Your Orthopedic Dynamic Alignment

NASA Wearables Challenge

Amisha Anand, Chloe Bowen, Marek Gutke,

Jeffrey Hickman, Lyla Toth

University of Colorado Boulder

Collette Wilfong, Ayush Srivastava, Barbra Sobhani.

Caleb Johnson, Annie Strange

Table of Contents

Abstract

Poor posture develops gradually, and these adjustments to the spine often go unnoticed; This leads to long-term discomfort in sedentary individuals such as students. Many existing posture monitoring systems rely on single sensors, which are unable to fully distinguish between healthy bending motions and harmful postural behaviors (e.g., slouching). This research, conducted by the Wearables Team, presents the design and validation of a wearable posture monitoring system integrated into a compression shirt for daily use. The system uses two inertial measurement units placed along the spine at the upper thoracic and lower lumbar regions. By calculating the relative angle between spinal segments, the system aims to distinguish healthy bending from poor posture more accurately than other single-sensor approaches.

Data is processed using a microcontroller, logged locally, and transmitted wirelessly via Bluetooth for analysis and user feedback. Preliminary results have shown that the inertial measurement units have the capacity to be used for angle measurements between the regions of the back. It is hypothesized that once the Bluetooth system in the microcontroller is functioning and the inertial measurement units can collect readable data, the spinal angles can be exported and analyzed successfully. Overall, this project investigates whether relative spinal angle measurements improve posture classification, how sensor placement can affect sensor detection, and whether a compression shirt can maintain constant sensor alignment during motion.

Introduction

There are some existing posture monitoring systems attempting to address this issue; however, they rely on a single sensor placed on the back [1]. While these systems can detect general movement, they often struggle to differentiate between natural, healthy bending motions (such as leaning forward briefly) and harmful posture habits like sustained slouching. This limitation reduces their accuracy and usefulness in real world situations. The goal of this project is to design and evaluate a wearable posture monitoring system that eliminates these limitations by using two inertial measurement units (IMUs) positioned at different points along the spine. By measuring the relative angle between the upper thoracic and lower lumbar regions, the system aims to provide an accurate representation of spinal alignment. Additionally, this project explores whether integrating the sensors into a compression shirt can improve consistency in sensor placement and data reliability during movement. The system is designed not only to collect meaningful posture data but also to be comfortable and practical for everyday wear.

Research in biomechanics and spinal alignment helped indicate that posture-related issues most commonly occur in the thoracic and lumbar regions of the spine, where curvature changes are the most pronounced during slouching. The thoracic spine (upper back) is more associated with kyphotic curvature, while the lumbar spine (lower back) maintains a lordotic curve. Poor posture typically exaggerates thoracic curvature while reducing lumbar support. Based on this research, sensors were internally placed at the upper thoracic and lower lumbar regions to effectively capture the relative change between these two key segments. Measuring the angular difference between these regions allows for a more biomechanically meaningful assessment of posture compared to single point measurements due to a more accurate reading of how the spine deforms rather than simply how it tilts [2], [3].

Materials and Methods

The wearable system is integrated into a compression shirt via a fabric shell to maintain close contact with the body and ensure consistent sensor alignment throughout movement (see Figure 2). This design choice helps minimize shifting of the sensors relative to the spine, which is important for collecting accurate and reliable data over time. The compression material applies even pressure across the back, helping keep the sensors in fixed positions during bending or dynamic motion. The shell is attached to the compression shirt by using Velcro applied vertically along the spine and in line with the sensors (Figure 1). The fixed position of the sensors and the fabric shell are also assisted by three Velcro straps that limit the movement of the fabric shell horizontally. At the same time, the system is designed with comfort and flexibility in mind so that it can be worn during normal daily activities without restricting motion. This balance between stability and wearability is critical, as the system is intended for extended use rather than short term testing only. As a result, ergonomics and comfort were a priority in structural design.

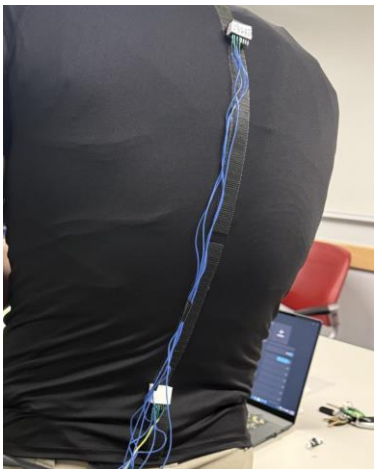

Figure 1: Sensors and wiring outside protective casing on compression shirt.

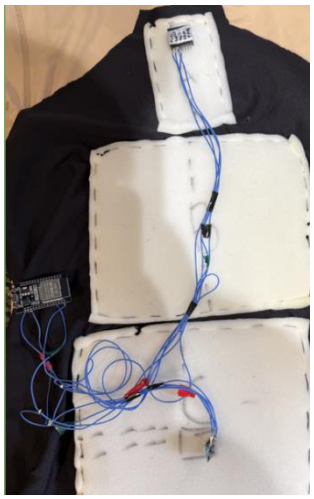

Figure 2: Internal foam layer for sensor protection and comfort

Figure 3: Sensors and wiring inside protective casing on compression shirt.

The primary sensors used in this project are MPU-6050 inertial measurement units (IMUs), which collect both gyroscopic and accelerometer data, for a total of six degrees of freedom. These raw data points are processed using an algorithm called the Madgwick filter [8], which converts them into quaternions (a four-dimensional representation of orientation in 3D space). Quaternions are used because they provide a stable and continuous way to track rotation [10]. The two streams of noisy data (the accelerometer and gyroscopic data) are put through a function that combines the data into one stream of clean angle measurements, which reduces overall inconsistencies, and accurately produces angle measurements. From these quaternions, orientation vectors are generated for each sensor to represent their direction in space. The angle between the two IMUs is then calculated using a dot product between these two normalized vectors, allowing us to determine the relative curvature of the spine between the upper and lower back. All the sensor data is processed using an ESP32 microcontroller, which was selected for its compact size, low power consumption, affordable cost, and built in Bluetooth capabilities. The ESP32

receives data from both IMUs simultaneously, performs the necessary calculations and filtering, and prepares the data for wireless transmission.

To support the hardware, a protective casing system was designed and manufactured to securely house the sensors and maintain their placement (see Figure 4). This casing holds the IMUs in fixed positions using custom 3D printed enclosures that match the size and shape of the sensors. Foam padding is included around each IMU to reduce the effects of impact, vibration, and sudden movement, which could otherwise damage the sensors or introduce additional noise into the data. The foam is attached onto an elastic fabric backing, which not only secures the components but also allows the system to flex naturally with the user's body. This flexibility is important for maintaining both comfort and consistent sensor contact. Zippers are incorporated into the design at the top and bottom of the shell so that the user can easily access and adjust the senor position if needed (see Figure 3). The ESP32 and battery are housed separately in a 3D printed holster that attaches externally to a belt or waistband (see Figure 5). This separation helps reduce bulk on the back and prevents additional weight from interfering with sensor readings. In future iterations, this holster could be integrated directly into the compression shirt, such as through a side pocket, to improve usability and create a more compact system.

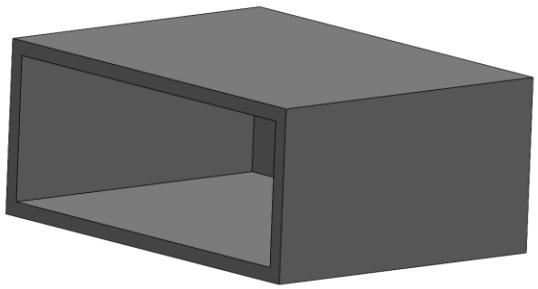

Figure 4: 3D printed IMU holder (one for each IMU).

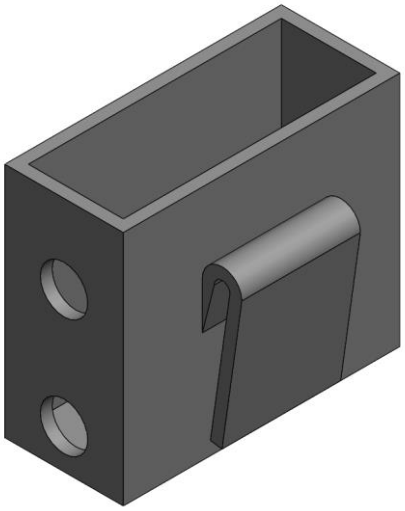

Figure 5: 3D printed holster for the battery pack and ESP32

After the IMU data is converted on the microcontroller, the data then uses a Bluetooth Low Energy (BLE) system to display on the website. A BLE is commonly used in wearables and sensors, and it is an efficient, low-power piece of technology [9]. Using timestamps, the angle data is graphed in real time on the user interface, allowing communication of the change in posture to reach the user. The BLE operates in short bursts, so data is collected throughout the day to create a comprehensive summary of the sensor values. In building the website framework, a combination of JavaScript, HTML, and CSS files was used. The website handles user authentication, Bluetooth Low Energy device communication, CSV data ingestion, and posture visualization through Chart.js. The website also contains the option for manual upload of CSV files for the user to see past data (Figure 6).

Figure 6: BLE connection buttons, data attaining buttons, and angle calibration button.

The file structure of the microcontroller and website can be reduced to this:

- Front End/User interface
  - index.html — Page structure, authorization screen, home screen, and navigation
  - index1.js — Application logic
  - style.css — Styling and layout
- Back End
  - _Current_IMU_Code.ino — ESP32 Code for data collection

The Bluetooth implementation allows us to connect the backend to front end code efficiently, so the website updates in a seamless fashion. The process of data movement from the microcontroller to interface can be seen in Figure 7.

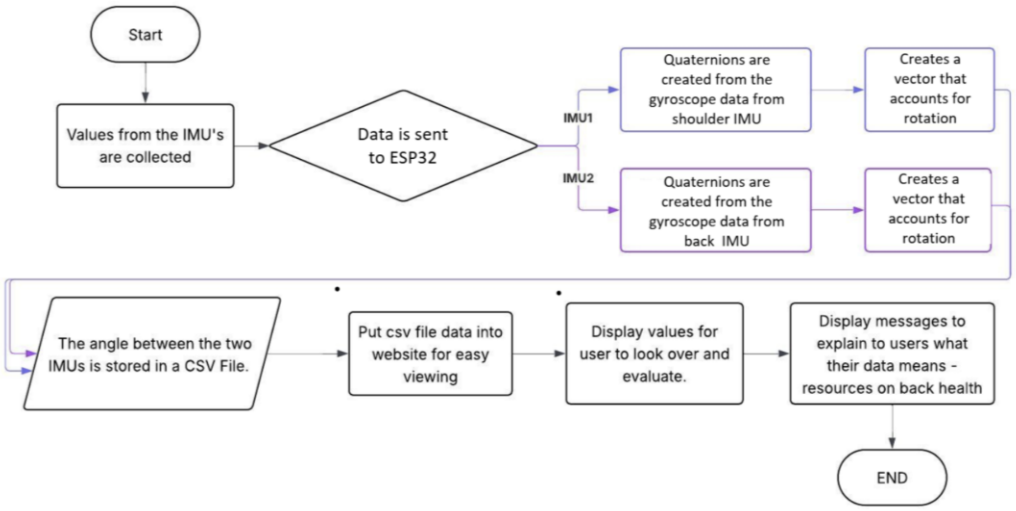

Figure 7: Software Flowchart

The JavaScript file updates the content on the website value via Bluetooth, allowing the user to access the microcontroller data collected by the IMUs. The user can then save this information by downloading the displayed angle data graphs. The website is currently running locally on the user's personal laptop. As this is a current prototype, there is room for improvement in the structure of accessing the website. The website also contains a status bar based on the angle configurations in Figure 9, which notifies the user immediately in case of bad posture. Outside of data collection, the website also includes

resources for the users to inform themselves about posture in general as well as simple ways to improve on their own (Figure 8).

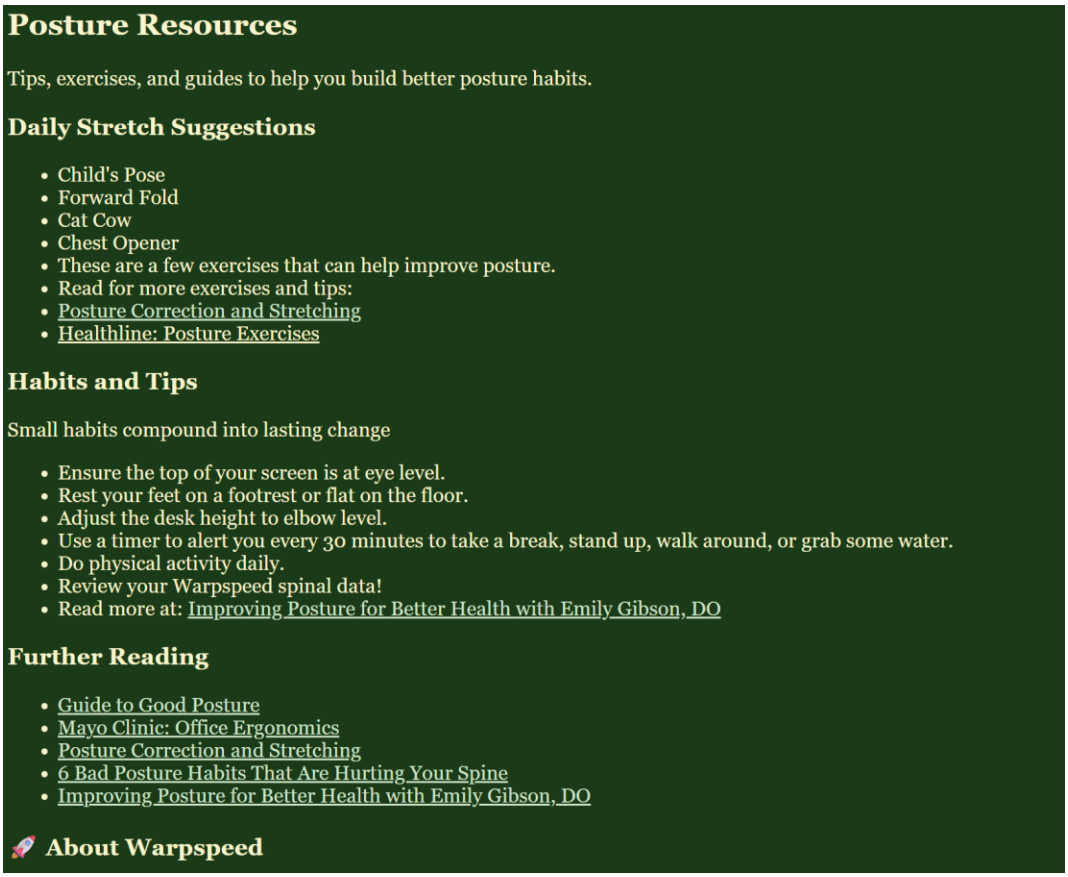

Figure 8: Posture Resources

Most of the mathematic data processing is done directly on the ESP32 Microcontroller. It takes an integral of gyroscopic data from the IMUs to express their orientation as a quaternion [10]. These quaternions are then passed through an accelerometer-based gradient descent filter (Modified Madgwick filter, beta set to 0.05 - see Appendix A.1 for code) to correct gyroscopic drift in the pitch and roll directions. Because the IMUs lack a magnetometer, yaw drift cannot be corrected, so the quaternion only represents orientation in the pitch and roll directions. When the vectors are normalized, the dot product of the two IMU quaternions can be taken to get the angle between them. Further drift correction is done by keeping the IMUs constrained to +/- 0.5deg when the accelerometer reads values near gravity (meaning the IMUs are stationary). Additionally, the zero point for this angle is set as the user's good posture, which can be calibrated on the user interface.

Overall, the ESP32 connects to both IMUs through a foam shell that runs down the back of the spine. The microcontroller and battery casing are attached to a belt/waistband accessory, while wires are routed through the foam enclosure to each sensor, keeping them organized and protected. As the IMUs collect data during movement, the ESP32 processes it in real time. The calculated angle, along with a qualitative 'good', 'mild/slight bend', 'poor/slouching', or 'bad/severe bend' posture reading are uploaded

to a website via Bluetooth LE (see Figure 8), where the data is displayed on a graph; This system provides easy access to the angle outputs and analysis of the user's posture over time.

```
const char* posture;
float absBackAngle = fabsf(backAngle);
if (absBackAngle < 10)      posture = "GOOD - Straight";
else if (absBackAngle < 20) posture = "MILD - Slight bend";
else if (absBackAngle < 35) posture = "POOR - Slouching";
else                        posture = "BAD - Severe bend";

// Serial debug (every loop ~ 10ms)
Serial.print("Angle:");
Serial.print(backAngle, 2);
Serial.print(" Status:");
Serial.println(posture);
```

Figure 9: Angle benchmarks and qualitative comments in the VS code. Can be found in Appendix A.

Testing

For initial testing of the IMU system, we only used the acceleration vectors to compute the relative angle between the two IMUs. This provided a basic understanding of how posture changes affected the measured angle between the sensors. However, under certain orientations the usage of the 3D acceleration vectors decreased the number of degrees of freedom to which we were able to represent the angle, leading to inaccurate values. This is documented phenomena known as gimbal lock, which we solved by switching our representation of orientation to be 4D quaternions. We also saw large jumps in data during acceleration since the accelerometer has no way of distinguishing the acceleration from motion or from gravity.

These results led to the current system: quaternion multiplication, accelerometer based bias correction, and the Madgwick sensor fusion filter. This process of computing the angles was adjusted multiple times as we continued to test the accuracy of the IMUs. At first a major problem we saw was that angle values would drift - increase overtime - which led to only accounting for pitch and roll orientations. A calibration button (Figure 6) on the user interface was added so that the system could set the individual user's good posture benchmark as 'zero' degrees. With motion of the back, the angle would increase or decrease, but once the posture returned to the 'normal' orientation, the angle would return to a zero value. This calibration became more accurate in testing with the additions in our code.

One such addition was adding a constraint to the inverse sine function, which eliminates small rounding errors that could compound, so that the resulting values always remained between -1 and 1. The reduced floating point rounding errors overtime achieved more accurate angles over long run times. The gyroscope bias was also edited to check for stillness after subtracting the calibration offset instead of before, which significantly improved accuracy while the IMUs were moving. The way that the overall relative angle was computed was also changed. Instead of the dot product approach of the original formula, the product between the conjugate of one quaternion and the raw quaternion of the other IMU is computed. This is known as the relative quaternion, and pitch is extracted by taking the arcsine of the components representing pitch. What resulted was a much more accurate representation of the relative pitch between the two quaternions. The exact formula used is pictured in Figure 10.

```
float angleBetweenOrientations(MadgwickFilter &f1, MadgwickFilter &f2) {
  float rw =  f1.q0*f2.q0 + f1.q1*f2.q1 + f1.q2*f2.q2 + f1.q3*f2.q3;
  float rx = -f1.q0*f2.q1 + f1.q1*f2.q0 + f1.q2*f2.q3 - f1.q3*f2.q2;
  float ry = -f1.q0*f2.q2 - f1.q1*f2.q3 + f1.q2*f2.q0 + f1.q3*f2.q1;
  float rz = -f1.q0*f2.q3 + f1.q1*f2.q2 - f1.q2*f2.q1 + f1.q3*f2.q0;
  float pitch = asinf(constrain(2.0f * (rw*ry - rz*rx), -1.0f, 1.0f));
  return pitch * (180.0f / PI);
}
```

Figure 10: Quaternion Product and Pitch angle calculation.

Other factors in the accuracy was the adjustment of the beta value (the gradient descent multiplier), and the values used to detect stillness. The beta value is a measure of how much the filter trusts the accelerometer. The value used for the IMUs was on the larger end; typical beta values on a Madgwick filter are 0.033. In this project, a value of 0.1 wass used due to our decreased IMU quality and lack of degrees of freedom. Overall, this combined system decreased the error in the collection of angles to only vary by +/- 0.2deg when still, as seen in Figure 11. Accuracy in reading angles larger than about 60 degrees remained low due to the lack of yaw orientation, which is usually collected by magnetometers seen in IMUs with nine degrees of freedom.

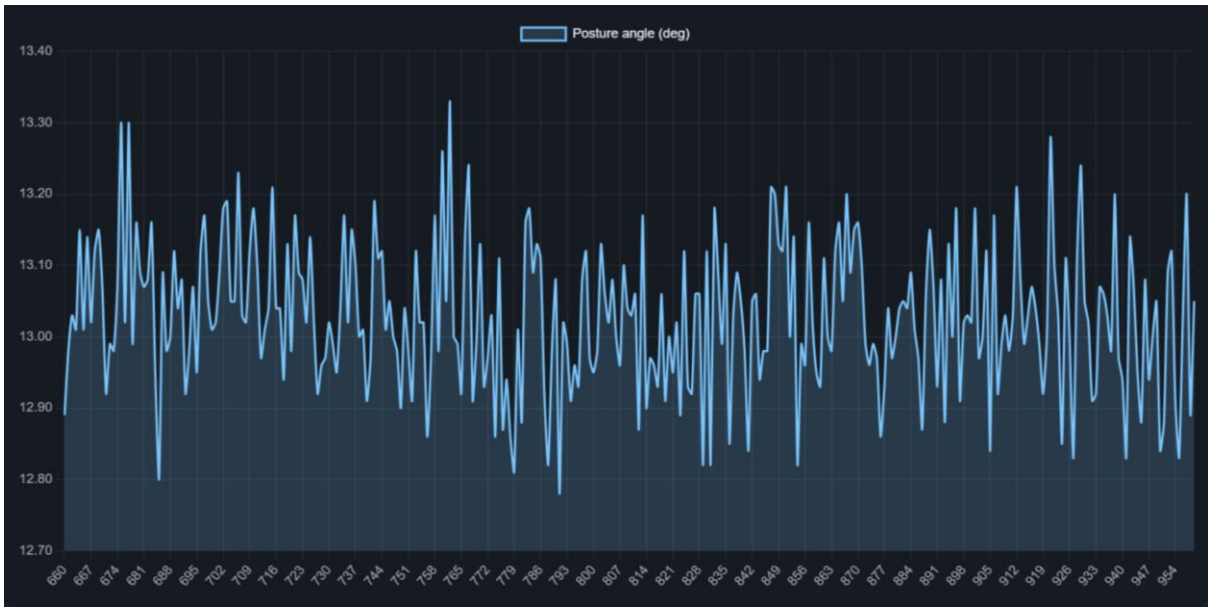

Figure 11: Error in angle calculations from testing of the IMU system.

Results and Analysis

During controlled testing, the user was asked to perform a variety of movements, including standing upright, slouching forward, bending forward initially, and moving naturally. These movements were selected to simulate both healthy and unhealthy posture behaviors. Data was recorded throughout these tests to observe how the system responded to each type of motion. Additional testing focused on wearability and comfort. Users wore the system for extended periods to evaluate whether the compression of shirt-maintained sensor alignment and if the device remained conformable during normal activities (see Figure 12). Observation was made regarding sensor shifting, data stability, and overall usability.

Figure 12: Y.O.D.A system attached to the user.

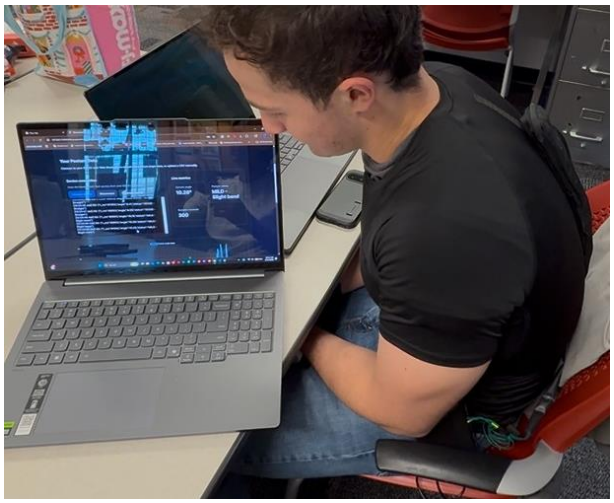

Figure 13: Real time posture angle data displayed on the prototype user interface during testing.

During testing, the system successfully detected distinguishable differences between upright posture, slouching, and forward bending motions (Figure 12). Upright posture consistently produced angle values near the calibrated baseline of 0 +/- 0.2, while slouching resulted in a sustained increase in the relative angle between the sensors. Forward bending produced larger but temporary angle deviations (see Figure 14).

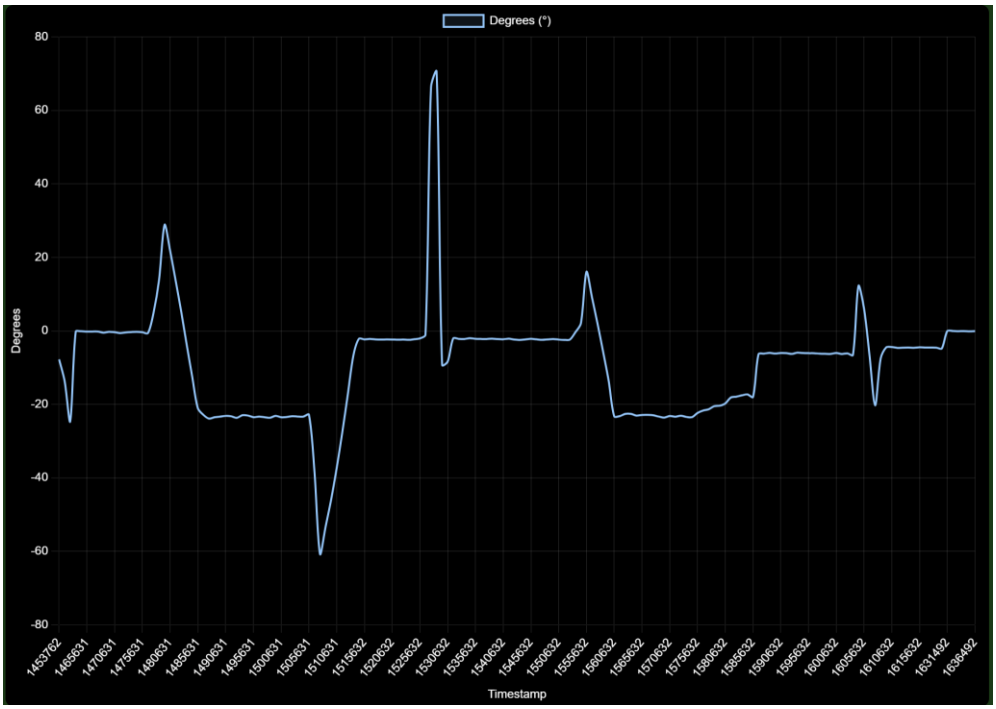

Figure 14: Live updated graph on the user interface with consistent good posture values, consistent 'poor' posture, and spikes of 'bad' posture. Timestamp is in milliseconds and side bar records angles in degrees.

Battery performance testing showed that the system could operate continuously for several hours, depending on Bluetooth usage and sample rate. Electronic connections remained stable during most testing scenarios, though occasional noise and signal interruptions were observed during rapid movement. These inconsistencies were minimized through filtering techniques recorded in the testing section and improved wire management with the ESP32 wire casing.

Wearability testing revealed that the compression shirt was generally effective at maintaining sensor alignment during normal activity. However, slight sensor shifting occurred during extreme bending or twisting motions, which gave minor variations in the angle of readings. Despite this, during normal movements (sitting, standing), the sensors remained accurate for their technological capacities.

Conclusion and Recommendations

This project successfully demonstrated the feasibility of a wearable posture monitoring system that uses dual IMUs to measure relative spinal alignment. The results suggest that measuring the angle between two points on the spine provides a more accurate representation of posture than traditional systems available to consumer market. While the system showed promising performance, some limitations were identified, including battery life constraints and minor sensor shifting during extreme motion. Despite these challenges, the system met its primary goal of detecting posture changes and distinguishing between healthy and unhealthy movement patterns.

During testing, improvements and next design steps that could be implemented were ideated. As an alternative to a six degree of freedom IMU, a nine degree of freedom IMU, involving a magnetometer would greatly increase accuracy at large angles. In addition, more points of data would reduce the compute power required and complexity of the embedded system since the magnetometer would find yaw orientation and add another layer of correction to the pitch and roll angles. The extreme motion of the IMUs could also be improved by adding a sliding track system to the shell, keeping the IMUs in one place vertically while also accommodating different spine structures and user size. Including data storage via a Secure Digital (SD) card would make data more accessible if the user wanted to compare spine data over a long period of time. However, if the user wanted to have feedback in the short-term, in-the-moment correction could be achieved through a small buzzer system hooked up to the ESP32, notifying the user about the changes in their posture based off the ESP32 results. In future iterations of the project, a main point of value to add is data handling and long term data evaluation embedded into the software itself. Adding a web app or cloud based functionality would make this product more accessible to any user.

Through experimentation and testing with this project, some advice to consider in future projects related to wearables would be to start small and ensure that the foundational technology works with the wearable component before adding complexity. However, having a schematic for more complicated implementations from the beginning can make future integration seamless and successful.

References

[1] Upright, “Transform Your Posture, Transform Your Life,” Available: https://www.uprightpose.com/

[2] J. C. Lam and T. Mukhdomi, *Kyphosis*. StatPearls Publishing, 2023. Available: https://www.ncbi.nlm.nih.gov/sites/books/NBK558945/

[3] C. A. Giglio and J. B. Volpon, “Development of thoracic kyphosis and lumbar lordosis,” *PubMed Central (PMC)*. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC12382527/

[4] Arduino, “Bluetooth Communication,” Available: https://docs.arduino.cc/learn/communication/bluetooth/

[5] Android Developers, “Bluetooth Low Energy Overview,” Available: https://developer.android.com/develop/connectivity/bluetooth/ble/ble-overview

[6] Novel Bits, “Bluetooth Low Energy (BLE) Complete Guide,” Available: https://novelbits.io/bluetooth-low-energy-ble-complete-guide/

[7] Arduino, “Curie IMU Raw Data Example,” Available: https://docs.arduino.cc/retired/library-examples/curie-imu/Genuino101CurieIMURawImuDataSerial/

[8] YouTube, “Madgwick Filter Explanation,” Available: https://www.youtube.com/watch?v=0Yvd_k0hbVs

[9] YouTube, “BLE with ESP32 Tutorial Part 2: The Client,” Available: https://www.youtube.com/watch?v=s3yoZa6kzus

[10] YouTube, “Visualizing the 4D Numbers Quaternions”, Available: https://www.youtube.com/watch?v=d4EgbgTm0Bg

[11] Chart.js, “Line Chart”, Available: https://www.chartjs.org/docs/latest/charts/line.html

[12] W3Schools, “JavaScript Tutorial”, Available: https://www.w3schools.com/js/default.asp

[13] W3Schools, “HTML Tutorial”, Available: https://www.w3schools.com/html/

[14] Sheldon Brown, “A Simple Sample Web Page”, Available: https://www.sheldonbrown.com/web_sample1.html

[15] Arduino Forum, "Simplest test code for MPU6050”, Available: https://forum.arduino.cc/t/simplest-test-code-for-mpu6050/1250345

Appendix

Appendix A: Software Code

A.1 The software code for both the microcontroller and website can be found at the link below using the following pathway:

Microcontroller:

WEARABLES---WARPSPEED / Arduino_Code /_Current_IMU_Code /_Current_IMU_Code.ino

Website:

WEARABLES---WARPSPEED / Website Code

https://github.com/A123anand/WEARABLES---WARPSPEED.git