**Week 8 Lab 5 Report**

**Name:** Javon Teo Tze Kai

**Student ID:** A0233706J

**Lab Group:** B02

**Task 1:**

“b” indicates that our Hello! message string is in bytes literal format, it means that the string is represented as a sequence of bytes.

**Task 2:**

def on\_message(client, userdata, message):

    print("Received message " + message.payload.decode())

**Task 3:**

I modified the HelloToMyself.ino:

char \*subscribeTopic = "hello/world";

char \*publishTopic = "hello/esp";

By changing the subscribeTopic in the HelloToMyself.ino, such that it subscribes to the topic “hello/world”, the data would then be sent to the ESP32 from the pub.py python script.

**Task 4:**

ESP32 sketch:

char \*publishTempTopic = "weather/temp";

char \*publishHumidTopic = "weather/humidity";

struct measurements {

  float temperature;

  float humidity;

};

void loop()

{

    struct measurements mmts = dht11\_loop(); // collects data DHT11 sensor.

    mqttClient.publish(publishTempTopic, String(mmts.temperature), 0, false);

    mqttClient.publish(publishHumidTopic, String(mmts.humidity), 0, false);

    delay(2000);

}

mqtt.py python script:

def on\_connect(client, userdata, flags, rc):

    print("Connected with result code: " + str(rc))

    # client.subscribe("hello/#")

    client.subscribe("weather/temp")

    client.subscribe("weather/humidity")

def on\_message(client, userdata, message):

    if message.topic == "weather/temp":

        print(f"Received message: {message.payload.decode()} °C on topic: {message.topic}")

    else:

        print(f"Received message: {message.payload.decode()} % on topic: {message.topic}")

**Task 5:**

States:

* Deep Sleep mode has power consumption of 150 µA. The device would last ???.
* Modem-sleep mode has power consumption of 27mA to 44mA . The device would last for ???.
* Active mode has power consumption of 240mA. The device would last for ???.

**Task 6:**

#define uS\_TO\_S\_FACTOR 1000000ULL

#define TIME\_TO\_SLEEP  20

char \*subscribeTopic = "hello/world";

char \*publishTopic = "hello/esp32";

ESP32MQTTClient mqttClient;

void setup(){

  Serial.begin(115200);

  esp\_sleep\_enable\_timer\_wakeup(TIME\_TO\_SLEEP \* uS\_TO\_S\_FACTOR);

  Serial.println("Setup ESP32 to sleep for every " + String(TIME\_TO\_SLEEP) + " Seconds");

  log\_i();

  log\_i("setup, ESP.getSdkVersion(): ");

  log\_i("%s", ESP.getSdkVersion());

  mqttClient.enableDebuggingMessages();

  mqttClient.setURI(server);

  mqttClient.enableLastWillMessage("lwt", "I am going offline");

  mqttClient.setKeepAlive(30);

  WiFi.begin(ssid, pass);

  WiFi.setHostname("c3test");

  while(WiFi.status() != WL\_CONNECTED) {

    Serial.print('.');

    delay(1000);

  }

  Serial.println("CONNECTED");

  mqttClient.loopStart();

}

void loop(){

  delay(5000);

  // Sends MQTT message

  mqttClient.publish(publishTopic, "Hello from ESP32!!!!", 0, false);

  // Prints to Serial that it has sent the message

  Serial.println("SENT");

  // We sent ESP32 to sleep.

  Serial.println("Going to sleep now");

  delay(5000);

  esp\_deep\_sleep\_start(); // This makes ESP32 go to sleep.

}

The ESP32 is set to go to deep-sleep every 20 seconds. When it wakes up, it firstly sets the parameters of the MQTT client, then connects to the WiFi network with the provided SSID and password. It then goes to the loop(), where after 5 seconds delay, sends an MQTT message (“Hello from ESP32!!!!”), prints “SENT” to the Serial, then prints “Going to sleep now” to the Serial before setting the ESP32 to deep sleep.

**Task 7:**

<Write a function in our mqtt.py python script to classify the temperature and send back the classification result. (1 mark)>

**Task 8:**

<Write a program (sketch) that reads in the DHT sensors, connects to wifi, publishes the DNT sensor readings, waits for a classifier response, actuates appropriately, and then puts the ESP32 device back into deep-sleep for 20 seconds. Explain and include this program in your report. (4 marks)>