CME466 Deliverable 1

Jesse Aguirre
Jaa369, 11273119
January 17, 2024
Professor Wahid

Part 1 + 2 - Basic MQTT program and Performace Analysis of the MQTT protocol

Code:

Publisher Part 1 (No Encryption):

```
import paho.mqtt.client as mqtt
import time
import json
import logging

# ---------------------------------
# broker = "broker.hivemq.com"
# ---------------------------------
# broker = "broker.emqx.io"
# ---------------------------------
# broker = "test.mosquitto.org"
# ---------------------------------
broker = "mqtt.eclipseprojects.io"
# ---------------------------------

FIRST_RECONNECT_DELAY = 1
RECONNECT_RATE = 2
MAX_RECONNECT_COUNT = 12
```

```python
MAX_RECONNECT_DELAY = 60

def publish(client):
    global PACKETS_SENT
    global FLAG_EXIT
    PACKETS_SENT = 0
    FLAG_EXIT = False
    while not FLAG_EXIT:
        if PACKETS_SENT >= 30:
            FLAG_EXIT = True
            client.loop_stop()
        print(f"Published {PACKETS_SENT} packets to topic `jaa369_publish`")
        # Encode message as JSON which includes the message with a number and
        # a nicely formatted timestamp
        message = json.dumps({"message": f"Hello, World! {PACKETS_SENT}",
                                "timestamp": time.strftime("%Y-%m-%d
%H:%M:%S")})
        # Publish the message to the topic jaa369_publish

        # PART 1 Q4a: Show in your code how to change QoS
        # PART 1 Q4b: Show in your code how to change retain
        result = client.publish("jaa369_publish", message, qos = 0, retain =
False)
        status = result[0]

        if status == 0:
            print(f"Send `{message}` to topic `jaa369_publish`")
            PACKETS_SENT += 1
        else:
            print(f"Failed to send message to topic `jaa369_publish`")
        time.sleep(1)


def run():
    global PACKETS_SENT
    client = mqtt.Client("jaa369_publish")
    client.connect(broker)
    client.loop_start()
    time.sleep(1)
    if client.is_connected():
        print(f"Connected to {broker}")
        publish(client)
    else:
        print(f"Connection to {broker} failed")
        client.loop_stop()
        exit()



if __name__ == "__main__":
    run()
```

Receiver:

```
from datetime import datetime
import paho.mqtt.client as mqtt
import time
import json
import logging

# --------------------------------
# broker = "broker.hivemq.com"
# --------------------------------
# broker = "broker.emqx.io"
# --------------------------------
# broker = "test.mosquitto.org"
# --------------------------------
broker = "mqtt.eclipseprojects.io"
# --------------------------------


"""
Python to receive MQTT messages by subscribing to a topic named
```

```
jaa369_publish and parse the JSON message to print the message and
the timestamp as well as the time it took to receive the message
"""
FIRST_RECONNECT_DELAY = 1
RECONNECT_RATE = 2
MAX_RECONNECT_COUNT = 12
MAX_RECONNECT_DELAY = 60


# The callback for when the client receives a CONNACK response from the server.
def on_connect(client, userdata, flags, rc):
    if str(rc) == "0" and client.is_connected():
        print(f"Connected to {broker} with result code {str(rc)}")
         client.subscribe("jaa369_publish", qos = 0, options = None, properties =
None)
    else:
        print(f"Connection to {broker} failed with result code {str(rc)}")
        exit()

# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    # Decode the message payload from JSON
    payload = json.loads(msg.payload)
    # Print the message and the timestamp in
    # Parse JSON to print to console and write to file
    message = payload["message"]
    timestamp = payload["timestamp"]
    # Calculate the time it took to receive the message
    time_received = datetime.now()
    time_diff = time_received - datetime.strptime(timestamp, "%Y-%m-%d %H:%M:%S")
             print(f"Received `{message}` at {timestamp}. (Time taken:
{time_diff.total_seconds()} seconds)")
    # Write the message and the timestamp to a file
    with open(f"jaa369-d2p1_recv-{broker}.txt", "a") as file:
               file.write(f"Received `{message}` at {timestamp}. (Time taken:
{time_diff.total_seconds()} seconds)\n")

    if message.endswith("29") or message.endswith("30"):
        client.disconnect()
        client.loop_stop()
        exit()


def run():
    client = mqtt.Client("jaa369_subscribe")
    client.on_connect = on_connect
    client.on_message = on_message

    client.connect(broker, keepalive=60)
    client.loop_forever()


if __name__ == "__main__":
    run()
```

Part 1 Questions:

3a) Data type can be encoded inside a JSON file and it will be mainly flexible in regards to different data types. Type conversion could be performed using basic python functions such as str(), int() float(), etc. to get a desired data type.
3b) onMessage function within MQTT loads the JSON payload. By referencing similar to a dictionary we can acquire the data value and perform some arithmetic and logical operation.

3c) Similar to question 3b. We can acquire a set of data and print its contents as we would normally in Python

Question 4)

```
        # PART 1 Q4a: Show in your code how to change QoS
        # PART 1 Q4b: Show in your code how to change retain
        # PART 2 Q6: Encrypt the message using AES-256

        # Generate a key for AES-256
        key = Fernet.generate_key()
        # Store the key in a file
        with open("jaa369-d2p2-key.key", "wb") as file:
            file.write(key)
            file.close()

        # Create a Fernet instance with the key
        f = Fernet(key)
        # Encrypt the message
        encrypted_message = f.encrypt(message.encode())
        # Publish the encrypted message
          result = client.publish("jaa369_publish", encrypted_message, qos = 0,
retain = False)
        status = result[0]
```

Part 2 - Performance Analysis

Code:

```
from datetime import datetime
import paho.mqtt.client as mqtt
import json
from cryptography.fernet import Fernet

# ---------------------------------
# broker = "broker.hivemq.com"
# ---------------------------------
# broker = "broker.emqx.io"
# ---------------------------------
# broker = "test.mosquitto.org"
# ---------------------------------
broker = "mqtt.eclipseprojects.io"
# ---------------------------------


"""
Python to receive MQTT messages by subscribing to a topic named
jaa369_publish and parse the JSON message to print the message and
the timestamp as well as the time it took to receive the message
"""
```

```python
# The callback for when the client receives a CONNACK response from the server.
def on_connect(client, userdata, flags, rc):
    if str(rc) == "0" and client.is_connected():
        print(f"Connected to {broker} with result code {str(rc)}")
         client.subscribe("jaa369_publish", qos = 0, options = None, properties =
None)
    else:
        print(f"Connection to {broker} failed with result code {str(rc)}")
        exit()

# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    # Decode the message payload from JSON
    payload = msg.payload.decode()
    ## PART 2 Q6: Decrypt the message
    # Read the key from the file
    with open("jaa369-d2p2-key.key", "rb") as file:
        key = file.read()
        file.close()
    # Create a Fernet instance with the key
    f = Fernet(key)
    # Decrypt the payload
    decrypted_payload = f.decrypt(payload.encode())
    # Decode the decrypted message
    payload = json.loads(decrypted_payload.decode())
    # Print the message and the timestamp in
    # Parse JSON to print to console and write to file
    message = payload["message"]
    timestamp = payload["timestamp"]
    # Calculate the time it took to receive the message
    time_received = datetime.now()
    time_diff = time_received - datetime.strptime(timestamp, "%Y-%m-%d %H:%M:%S")
             print(f"Received   `{message}`   at   {timestamp}.   (Time   taken:
{time_diff.total_seconds()} seconds)")
    # Write the message and the timestamp to a file
    with open(f"jaa369-d2p2_decrypted-{broker}.txt", "a") as file:
               file.write(f"Received  `{message}`  at  {timestamp}.  (Time  taken:
{time_diff.total_seconds()} seconds)\n")
```

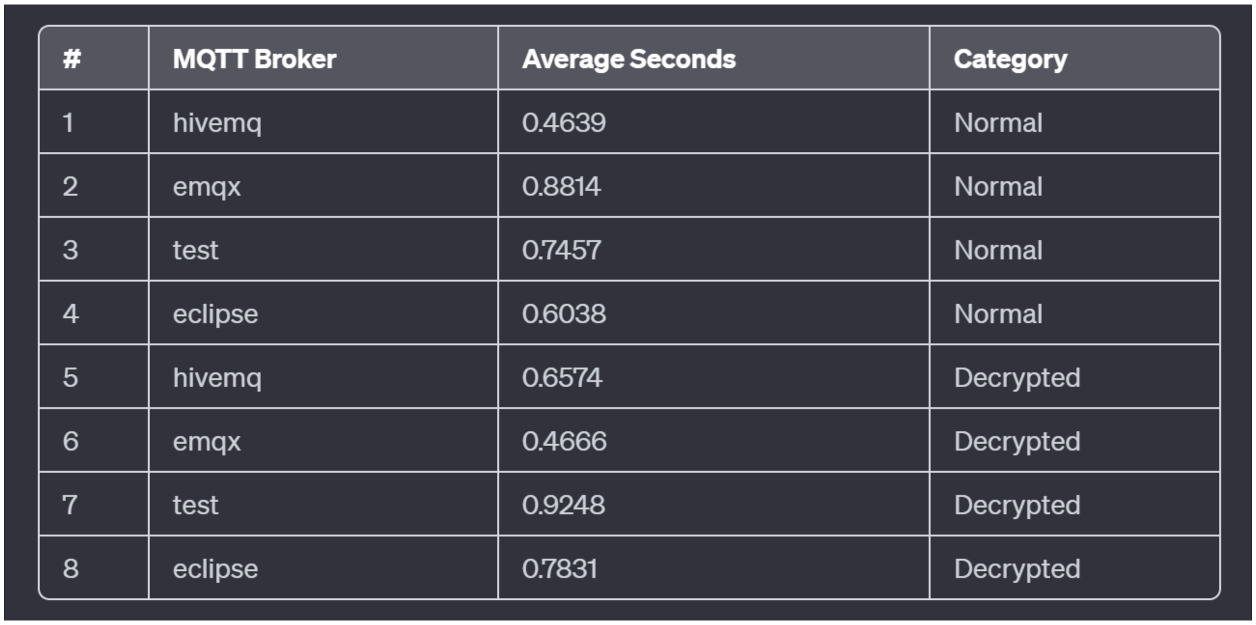
```
    if message.endswith("29") or message.endswith("30"):
        client.disconnect()
        client.loop_stop()
        exit()


def run():
    client = mqtt.Client("jaa369_subscribe")
    client.on_connect = on_connect
    client.on_message = on_message

    client.connect(broker, keepalive=60)
    client.loop_forever()


if __name__ == "__main__":
    run()
```

Output Example for One broker:

```
Received `Hello, World! 0` at 2024-01-25 20:34:32. (Time taken: 0.852887 seconds)
Received `Hello, World! 1` at 2024-01-25 20:34:33. (Time taken: 0.855148 seconds)
Received `Hello, World! 2` at 2024-01-25 20:34:34. (Time taken: 0.857108 seconds)
Received `Hello, World! 3` at 2024-01-25 20:34:35. (Time taken: 0.859179 seconds)
Received `Hello, World! 4` at 2024-01-25 20:34:36. (Time taken: 0.861244 seconds)
Received `Hello, World! 5` at 2024-01-25 20:34:37. (Time taken: 0.862918 seconds)
Received `Hello, World! 6` at 2024-01-25 20:34:38. (Time taken: 0.865066 seconds)
Received `Hello, World! 7` at 2024-01-25 20:34:39. (Time taken: 0.867011 seconds)
Received `Hello, World! 8` at 2024-01-25 20:34:40. (Time taken: 0.869107 seconds)
Received `Hello, World! 9` at 2024-01-25 20:34:41. (Time taken: 0.871189 seconds)
```

Etc.

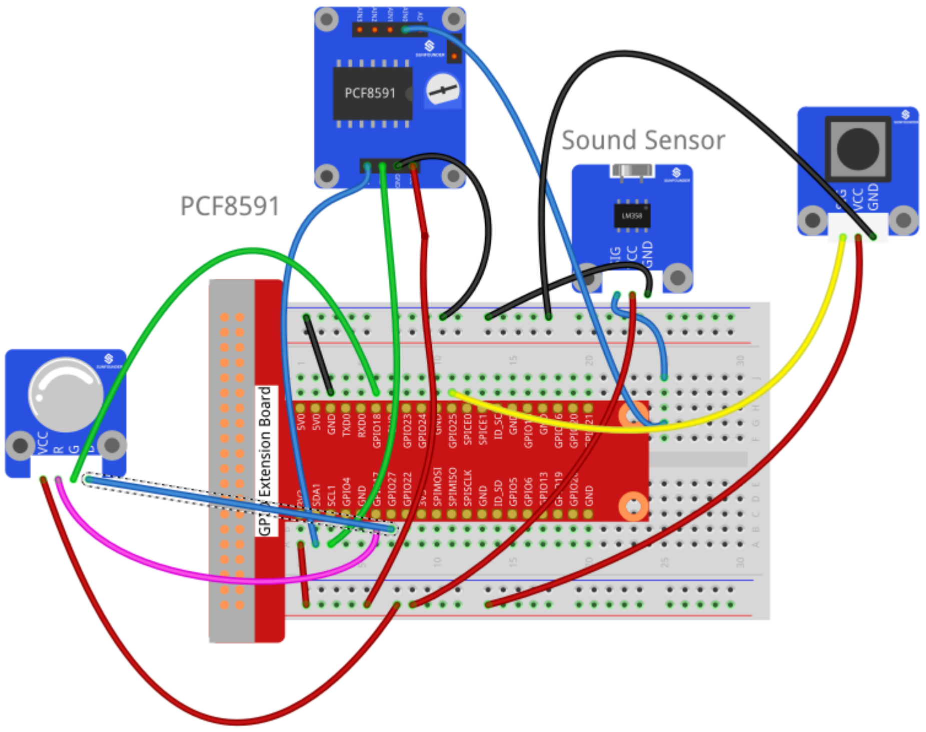
Table 1: Latency Results for Normal MQTT messages and Encrypted



#	MQTT Broker	Average Seconds	Category
1	hivemq	0.4639	Normal
2	emqx	0.8814	Normal
3	test	0.7457	Normal
4	eclipse	0.6038	Normal
5	hivemq	0.6574	Decrypted
6	emqx	0.4666	Decrypted
7	test	0.9248	Decrypted
8	eclipse	0.7831	Decrypted

Part 3 - MQTT and Edge Node

Schematic



Code

Part 8 Publisher:

```python
# Jesse Aguirre
# CME466: Deliverable 2 Part 3
# 2024-01-25

# Constants for RPI
import RPi.GPIO as GPIO
import time
import PCF8591 as ADC
import sys

# Broker
import paho.mqtt.client as mqtt
import json
from cryptography.fernet import Fernet

# NODE VARIABLES
PINS = (17, 18, 27, 25)  # Blue, Green, Red, Switch
RGB_B, RGB_G, RGB_R, SWITCH = PINS
ADC_ADDRESS = 0x48
VOICE_THRESHOLD = 80
VOICE_MIDDLE = 120
VOLUME_STATUS = ""

# MQTT functions
def connect_mqtt():
    client = mqtt.Client("jaa369_d2publish")
    client.on_connect = on_connect
    client.connect(broker_name, keepalive=60)
    return client

def publish(client):
    global payload
    client.publish("jaa369_sensorData", payload, qos = 0, retain = False)
    print(f"Published `{payload}` to topic `jaa369_sensorData`")
    time.sleep(0.2)
```

```python
def on_connect(client, userdata, flags, rc):
    if str(rc) == "0" and client.is_connected():
        print(f"Connected to {broker_name} with result code {str(rc)}")
    else:
        print(f"Connection to {broker_name} failed with result code {str(rc)}")



def setup():
    GPIO.setmode(GPIO.BCM)
    for pin in PINS[:-1]:
        print(f"Configuring pin: {pin}")
        GPIO.setup(pin, GPIO.OUT)

    GPIO.setup(SWITCH, GPIO.IN, pull_up_down=GPIO.PUD_UP)
    GPIO.add_event_detect(SWITCH, GPIO.FALLING, callback=button_pressed)

    print("Configuring ADC")
    ADC.setup(ADC_ADDRESS)

def button_pressed(channel):
    global voice_enabled
    # Invert the current state of the voice_enabled variable
    voice_enabled = not voice_enabled

def update_leds(voice_val):
    global VOLUME_STATUS
    if voice_val < VOICE_THRESHOLD:
        VOLUME_STATUS = "Too Loud!"
        # Turn on Yellow LED
        GPIO.output(RGB_R, GPIO.LOW)
        GPIO.output(RGB_G, GPIO.LOW)
        GPIO.output(RGB_B, GPIO.HIGH)
    elif VOICE_THRESHOLD <= voice_val < VOICE_MIDDLE:
        VOLUME_STATUS = "Just Right!"
        # Turn Purple
        GPIO.output(RGB_R, GPIO.LOW)
        GPIO.output(RGB_G, GPIO.HIGH)
        GPIO.output(RGB_B, GPIO.LOW)

    else:
        VOLUME_STATUS = "Dead Quiet!"
        # Turn Blue
        GPIO.output(RGB_R, GPIO.HIGH)
        GPIO.output(RGB_G, GPIO.LOW)
        GPIO.output(RGB_B, GPIO.LOW)



def loop():
    global voice_enabled
    global payload
    voice_enabled = False
    while True:
        if voice_enabled:
            voice_val = ADC.read(0)
            print(f"Voice val is: {voice_val}")
            # Package the value of the sensor as a JSON object along with
            # volume status and timestamp

            payload = json.dumps({"voice_val": voice_val,
                                  "volume_status": VOLUME_STATUS,
                                           "timestamp": time.strftime("%Y-%m-%d
%H:%M:%S")})

            update_leds(voice_val)
            publish(client)

def destroy():
    for pin in PINS[:-1]:
        GPIO.output(pin, GPIO.LOW)
    GPIO.cleanup()

if __name__ == '__main__':
    setup()
    global broker_name
    broker_name = sys.argv[1]
    global client
    # RPI
    global voice_enabled
    global payload
    voice_enabled = False
    try:
        client = connect_mqtt()
        client.loop_start()
        time.sleep(1)
        if client.is_connected():
            print(f"Connected to {broker_name}")
            loop()
        else:
            print(f"Connection to {broker_name} failed")
            client.loop_stop()
            exit()
    except KeyboardInterrupt:
        destroy()
```

Part 8 (Receiver):

```
from datetime import datetime
import paho.mqtt.client as mqtt
import time
import json
import logging
import sys

# ---------------------------------
# broker = "broker.hivemq.com"
# ---------------------------------
# broker = "broker.emqx.io"
# ---------------------------------
# broker = "test.mosquitto.org"
# ---------------------------------
# broker = "mqtt.eclipseprojects.io"
# ---------------------------------
```

```python
"""
Python to receive MQTT messages by subscribing to a topic named
jaa369_publish and parse the JSON message to print the message and
the timestamp as well as the time it took to receive the message
"""
FIRST_RECONNECT_DELAY = 1
RECONNECT_RATE = 2
MAX_RECONNECT_COUNT = 12
MAX_RECONNECT_DELAY = 60


# The callback for when the client receives a CONNACK response from the server.
def on_connect(client, userdata, flags, rc):
    if str(rc) == "0" and client.is_connected():
        print(f"Connected to {broker} with result code {str(rc)}")
        client.subscribe("jaa369_sensorData", qos = 0, options = None, properties
= None)
    else:
        print(f"Connection to {broker} failed with result code {str(rc)}")
        exit()

# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    # Decode the voice value, volume_status, timestamp from se payload from JSON
    payload = json.loads(msg.payload)
    # Print the message and the timestamp in
    # Parse JSON to print to console and write to file
    voice_val = payload["voice_val"]
    volume_status = payload["volume_status"]
    timestamp = payload["timestamp"]
    # Calculate the time it took to receive the message
    time_received = datetime.now()
    time_diff = time_received - datetime.strptime(timestamp, "%Y-%m-%d %H:%M:%S")
     print(f"Received `{voice_val}: {volume_status}` at {timestamp}. (Time taken:
{time_diff.total_seconds()} seconds)")
    # Write the message and the timestamp to a file
    with open(f"jaa369-d2p3_recv-{broker}.txt", "a") as file:
            file.write(f"Received `{voice_val}: {volume_status}` at {timestamp}.
(Time taken: {time_diff.total_seconds()} seconds)\n")

    if time_diff.total_seconds() > 30:
        client.disconnect()
        client.loop_stop()
        exit()



def run():
    client = mqtt.Client("jaa369_subscribe")
    client.on_connect = on_connect
    client.on_message = on_message
    global broker
    broker = sys.argv[1]

    client.connect(broker, keepalive=60)
    client.loop_forever()


if __name__ == "__main__":
    run()
```

Part 9 Publisher

```
import paho.mqtt.client as mqtt
import time

broker = "broker.hivemq.com"
client = mqtt.Client("jaa369_publish")
client.connect(broker)

run = True
listOfCommands = ["ledOn", "ledRed", "ledGreen", "ledBlue", "readSen"]

while(run):
    print("<----------------------------------------->")
    print("Choose the following commands to execute:")
    print("   1.) ledOn")
    print("   2.) ledRed")
    print("   3.) ledGreen")
    print("   4.) ledBlue")
    print("   5.) readSen")
    print("   6.) quit")
    print()

    msg = input("Enter the command you want to execute: ")

    print()
    if (msg == "quit"):
        run = False
    elif (msg in listOfCommands):
        client.publish("jaa369_read", msg)
        print("Executing Command: ", msg)
    else:
        print("Command is not unknown...")
    print()
    time.sleep(3)
```

Part 9 Receiver:

```
import RPi.GPIO as GPIO
import PCF8591 as ADC
import time
import math
import paho.mqtt.client as mqtt

broker = "broker.hivemq.com"
client = mqtt.Client("jaa369_raspi")
client.connect(broker)

BtnPin = 25
Rpin = 27
Gpin = 18
Bpin = 17
```

```python
def setUp():
    GPIO.setmode(GPIO.BCM)
    ADC.setup(0x48)
    GPIO.setwarnings(False)

    # Switch Trigger
    GPIO.setup(BtnPin, GPIO.IN)

    # LED Trigger
    GPIO.setup(Rpin, GPIO.OUT)
    GPIO.setup(Gpin, GPIO.OUT)
    GPIO.setup(Bpin, GPIO.OUT)

def Led(r, g, b):
    GPIO.output(Rpin, r)
    GPIO.output(Gpin, g)
    GPIO.output(Bpin, b)


def readSensor():
    AnalogVal = ADC.read(0)
    return AnalogVal

def destroy():
    GPIO.output(Gpin, GPIO.HIGH)
    GPIO.output(Rpin, GPIO.HIGH)
    GPIO.cleanup()

def on_message(client, userdata, message):
    msg = message.payload.decode("utf-8")
    r = False
    g = False
    b = False
    if (msg[0] == "0"):
        r = True
    if (msg[1] == "0"):
        g = True
    if (msg[2] == "0"):
        b = True
    Led(r,g,b)
    print("r g b")

def on_message(client, userdata, message):
    msg = message.payload.decode("utf-8")
    if (msg == "ledOn"):
        Led(False, False, False)
        print("LED turning on...")
    elif (msg == "ledRed"):
        Led(False, True, True)
        print("LED turning red on...")
    elif (msg == "ledGreen"):
        Led(True, False, True)
        print("LED turning green on...")
    elif (msg == "ledBlue"):
        Led(True, True, False)
        print("LED turning blue on...")
    elif (msg == "readSen"):
        print("Reading from the sensor..")
        temp = readSensor()
        print("Voice value is: ", temp)

# Program Start
if __name__ == '__main__':
    setUp()
    Led(False, False, False)
    try:
        client.loop_start()
        client.subscribe("jaa369_read")
        client.on_message = on_message
        time.sleep(100)
    except KeyboardInterrupt:
        destroy()
```

List of Packages

1. RPi.GPIO
2. Standard Python Libraries
3. PCF8591 for ADC Converter
4. Paho MQTT library

Notes:

- I have attached multiple files of outputs in the out folder. Please go there if you need any reference to my code. I have also published the results of some of my scripts that parse information from output given.