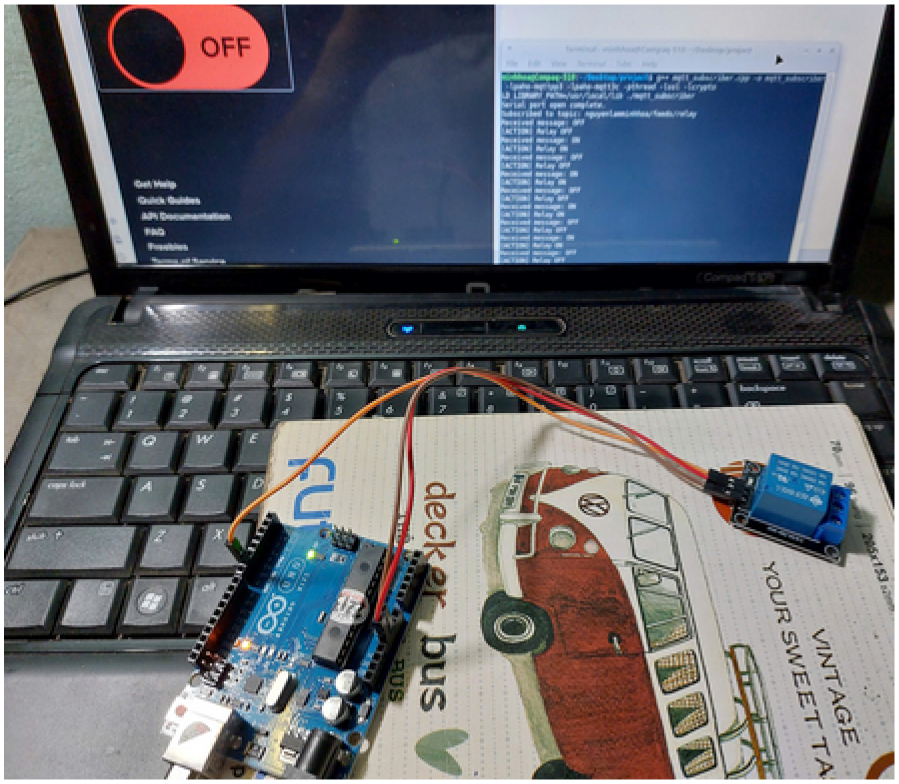
```
g++ mqtt_subscriber.cpp -o mqtt_subscriber -lpaho-mqttpp3 -lpaho-mqtt3c -pthread -lssl
-lcrypto
LD_LIBRARY_PATH=/usr/local/lib ./mqtt_subscriber
```



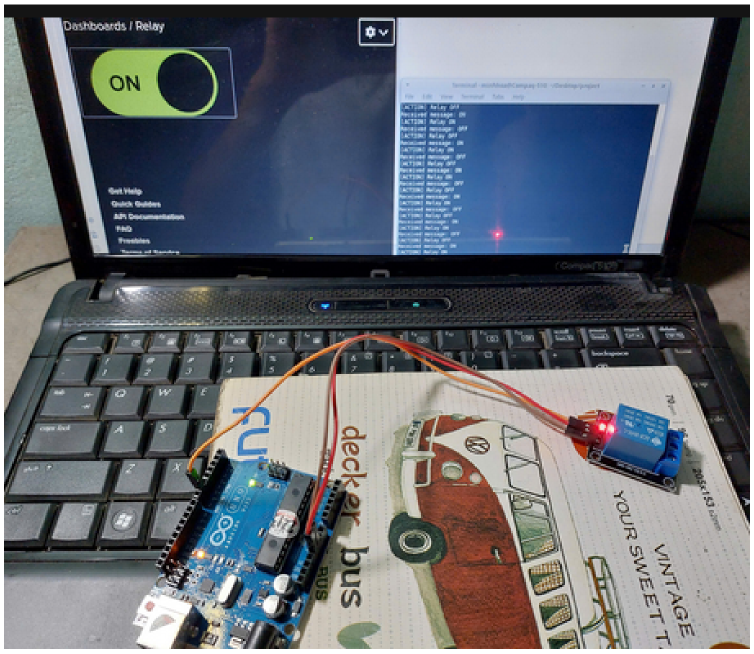
```
minhhoa@Compaq-510:~/Desktop/project$ g++ mqtt_subscriber.cpp -o mqtt_subscriber
 -lpaho-mqttpp3 -lpaho-mqtt3c -pthread -lssl -lcrypto
LD_LIBRARY_PATH=/usr/local/lib ./mqtt_subscriber
Serial port open complete.
Subscribed to topic: nguyenlamminhhoa/feeds/relay
```

When Terminal reports a successful connection, the information about the Adafruit Feed we created will appear. At this point, the connection has actually been established.

Then we can test by pressing the ON/OFF button on the Dashboard Relay.



When the button is in OFF mode, the Terminal will display the message "Relay Off" and we can see that the 5V Relay is not lit, which means it is not working while the Arduino is still lit.



When the button is in ON mode, the Terminal reports "Relay On" and the 5V Relay also shows a red light, which proves that the 5V Relay is active and receiving commands from Adafruit via MQTT.

6. Conclude

The 5V Relay has officially worked when it received commands from Adafruit IO via code and MQTT connection. This proves that the system works correctly and demonstrates the two functions required by the problem: Turning On/Off Relay via the Internet.