**Q3.**

**Arduino code to connect to HiveMQ**

#include <Arduino\_LSM6DS3.h>

#include <PubSubClient.h>

#include <SPI.h>

#include <WiFiNINA.h>

#include <ArduinoJson.h>

// WiFi Credentials

char ssid[] = "";

char pass[] = "";

int status = WL\_IDLE\_STATUS;

// Initialize WiFi client

WiFiSSLClient wifi\_client;

// Connect to Wifi Access Point

void connectToAP(){

while (status != WL\_CONNECTED){

Serial.print("Attempting to connect to SSID: ");

Serial.println(ssid);

// Connect to WPA/WP2 network

status = WiFi.begin(ssid, pass);

// wait 5 second for connection

delay(5000);

if (status == WL\_CONNECTED){

Serial.println("Connected to WiFi");

printWifiStatus();

}else{

Serial.println("Failed to connect. Retrying...");

}

Serial.println("Connected");

}

}

// Print results to serial monitor

void printWifiStatus(){

// Network SSID

Serial.print("SSID: ");

Serial.println(WiFi.SSID());

// Device IP address

IPAddress ip = WiFi.localIP();

Serial.print("IP Address: ");

Serial.println(ip);

}

// Gyroscope axis

float x, y, z;

/\*\*\*\*\*\*\* MQTT Broker Connection Details \*\*\*\*\*\*\*/

const char\* mqtt\_server = "943103f9e51648f9b2e42b0741f78511.s1.eu.hivemq.cloud";

const char\* mqtt\_username = "";

const char\* mqtt\_password = "";

const int mqtt\_port =8883;

/\*\*\*\* MQTT Client Initialisation Using WiFi Connection \*\*\*\*\*/

PubSubClient client(wifi\_client);

unsigned long lastMsg = 0;

#define MSG\_BUFFER\_SIZE (50)

char msg[MSG\_BUFFER\_SIZE];

/\*\*\*\*\*\*\*\*\*\*\*\*\* Connect to MQTT Broker \*\*\*\*\*\*\*\*\*\*\*/

void reconnect() {

// Loop until we're reconnected

while (!client.connected()) {

Serial.print("Attempting MQTT connection...");

String clientId = "LongID"; // Create a random client ID

//clientId += String(random(0xffff), HEX);

// Attempt to connect

if (client.connect(clientId.c\_str(), mqtt\_username, mqtt\_password)) {

Serial.println("connected");

//

} else {

Serial.print("failed, rc=");

Serial.print(client.state());

Serial.println(" try again in 5 seconds"); // Wait 5 seconds before retrying

delay(5000);

}

}

}

/\*\*\*\*\* Call back Method for Receiving MQTT messages \*\*\*\*/

void callback(char\* topic, byte\* payload, unsigned int length) {

String incommingMessage = "";

for (int i = 0; i < length; i++) incommingMessage+=(char)payload[i];

Serial.println("Message arrived ["+String(topic)+"]"+incommingMessage);

//--- check the incomming message

if( strcmp(topic,"led\_state") == 0){

//

}

}

/\*\*\*\* Method for Publishing MQTT Messages \*\*\*\*\*\*\*\*\*\*/

void publishMessage(const char\* topic, String payload , boolean retained){

if (client.publish(topic, payload.c\_str(), true)){

Serial.println("Message publised ["+String(topic)+"]: "+payload);

} else{

Serial.println("Message publish failed");

}

}

void setup() {

Serial.begin(9600); // set baud rate

while (!Serial); // wait for port to init

if (!IMU.begin()) {

while (1);

}

// Check for the WiFi module

if (WiFi.status() == WL\_NO\_MODULE){

Serial.println("WiFi module failed!");

while (true);

}

// wifi connection

connectToAP();

printWifiStatus();

//mqtt connection

client.setServer(mqtt\_server, mqtt\_port);

client.setCallback(callback);

}

void loop() {

// check if client is connected

if (!client.connected()) reconnect(); // check if client is connected

client.loop();

// read accelero data

if (IMU.accelerationAvailable()) {

IMU.readAcceleration(x, y, z);

}

Serial.println(String(x) + "," + String(y) + "," + String(z));

DynamicJsonDocument doc(1024);

doc["x"] = x;

doc["y"] = y;

doc["z"] = z;

char mqtt\_message[128];

serializeJson(doc, mqtt\_message);

publishMessage("data", mqtt\_message, true); // send topic and data

delay(5000); // delay 5s

}

This code works by connecting the Arduino Nano 33 to the internet and connecting to the MQTT broker to send our topic and message. The code for connecting to the internet I got it from DroneBot Workshop (2021) and for the HiveMQ connection it’s from Kudzai Manditereza (2023). There are comments for each functions, like how the connection and status to the internet, the connection, callback, publish function for the MQTT.

**Python code for Mongo DB connection**

# MongoDB connection

mongo\_username = quote\_plus("")

mongo\_password = quote\_plus("")

mongo\_uri = f"mongodb+srv://{mongo\_username}:{mongo\_password}@cluster0.950if.mongodb.net/" # MongoDB connection URI

mongo\_client = MongoClient(mongo\_uri)

db = mongo\_client["sensor\_data"]

collection = db["accelerometer"]

# MQTT brokers details

broker = "943103f9e51648f9b2e42b0741f78511.s1.eu.hivemq.cloud"

port = 8883

username = ""

password = ""

topic = "data" # topic to subscribe to

# Function to get the current time

def timestamp():

    return datetime.now().strftime('%Y%m%d%H%M%S')

# Callback when the client connects to the broker

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

    if rc == 0:

        print("Connected to broker")

        client.subscribe(topic)

    else:

        print(f"Failed to connect, return code {rc}")

# Callback when a message is received

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

    payload = msg.payload.decode()

    print(f"Message received: {msg.topic} {payload}")

    # Parse the data

    data = json.loads(payload)

    formatted\_data = {

        "Timestamp": timestamp(),

        "x": data["x"],

        "y": data["y"],

        "z": data["z"]

    }

    # Save to CSV file

    with open("data.csv", mode='a', newline='') as csv\_file:

        csv\_writer = csv.writer(csv\_file)

        csv\_writer.writerow(formatted\_data.values())

    # Save to JSON file

    df = pd.read\_csv("data.csv", header=None, names=['Timestamp', 'x', 'y', 'z']) # read csv file

    # json\_data = df.to\_json(orient='records') # convert dataframe to json dictionaries

    # with open('data.json', 'w') as file2:

    #     json.dump(json\_data, file2, indent=4)

    # Upload data to MongoDB

    collection.insert\_one(formatted\_data)

    print("Data inserted into MongoDB")

# Initialize client

client = mqtt.Client()

# Set callbacks

client.on\_connect = on\_connect

client.on\_message = on\_message

# Set username and passords

client.username\_pw\_set(username, password)

# Enable SSL/TLS

client.tls\_set()

# Connect to the broker

client.connect(broker, port)

# Start the loop to process received messages

client.loop\_forever()

The code works as follows. First, make a connection to MongoDB, the define the details for MQTT brokers, define some functions for callbacks. When the message is received, I parse the message, save the data in a csv for later analysis, and upload it the MongoDB. The code for Redis is quite similar to this.

**Q4.**

<https://www.youtube.com/watch?v=C8xQnIwlwL8>

**Q5.**

References:

CONTEC. (2022). *Serial communication Basic Knowledge -RS-232C/RS-422/RS-485- | CONTEC*. [online] Available at: <https://www.contec.com/support/basic-knowledge/daq-control/serial-communicatin/#:~:text=and%20synchronous%20communication->.

DroneBot Workshop (2021). *Arduino Nano 33 IoT - Getting Started*. [online] www.youtube.com. Available at: <https://www.youtube.com/watch?v=KjETPxv_VR4>.

Kudzai Manditereza (2023). *Getting Started with MQTT on Arduino Using NodeMCU ESP8266*. [online] Hivemq.com. Available at: <https://www.hivemq.com/blog/mqtt-on-arduino-nodemcu-esp8266-hivemq-cloud/#heading-setting-up-a-hive-mq-cloud-mqtt-broker>.

P, P.K. (2021). *The Modern industrial world is headed to IIoT digital platforms in these days, but the difficulties being faced by industries are the protocols that supports IoT hubs and clouds are limited to some light weight protocols. Most of the industries have still legacy serial communication equipment in the*. [online] Linkedin.com. Available at: <https://www.linkedin.com/pulse/serial-mqtt-protocol-conversion-prasanth-k-p/>.