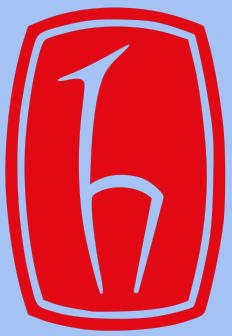
EMBEDDED SYSTEMS TERM PROJECT

Subject: 3D Modelling and Data Plotting using Web Sockets

Supervisor: Harun Artuner
Prepared by: Sertaç Güler
School ID: 21986764



Summary:

For the embedded systems course project I chose to send sensor data from Arduino Nano 33 Ble to Raspberry Pi using Bluetooth Low Energy. After fetching data raspberry pi should host a website and give 3d modelling of a cube and graphs of sensor data to users entering the website in real time. For this Arduino Nano 33 Ble has a built in 9 axis LSM9DS1 IMU sensor. It consists of a 3 axis accelerometer , a 3 axis gyroscope and a 3 axis magnetometer sensor. Accelerometer sensor measures the acceleration according to the axis. Its default values are -1 for the axis looking towards the ground because of the earth gravity acceleration and the others should be 0 when not moving. In addition the 3 axis gyroscope provides a measuring of angular momentum and can be used in applications that need a physical balance. Lastly the 3 axis magnetometer which uses magnetic fields and can be used to navigate devices.

BLE is a bluetooth low energy component to send and receive data to and from computers while preserving low energy. Its range is similar to classical Bluetooth.However because it is designed for low energy and restricts data packet size that can be sent at a time.

To connect to Arduino Nano 33 Ble Raspberry pi should know its mac address and to communicate with it a unique identifier for each data channel. After using a scan on bluetoothctl in raspberry pi os I could manage to find the mac address of Arduino Nano 33 Ble. Arduino Nano 33 Ble code defines unique identifiers for each sensor data channel and raspberry pi copies these identifiers into python code which uses flask to send data to websocket. Websockets are better in cases when you need to have a real time to represent in web page. In contrast to normal http it has a lower overhead and does not establish connection every time it needs to send data.

Raspberry Pi should also have a Html and Javascript code to handle the frontend of the website. These are combined into one file as index.html which has a script tag to include javascript codes to handle charts and data flowing into the website.

In addition to these because my raspberry pi is connected to the internet via Eduroam, which I am not an admin in, I needed to set up a tunneling to serve the raspberry pi to the whole internet. For this reason I set ngrok which is used for tunneling in these situations.

Arduino Side:

In this part of arduino code I imported necessary libraries for Arduino BLE communication and sensor library of IMU which is LSM9DS1.After I needed to define Ble service with unique identifier which is 128 bit long. These are coded in hexadecimal and we have 32 hexadecimal digits. The Ble string characteristics are defined to notify the connected device when its value changes. I have 3 of them each for accelerometer , gyroscope and magnetometer data. Because I couldnt have more than 20 byte long for my BLE device I needed to define them separately.

```
#include <ArduinoBLE.h>
#include <Arduino_LSM9DS1.h>

BLEService customService("12345678-1234-1234-1234-123456789ABC");
BLEStringCharacteristic accCharacteristic("87654321-4321-4321-4321-210987654321", BLENotify, 20);
BLEStringCharacteristic gyroCharacteristic("87654321-4321-4321-4321-210987654322", BLENotify, 20);
BLEStringCharacteristic magCharacteristic("87654321-4321-4321-4321-210987654323", BLENotify, 20);
```

In the setup code of Arduino I set the name to find the mac address of Arduino Ble device when I run scan on raspberry pi. I add an advertised service to show the Ble device in other devices. And this service is added each 3 string characteristics.And I finalize with adding the configured service.

```
    BLE.setLocalName("Nano33BLE");
    BLE.setAdvertisedService(customService);
    customService.addCharacteristic(accCharacteristic);
    customService.addCharacteristic(gyroCharacteristic);
    customService.addCharacteristic(magCharacteristic);
    BLE.addService(customService);
    BLE.advertise();
```

In the loop function I checked if Ble is connected to raspberry pi and if it is connected and while connected I read the sensor data from 3 sensors and each of them has 3 axis.

```
void loop() {
    if (BLE.connected()) {
        while (BLE.connected()) {
            float ax, ay, az, gx, gy, gz, mx, my, mz;

            if (IMU.accelerationAvailable() && IMU.gyroscopeAvailable() && IMU.magneticFieldAvailable()) {
                IMU.readAcceleration(ax, ay, az);
                IMU.readGyroscope(gx, gy, gz);
                IMU.readMagneticField(mx, my, mz);
```

Then I format the data with 2 floating point precision. After each data formatted I wrote the values into String characteristics of Ble they belong to. After each data sending I delay for 50 milliseconds and after 1 full loop I delay 200 milliseconds to wait the data to be transmitted.

```
                String accData = "a:" + String(ax, 2) + "," + String(ay, 2) + "," + String(az, 2);
                String gyroData = "g:" + String(gx, 2) + "," + String(gy, 2) + "," + String(gz, 2);
                String magData = "m:" + String(mx, 2) + "," + String(my, 2) + "," + String(mz, 2);

                accCharacteristic.writeValue(accData);
                delay(50);
                gyroCharacteristic.writeValue(gyroData);
                delay(50);
                magCharacteristic.writeValue(magData);
                delay(200);
    }
```

Raspberry Pi Side:

In this part I define a thread that works in background. It is necessary because the BLE connection between Arduino and Raspberry Pi shouldn't stop. I define a web socket to be available from all addresses means it will be accessed from everywhere and its port on raspberry pi is 5000. “index” method run when the client accesses the root page. It renders the index.html file that I provide later and also has javascript text to render graph also.

```python
@app.route('/')
def index():
    return render_template('index.html')

def background_thread():
    device_address = "7a:d8:ee:7d:4d:1e"
    connect_and_listen(device_address, socketio)

if __name__ == '__main__':
    thread = threading.Thread(target=background_thread)
    thread.daemon = True
    thread.start()
    socketio.run(app, host='0.0.0.0', port=5000)
```

In the next part I define a connect_and_listen function that connects to Arduino Ble using bluepy library of python. This creates a peripheral object and gets Service using the above defined in Arduino unique identifier.It then checks each String characteristics from Arduino Ble and receives the message. In the later loop I wanted to wait for notifications from Arduino Ble.

```python
def connect_and_listen(device_address, socketio):
    try:
        nano_peripheral = Peripheral(device_address)
        nano_peripheral.setDelegate(MyDelegate(socketio))

        print("Connected to Nano33BLE")
        nano_service = nano_peripheral.getServiceByUUID("12345678-1234-1234-1234-123456789ABC")

        # Subscribing to notifications for each characteristic
        for characteristic in nano_service.getCharacteristics():
            if characteristic.uuid == "87654321-4321-4321-4321-210987654321" or \
               characteristic.uuid == "87654321-4321-4321-4321-210987654322" or \
               characteristic.uuid == "87654321-4321-4321-4321-210987654323":
                nano_peripheral.writeCharacteristic(characteristic.valHandle + 1, b"\x01\x00")

        while True:
            if nano_peripheral.waitForNotifications(1.0):
                continue

    finally:
        nano_peripheral.disconnect()
        print("Disconnected")
```

In the last part I set up Flask object which is necessary to start Flask web framework. I define an encryption for my app which is the Secret key. I define cross origin Socket. In MyDelegate class I inherit from DefaultDelagate and initialize socket objects. After the notification of data received from the Ble device I emit the data to my website.

```
app = Flask(__name__)
app.config['SECRET_KEY'] = b'*z\xf8\x99i\xed\xc6\xb1\xeb\xd6\xc4?\xde\xabp\xac\xa8\xd0j\x81"x3F'

# Allow cross-origin requests
socketio = SocketIO(app, cors_allowed_origins="*")

class MyDelegate(DefaultDelegate):
    def __init__(self, socketio):
        super().__init__()
        self.socketio = socketio

    def handleNotification(self, cHandle, data):
        decoded_data = data.decode()
        if decoded_data.startswith('a:') or decoded_data.startswith('g:') or decoded_data.startswith('m:'):
            self.socketio.emit('sensor_data', {'data': decoded_data})
```

The next part is about the code in index.html file which consists of the html and javascript code.

I define the title, font size, graph sizes and statistics data positions here:

```
<!DOCTYPE html>
<html>
<head>
    <title>Sensor Data Charts</title>
    <script src="https://cdn.socket.io/3.1.3/socket.io.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <style>
        .chart-container {
            display: flex; /* Use flexbox to align chart and stats */
            flex-direction: column; /* Stack them vertically */
            margin-bottom: 40px; /* Space between chart groups */
            align-items: center; /* Center charts */
        }
        .chart-wrapper {
            display: flex; /* Use flexbox to align chart and stats */
            align-items: center; /* Align items vertically */
            margin-bottom: 200px; /* Space between each chart */
        }
        canvas {
            width: 450px; /* Reduced width for canvas */
            height: 200px; /* Reduced height for canvas */
        }
        .stats {
            display: flex; /* Use flexbox for stats */
            flex-direction: column; /* Stack stats vertically */
            justify-content: center; /* Center stats vertically */
            padding-left: 20px; /* Space between chart and stats */
            font-size: 30px; /* Increased font size for better visibility */
            min-width: 200px; /* Ensure stats block does not shrink */
        }
        button {
            margin-right: 5px;
            padding: 5px 10px;
        }
        #charts-container {
            margin-top: 20px;
        }
    </style>
</head>
```

In the next part of html code I defined 3 buttons for each sensors. When these buttons are pressed the real time sensor data is displayed in graphs and their statistics (mean, min, max, std deviation) are displayed next to corresponding graphs.

```
<body>
    <h1>Sensor Data from Arduino Nano 33 BLE</h1>
    <div>
        <button onclick="showCharts('acc')">Accelerometer</button>
        <button onclick="showCharts('gyro')">Gyroscope</button>
        <button onclick="showCharts('mag')">Magnetometer</button>
    </div>
    <div id="charts-container">
        <div id="accCharts" class="chart-container">
            <div class="chart-wrapper">
                <canvas id="accXChart"></canvas>
                <div id="accXStats" class="stats"></div>
            </div>
            <div class="chart-wrapper">
                <canvas id="accYChart"></canvas>
                <div id="accYStats" class="stats"></div>
            </div>
            <div class="chart-wrapper">
                <canvas id="accZChart"></canvas>
                <div id="accZStats" class="stats"></div>
            </div>
        </div>
```

The I write a script to connect to websocket which was defined in the Flask app.I define 9 arrays which correspond to each axis of each sensors.

```
<script>
    var socket = io.connect();

    var accXData = [], accYData = [], accZData = [];
    var gyroXData = [], gyroYData = [], gyroZData = [];
    var magXData = [], magYData = [], magZData = [];

    function createChart(ctx, data, label) {
        return new Chart(ctx, {
            type: 'line',
            data: {
                labels: Array(data.length).fill(''),
                datasets: [{
                    label: label,
                    data: data,
                    borderWidth: 1
                }]
            },
            options: {
                scales: {
                    y: {
                        beginAtZero: true
                    }
                }
            }
        });
    }
```

After I define variables for each axis Chart. I parse the data received from Flask app.Then I assign each data to each graph.

```
var accXChart = createChart(document.getElementById('accXChart'), accXData, 'Accelerometer X');
var accYChart = createChart(document.getElementById('accYChart'), accYData, 'Accelerometer Y');
var accZChart = createChart(document.getElementById('accZChart'), accZData, 'Accelerometer Z');
var gyroXChart = createChart(document.getElementById('gyroXChart'), gyroXData, 'Gyroscope X');
var gyroYChart = createChart(document.getElementById('gyroYChart'), gyroYData, 'Gyroscope Y');
var gyroZChart = createChart(document.getElementById('gyroZChart'), gyroZData, 'Gyroscope Z');
var magXChart = createChart(document.getElementById('magXChart'), magXData, 'Magnetometer X');
var magYChart = createChart(document.getElementById('magYChart'), magYData, 'Magnetometer Y');
var magZChart = createChart(document.getElementById('magZChart'), magZData, 'Magnetometer Z');

socket.on('sensor_data', function(data) {
    var values = data.data.split(':')[1].split(',');
    var x = parseFloat(values[0]), y = parseFloat(values[1]), z = parseFloat(values[2]);

    if (data.data.startsWith('a:')) {
        updateChartData(accXChart, accXData, x, 'accXStats');
        updateChartData(accYChart, accYData, y, 'accYStats');
        updateChartData(accZChart, accZData, z, 'accZStats');
    } else if (data.data.startsWith('g:')) {
        updateChartData(gyroXChart, gyroXData, x, 'gyroXStats');
        updateChartData(gyroYChart, gyroYData, y, 'gyroYStats');
        updateChartData(gyroZChart, gyroZData, z, 'gyroZStats');
    } else if (data.data.startsWith('m:')) {
        updateChartData(magXChart, magXData, x, 'magXStats');
        updateChartData(magYChart, magYData, y, 'magYStats');
        updateChartData(magZChart, magZData, z, 'magZStats');
    }
});
```

In this part I calculate the real time statistics on sensor data and show the corresponding statistics next to their graphs.

```
function updateStats(dataArray, statsDivId) {
    var avg = dataArray.reduce((a, b) => a + b, 0) / dataArray.length;
    var min = Math.min(...dataArray);
    var max = Math.max(...dataArray);
    var stdDev = Math.sqrt(dataArray.map(x => Math.pow(x - avg, 2)).reduce((a, b) => a + b) / dataArray.length);

    document.getElementById(statsDivId).innerHTML = `Avg: ${avg.toFixed(2)}, Min: ${min.toFixed(2)}, Max: ${max.toFixed(2)}, Std Dev:
}
```

I display the charts only if the corresponding sensor button is clicked.

```
// Function to show charts based on button clicked
function showCharts(sensorType) {
    // Hide all chart containers
    document.getElementById('accCharts').style.display = 'none';
    document.getElementById('gyroCharts').style.display = 'none';
    document.getElementById('magCharts').style.display = 'none';

    // Show the selected chart container
    document.getElementById(sensorType + 'Charts').style.display = 'block';
}
```

,

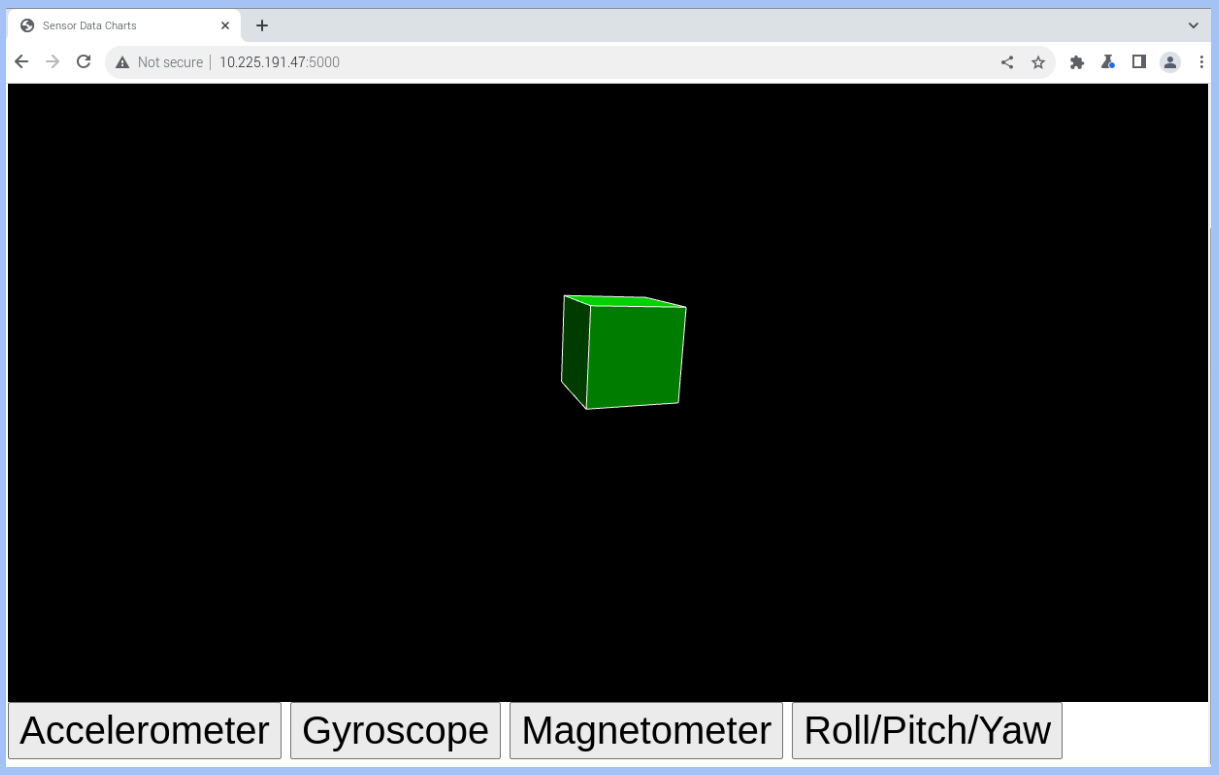
I also calculated yaw, pitch and roll values using accelerometer and gyroscope data. These define the orientation of Arduino Nano 33 Ble. These values can also be used in balance applications like multicopter or robots.

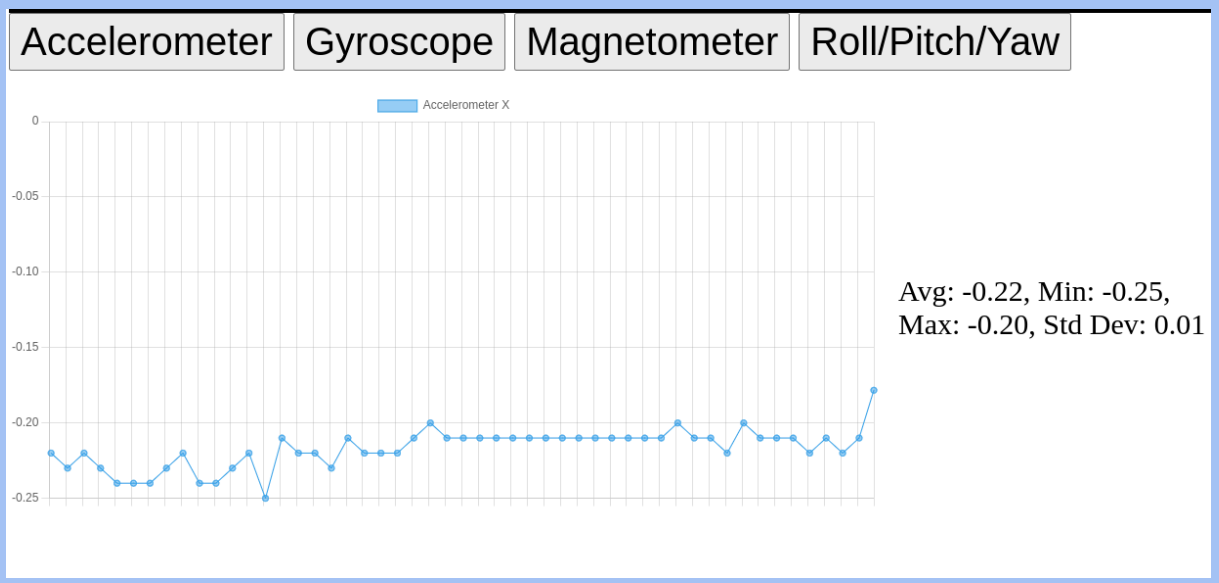
Our job is not done yet. Even though I can access the website from the local network that is Eduroam I can not access it from outside yet. For this I had to configure a network tunneling because I dont access the admin rights to

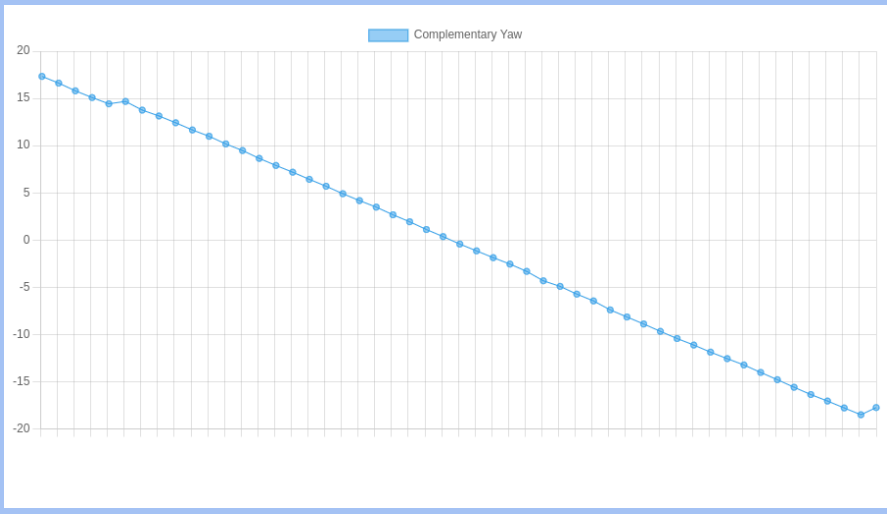
eduroam. For this I used Ngrok. I set up the Ngrok and it provided me with some random url. However I subscribe I could set up a static url instead.

Note: Some trivial parts of javascript code are excluded in the report.

RESULTS:







Avg: -0.53, Min: -19.42, Max: 17.33, Std Dev: 10.98