CME 466 Lab 1: IoT Protocols and MQTT

Nandish Jha (naj474)

Python Code

All code is included in the ZIP file submitted with this report. You will find major files in this ZIP archive. The personal laptop or PC that takes sensor data from the Pi or sends commands to the Pi has its own **subscribe_3_8.py** and **publish_3_9.py** files for the respective parts.

1. **a_temp_sensor.py**
   This file contains code for initializing the ADC module and the analog temperature sensor. The loop() function in this file/class will keep checking the temperature with frequency of 1 second.

2. **button.py**
   This file contains code for initializing a simple pushbutton. The on() function in this file/class will constantly poll the button to see whether to power on the system or not.

3. **main.py**
   This file contains code for initializing and creating object instances of all the other classes in other files. The **main.py** combines all the modules into one to make it work together, coherently.

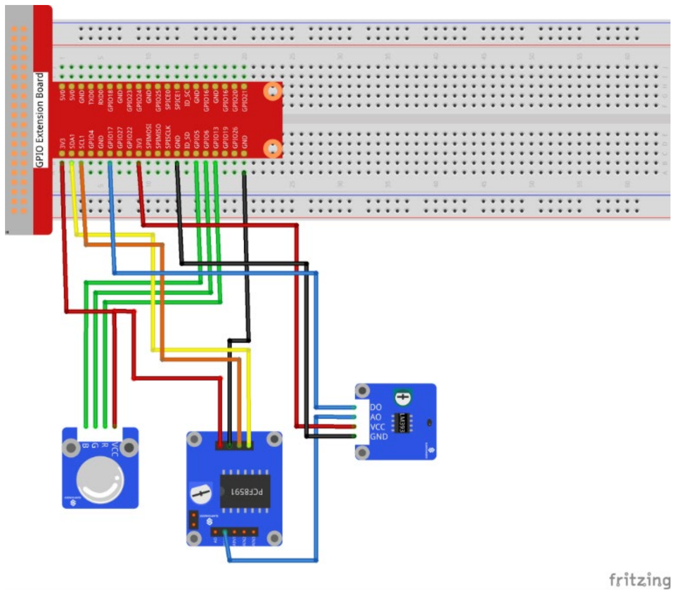
4. **rgb_led_module.py**
   This file contains code for initializing the big RGB module. There are 2 major functions in this file/class, setColor() and destroy(). The setColor() takes in one parameter which sets the color of the RGB led. The destroy() turns off and stops sending any and all signals to the RGB led.

5. **subscribe_1.py (*basic*)**
   This Python script subscribes to an MQTT broker, receives encrypted messages, and then decrypts and prints these messages.
   subscribe_2.py (*with cryptography*)
   This Python script subscribes to an MQTT broker receives encrypted JSON payloads, decrypts the 'message' field from the payload using a Fernet key, and prints the decrypted message. It also handles connection and message events for the MQTT client.
   subscribe_3.py (*for part 3: point number 9*)
   This Python script works the same as the above one but with additional functionality to send the sensor data from the Pi to the Laptop or a PC.

6. **publish_1.py (*basic*)**
   Python script that connects to an MQTT broker, encrypts messages using a Fernet key, and publishes them to a specific topic. It also includes a function to handle publish events and print the message ID and latency time.
   publish_2.py (*with cryptography*)
   Python script that connects to an MQTT broker, generates a random encryption key, encrypts messages with it, and publishes them to a specific topic. It also includes a function to handle publish events, print the message ID, and calculate latency time.
   publish_3.py (*for part 3: point number 8*)
   This Python script works the same as the above one but with additional functionality to do a command sent from the Laptop or a PC to the Pi.

Next page continuation

Circuitry



List of Python Packages

1. PCF8591.py for the Analog to Digital Converter (ADC) module.
2. RPi.GPIO for interacting with the GPIO pins in various modes.
3. The time library for delays and sleeps times.
4. The math library for doing calculations in the temperature files.
5. The datetime library
6. The cryptography library for encryption and decryption of payloads.
7. The paho-mqtt library for using the MQTT protocol service.

Some Questions

Part 1: Basic MQTT Program

(a) The result can undergo a change in data type through explicit typecasting, provided a valid type is transmitted via MQTT. For instance, a numeric message can be subjected to the int() function for this purpose.

(b) Certainly. After undergoing typecasting and being assigned to a variable, it becomes susceptible to manipulation, just like any locally assigned value.

(c) As MQTT transmits data in the form of bytes, preserving intricate data structures such as arrays becomes challenging. Nevertheless, we can reconstruct the structure at the receiving end by incorporating delimiters in the byte stream.

(d) Setting the Quality of Service (QoS) is performed within the publish() command.

- (rc, mid) = client.publish('messages/topic', encrypted_message, **qos=1**)

(e) Configuring a retain message is also accomplished within a publish() call.

Part 2: Performance analysis of the MQTT protocol

(a) Latency before adding the encryption.
broker.mqttdashboard.com = 3757 ms
mqtt.eclipseprojects.io = 3654 ms
test.mosquitto.org = 3743 ms
broker.emqx.io = 3642 ms

(b) Latency after adding the encryption.
broker.mqttdashboard.com = 3616 ms
mqtt.eclipseprojects.io = 3485 ms
test.mosquitto.org = 3509 ms
broker.emqx.io = 3438 ms