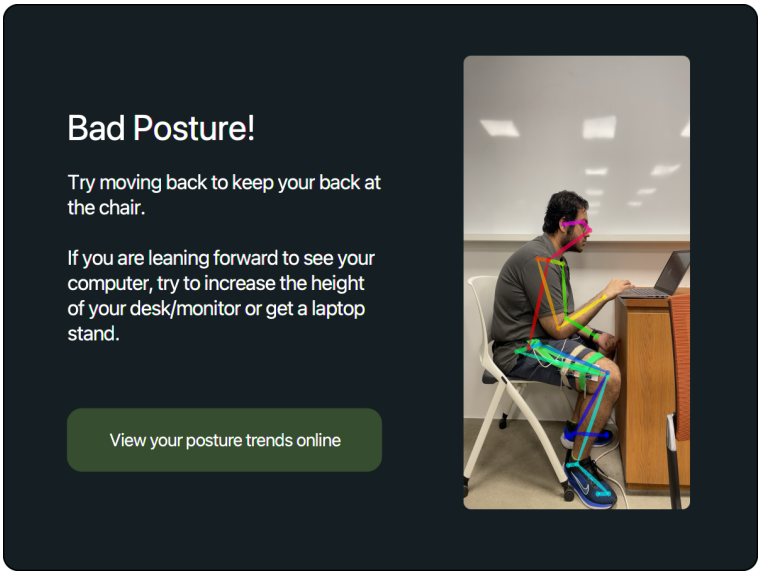
CS3237 Introduction to IoT

Project Report

Team 02: FineSpine



Charisma Kausar - A0226593X

Nicole Joseph - A0226610R

Pratyush Ghosh - A0170549Y

Siew Yang Zhi - A0218295X

Table of Contents

1. Project Overview

Research shows that the average adult professional spends seven to twelve hours a day seated – and as a result of today's work-from-home paradigm, this figure is higher than ever before. These hours spent hunched over take their toll on the body, leading to pain in the neck and back. If left untreated, the consequences of bad posture can be irreversible.

Our IoT system – **FineSpine** – aims to help the user maintain good posture while seated at their desk, and to encourage the user to stand up and get some exercise from time to time.

The system uses a desktop alert system to let the user know when their posture is improper. It also issues reminders – in the form of a buzzer going off – to stand up and walk around when the user hasn't done so for over an hour. Moreover, it provides the user with detailed insights regarding their long-term posture and activity trends via a web dashboard.

2. System Architecture

2.1. Overview

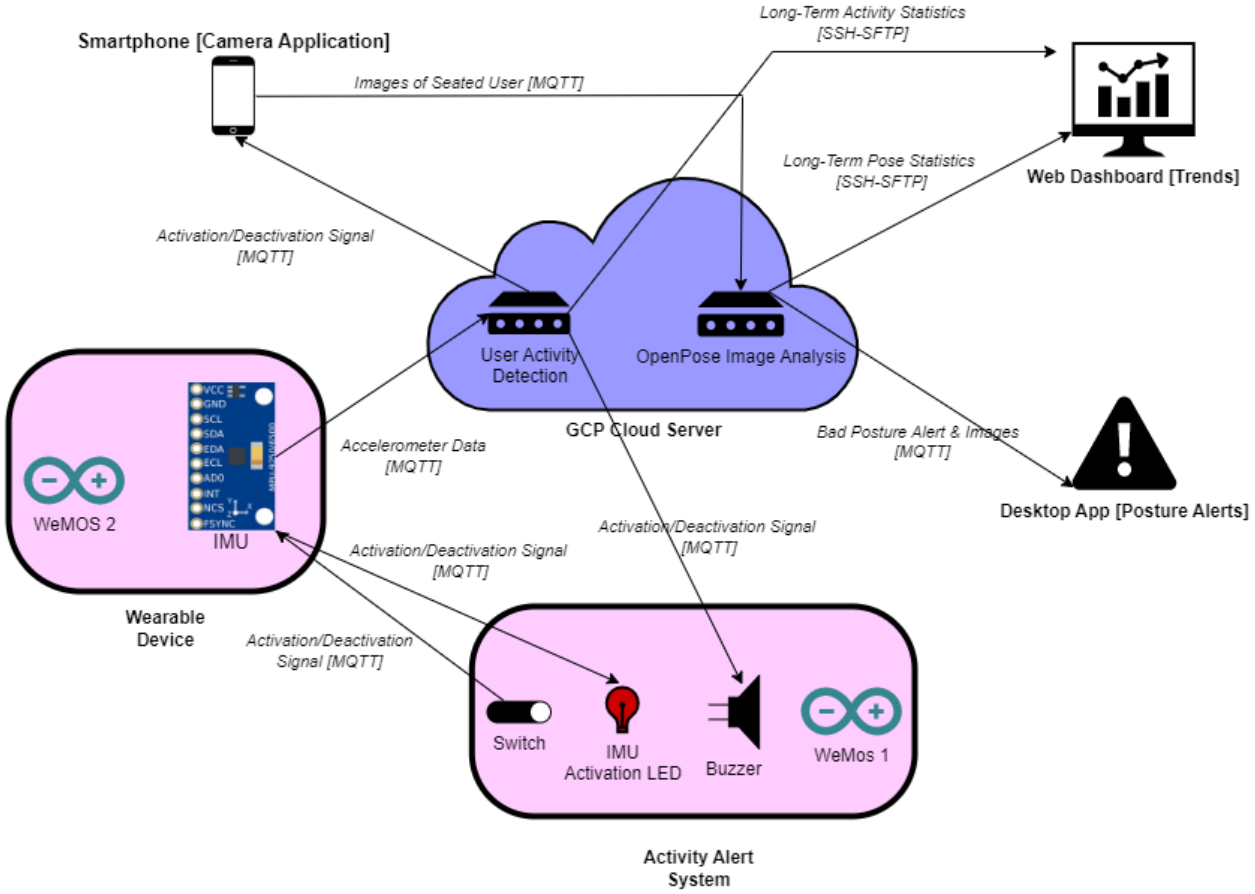


Fig 2.1.1 – System Architecture

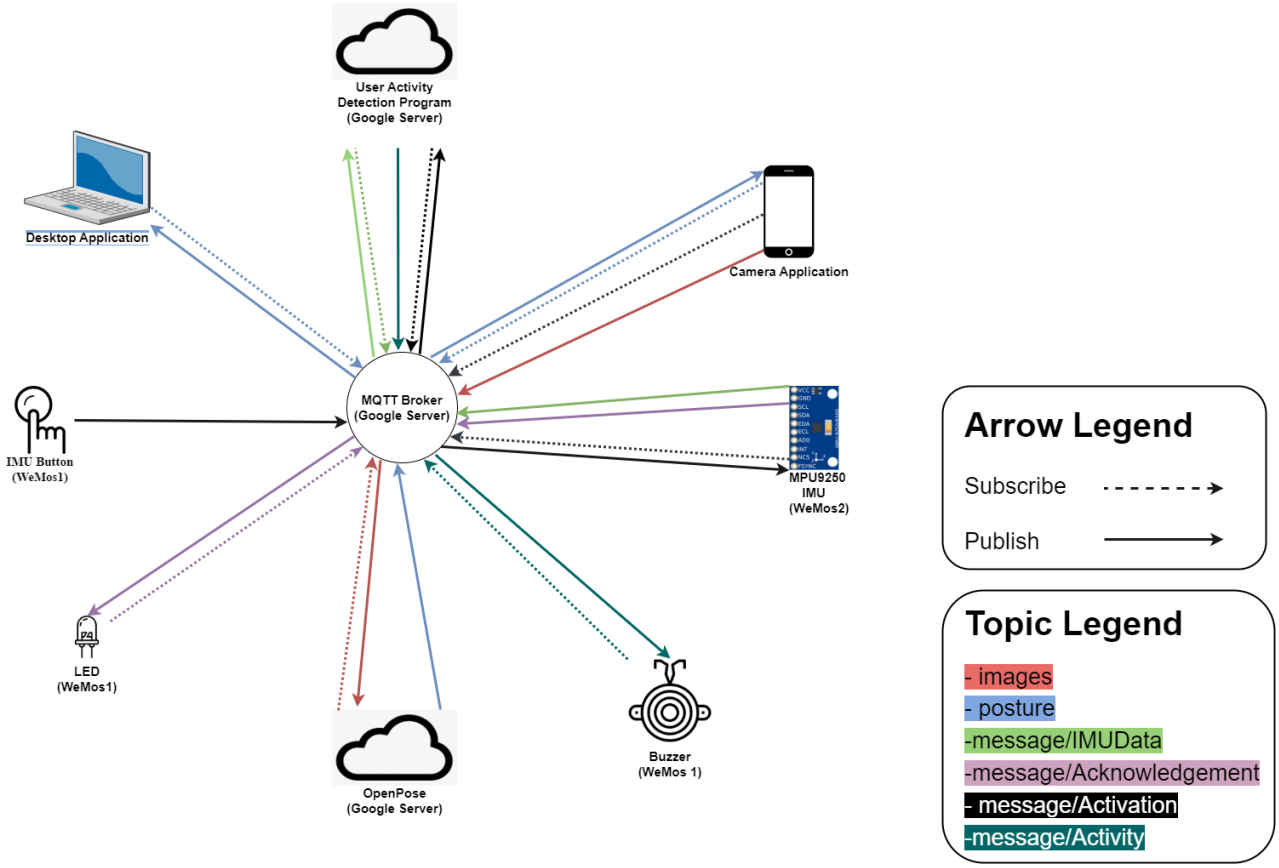
The overall FineSpine system consists of two WeMos devices, as well as several sensors and actuators. WeMos 1 is connected to a buzzer, switch and LED, which together act as the

Activity Alert System that prevents sedentary behavior. WeMos 2 is connected to an Inertial Measurement Unit (IMU) strapped to the user's thigh, which acts as a wearable device that collects accelerometer data from the user, enabling the system to turn on posture detection when they are seated and to sound the buzzer if they haven't been active for an hour. The system also includes a sensor - a multi-platform (iOS/Android) camera application which takes images of the seated user to analyze their posture.

In addition to this, the system includes actuators such as a desktop app which displays alerts when the user is exhibiting poor posture, and a web dashboard which displays long-term trends in the user's posture and activity over the duration of their use of the system.

Finally, the system includes a Google Cloud Platform server running a pair of machine learning models. The first is an LSTM model trained by the team which classifies the user's real-time accelerometer data as Sitting, Standing or Walking. The second is the open-source OpenPose[1] model, which determines the angle of the user's craniovertebral column, and uses this to determine whether their posture is good or bad. This GCP server also runs a Mosquitto broker used by all sensors and actuators in the system to communicate with each other.

Fig. 2.1.2 – Communication Protocol



[1] *CMU-Perceptual-Computing-Lab*. (n.d.). *CMU-Perceptual-Computing-Lab/openpose: OpenPose: Real-time multi-person keypoint detection library for body, face, hands, and foot estimation*. GitHub. Retrieved November 18, 2022, from https://github.com/CMU-Perceptual-Computing-Lab/openpose

The FineSpine system uses the MQTT protocol for communication between its various components. The channels used are as follows:

- **message/Activation** – Used by the switch (to turn on and off the IMU) and the User Activity Detection cloud program (to turn on and off the camera application).
- **images** – Used by the camera application to send captured images of the seated user to the OpenPose cloud program.
- **message/Acknowledgement** – Used by the IMU to cause the LED attached to the Activity Alert System to blink, to let the user know the IMU is operational.
- **message/Activity** – Used by the User Activity Detection cloud program to turn on and off the buzzer attached to the Activity Alert System.
- **message/IMUData** – Used by the IMU to send accelerometer data to the User Activity Detection cloud program.
- **posture** – Used by the OpenPose cloud program to send the user's posture classification to the desktop application, and to send an acknowledgement to the camera application so it can send the next image.

2.2. Sensors

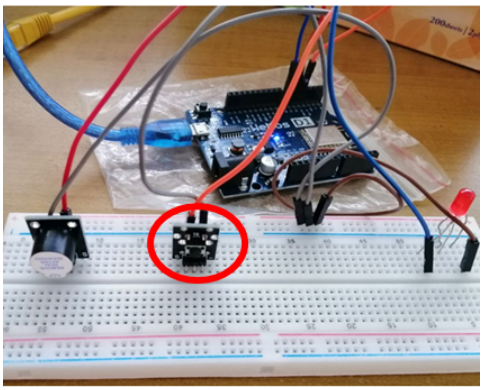


Fig 2.2.1: Activity Alert System (switch circled in red)

Switch: Used to turn on and off the IMU.

MPU9250 - IMU: Used to collect the Accelerometer's x- and y-axis angles to classify the user's activity.

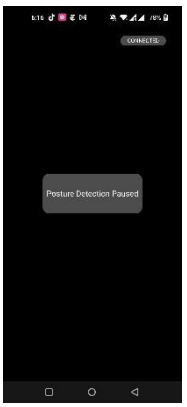


Fig 2.2.2: Android Camera Application

Camera Application: Used to capture side-view images of the user to analyze their posture.

2.3. Actuators

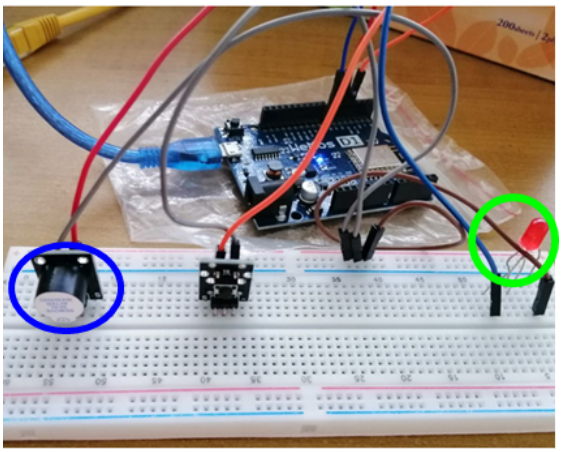


Fig 2.3.1: Activity Alert System (buzzer circled in blue, LED circled in green)

Buzzer: Used to inform the user that he/she has been stationary for an hour. The buzzer would only turn off when the user has walked for 3 minutes.

Red LED: Used to inform the user that the IMU has received IMU Activation Signal On / IMU Activation Signal Off command via MQTT.

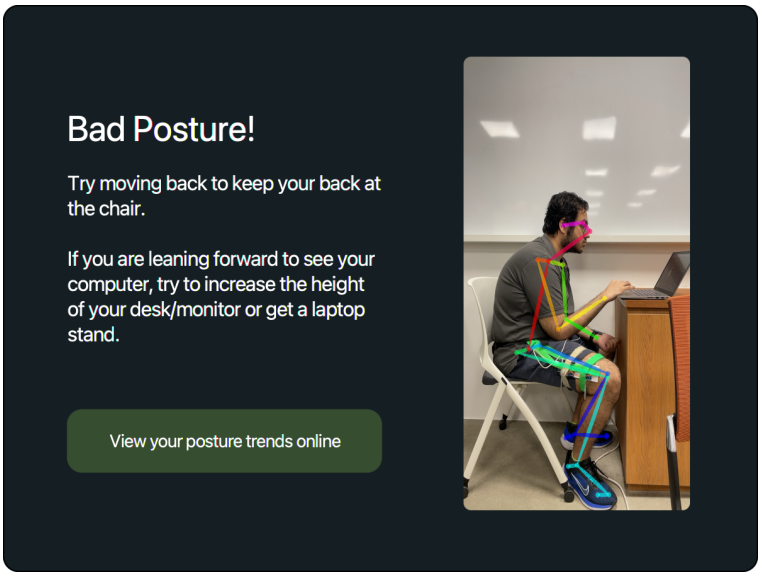


Fig 2.3.2: Desktop App

Desktop Alert App: Used to inform the user that their posture needs to be corrected. Only turns off when bad posture is not detected anymore.

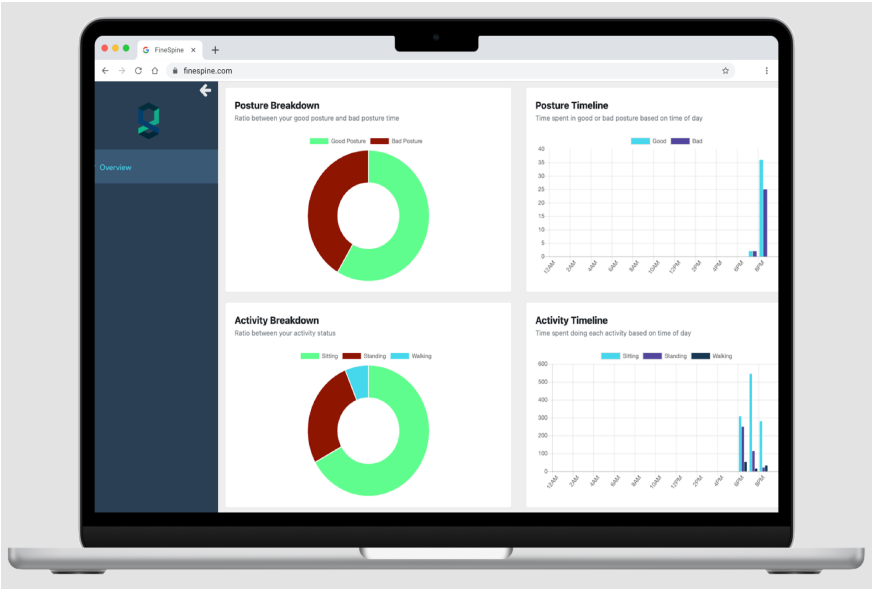


Fig 2.3.3: Web Dashboard

Web Dashboard: Used to provide the user with long-term trend analysis of their activity and posture.

2.4. Cloud

The Google Cloud Platform (GCP) server running on a reserved (static) IP was used as part of this project. Its key technical specifications were as follows:

Operating System	Ubuntu 16.04.7 LTS
CPU(s)	8x vCPUs (Google Compute Engine n1-standard-8)
RAM	30 GB
GPU(s)	1x NVIDIA Tesla K80

3. Implementation

3.1. Activity Detection

IMU Component

When the user presses the switch connected to WeMos 1 (attached to the Activity Alert System), a signal is published to WeMos 2, which turns on the MPU9250 IMU (both attached to the wearable device). In return WeMos 2 sends an acknowledgment signal to WeMos 1, causing the LED to flash. This informs the user that the IMU is on.

[Note: When the IMU is first turned on it carries out auto-calibration to initialize its internal state. The User Activity Model will then use this state as a reference point for the activity 'Sitting'. Therefore, the system requires the user to be seated at the time when they first turn on the IMU.]

WeMos 2, which is attached to IMU, then begins sampling its readings at 20Hz. This sampling rate has been indicated by research[2] to exhibit peak ML classification accuracy for human activity. After each sampling, the x- and y-axis angle accelerometer readings are added to a buffer.

Once the IMU data has been sampled 40 times, the WeMos proceeds to send the buffered sequence to the User Activity Detection program using MQTT.

User Activity Detection Component

[2] Khusainov, R., Azzi, D., Achumba, I., & Bersch, S. (2013). Real-time human ambulation, activity, and physiological monitoring: Taxonomy of issues, techniques, applications, challenges and limitations. *Sensors*, *13*(10), 12852–12902. https://doi.org/10.3390/s131012852

The User Activity Detection program is a Python script hosted on the GCP server. Upon receiving a sequence of IMU data, it passes this sequence of IMU data into the activity classification ML model. The ML model then outputs the detected user activity (Sitting, Standing or Walking).

To prevent mispredictions resulting from temporary activity changes, the system only outputs a change in user activity state if two instances of the new activity are detected consecutively. As a result, the user must maintain their new activity state for roughly 4 seconds for the system to update their activity state.

If the user has been sitting or standing for an hour[3] with no interruptions, the User Activity Program will publish a signal causing the buzzer on the Activity Alert System to begin sounding. This is intended to prompt the user to stand up and take a walk.

Once the user has been walking for three minutes with no interruptions, the User Activity Program will publish a signal to turn off the buzzer.

3.2. Posture Detection

Camera App

The camera app component has been developed using Flutter and is a cross-platform mobile application supported by both iOS and Android. It runs on a camera which is positioned to the user's right side, viewing them in the seated position at an adequate distance to include their entire body in frame.

When the user is detected as being in the "Sitting" position, the User Activity Detection program publishes a signal to activate the camera app. The app then takes a snapshot, sends it to the Posture Classification program, and turns off the camera. Upon receiving an acknowledgement from the posture classifier, this process repeats once again.

If the User Activity Detection detects that the user is Standing or Walking, it publishes a signal to deactivate the camera app.

Posture Detection Program

Posture detection is carried out by a Python script on the GCP server. This script receives JPEG images of the seated user from the camera application and passes them to the OpenPose binary.

---

[3] *Time to move*. Human Resources University of Michigan. (2020, March 12). Retrieved November 18, 2022, from https://hr.umich.edu/benefits-wellness/health-well-being/mhealthy/faculty-staff-well-being/physical-activity/time-move#:~:text=Remember%20to%20move%20for%20approximately,helps%20us%20feel%20our%20best

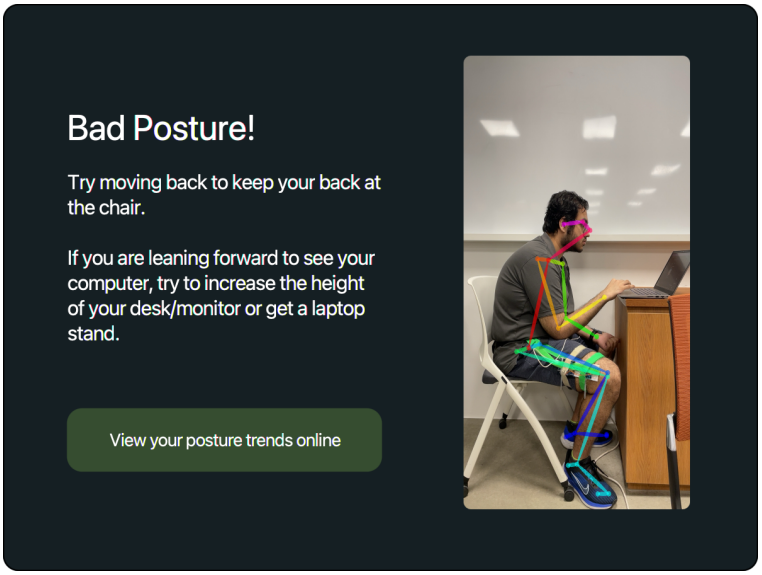
The binary generates a JSON file containing the coordinates of 27 “key points” on the user’s body, as well as a colored graph of these points overlaid on the image of the user. Two of the key points in the JSON file (nose, and tip of the cervical spine) are used to calculate the user’s craniovertebral angle. Based on information from medical literature[4] the user’s posture is classified as poor if their craniovertebral angle is found to fall outside the range of 40 to 65 degrees. In this scenario, the nature of poor posture (high or low angle) and rendered image of the user with key points overlaid are published to the desktop application to be displayed in an alert.

In other scenarios, the script publishes the corresponding acknowledgements (“good posture”, “no human detected” and “math error”) as well. (The “math error” scenario results on rare occasion from undefined behavior, such as the user having their back to the camera, or multiple individuals being in the image, resulting in invalid keypoint generation.) A background thread ensures that if no image is received from the camera application for ten seconds, timeout messages are published. All five categories of publication are received by the camera app, and act as acknowledgement messages, causing the next image to be captured and sent to the cloud program.

The script also stores all craniovertebral angles and classifications, indexed by timestamp, in a file to be read periodically by the web dashboard for long-term analysis generation.

Desktop Pop-up Application

The desktop pop-up is activated each time the OpenPose model detects bad posture. It lets the user know how their posture was by displaying the image that was classified as bad posture, and gives them tips to improve it, depending on whether they were slouching forward or leaning behind. The user can also click on the button in the desktop application to open the web analysis dashboard from the pop-up.



This pop-up app was created using Java and JavaFX, and is activated using Shell and Python scripts.

[4] Choi, K.-H., Cho, M.-U., Park, C.-W., Kim, S.-Y., Kim, M.-J., Hong, B., & Kong, Y.-K. (2020). A comparison study of posture and fatigue of neck according to monitor types (moving and fixed monitor) by using flexion relaxation phenomenon (FRP) and craniovertebral angle (CVA). *International Journal of Environmental Research and Public Health*, *17*(17), 6345. https://doi.org/10.3390/ijerph17176345

3.3. Cloud Server

A Google Cloud Platform server was used to host an MQTT broker employed by all sensors, actuators and applications to communicate with each other. The broker was configured to run on non-default port 1884, with username/password access for 7 devices.

The server also hosted two trained ML models (one for pose analysis of images of the user and one for activity analysis of the user's accelerometer data). In order to enable the former, binaries of OpenPose were built on the server (this required the choice of Ubuntu 16.04 instead of a current release), and the NVidia ML toolkits CUDA and cuDNN were installed on it. For predictability of output, the accelerometer ML model was also trained on the GCP server itself.

SSH private/public key pairs were generated in order to enable the web dashboard to retrieve long-term analysis data from the server using SSH-SFTP, as well to enable team members to SSH into the server securely.

3.4. Long Term Analysis

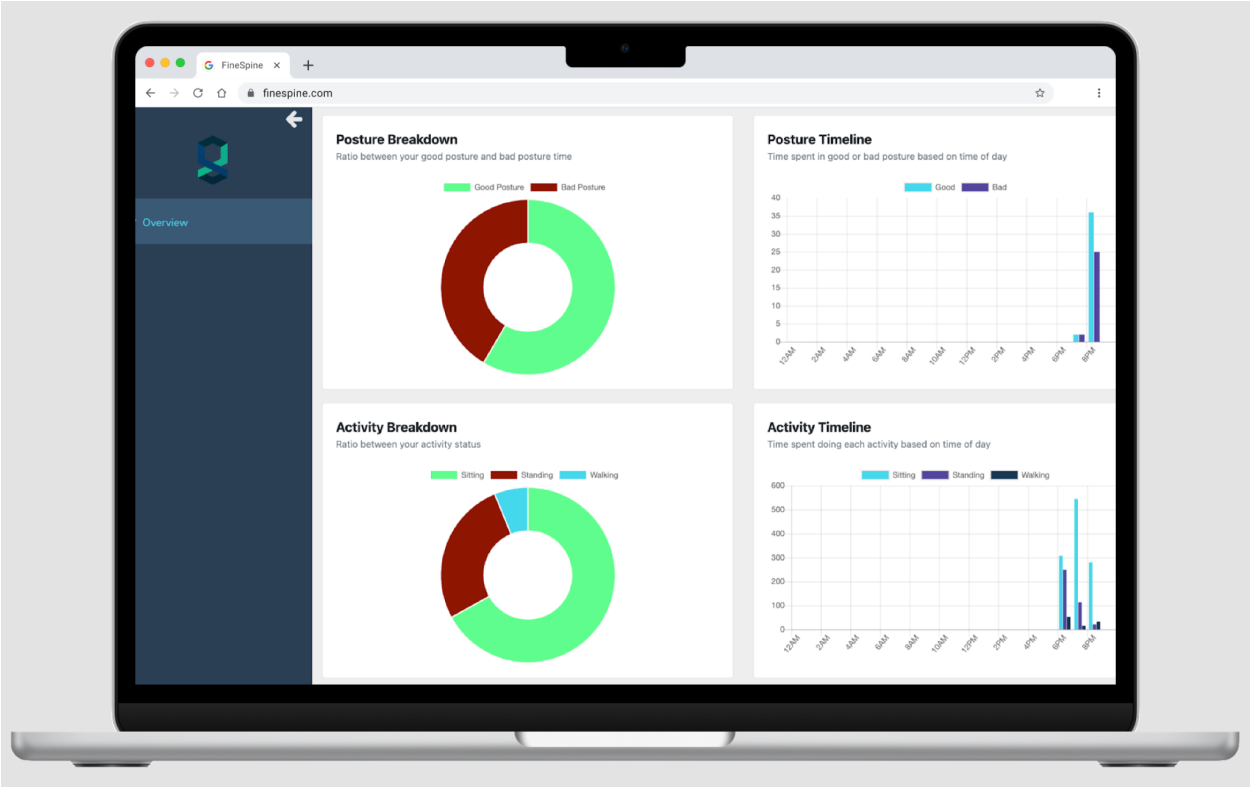


Fig 3.4.1: Web Dashboard

Web Dashboard

A locally hosted web app shows all of the user's health data, including both activity and posture. There are 4 different charts on the webpage:

- *Posture Breakdown:* Ratio of the time spent in good posture and bad posture
- *Posture Timeline:* Hourly posture data across all days
- *Activity Breakdown:* Ratio of the time spent sitting, standing and walking
- *Activity Timeline:* Hourly activity data across all days

All data collected from the user activity detection program and the posture detection model are stored on the GCP server as text files.

On startup or refresh of the React app, these text files are retrieved by a Node.js server and are converted to a .csv file and processed by a Flask server. The processed data is sent back to the React app to be binded to the charts for long-term health data visualization.

4. ML Model:

4.1. User Activity Detection

4.1.1. Data Collection Steps

We were unable to find any accelerometer or gyroscope datasets whose readings were similar (even upon scaling) to those outputted by our IMU. As a result, we created our own dataset.

Fig 4.1.1: Teammate wearing the FineSpine wearable device for data collection

For the data collection process, each member of the team wore the FineSpine wearable and performed the three classified activities (Sitting, Standing and Walking) for about eight minutes each to get a total of 50 000 samples per activity. Each sample contained the x-, y- and z-axis readings for both the gyroscope and accelerometer each, as well as the x-, y- and z-axis angle readings for the accelerometer.

After testing different combinations of data collected from the IMU, we found that using the Accelerometer's x and y-axis angles as features provided the highest classification accuracy for the desired three classes.

The placement of the IMU and wearable device was inspired[5] by the Body Position Sensing project by Angelica Torres, Erik Welsh and Alex Lammers.

4.1.2. Data Preprocessing

IMUAngleStandingNicole.txt - Notepad

File Edit Format View Help

```
Standing,51.89711761,46.78004837
Standing,55.48241425,46.6984024
Standing,52.59968567,46.21615219
Standing,50.24961853,46.42882919
Standing,57.98161697,47.69168472
Standing,57.84337234,47.67090988
Standing,51.45722961,46.8567009
Standing,50.48536682,47.02059937
Standing,52.89431381,46.78515625
Standing,51.51900864,47.09246445
Standing,53.94470596,47.65533066
Standing,52.97554779,46.90786743
Standing,50.62296677,47.15413666
Standing,52.60544586,46.92834854
Standing,54.76684189,47.19529343
```

Fig 4.1.2: Feature Data Sample

A Python program was used to extract the accelerometer's x- and y-axis angles from the sample received from the IMU and aggregate them into batches.

4.1.3. ML Models Used

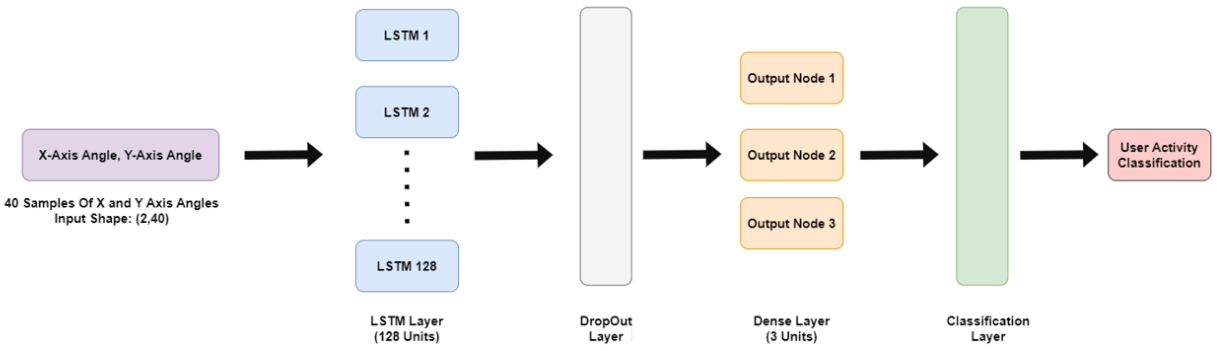


Fig 4.1.3: LSTM Model used

[5] Torres, A. (2021, November 9). *Body position sensing with an accelerometer/gyroscope*. Hackster.io. Retrieved November 18, 2022, from https://www.hackster.io/431666/body-position-sensing-with-an-accelerometer-gyroscope-5139e5

An LSTM, designed as in the diagram above, was trained on the collected dataset. This model was inspired by an existing bidirectional LSTM project created by Venelin Valkov[6] for activity classification. We tweaked this BiLSTM approach to eliminate unneeded classification categories such as Jogging, and reduced the number of dense layers. We switched to a unidirectional LSTM upon finding that it did not result in loss of accuracy.

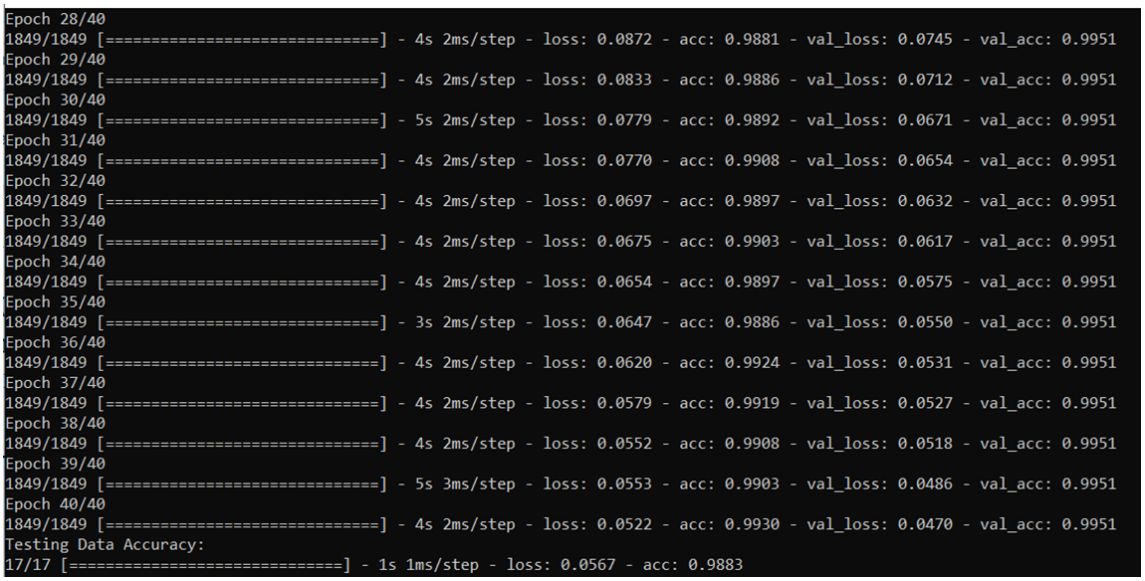


Fig 4.1.4: LSTM Accuracy

By training the LSTM time series model on the custom dataset that we had created, we managed to achieve an accuracy of 99%. Thus, even though we had reduced the complexity of the model, it did not come at the expense of accuracy.

[6] *Time Series classification for human activity recognition with lstms ...* (n.d.). Retrieved November 18, 2022, from https://towardsdatascience.com/time-series-classification-for-human-activity-recognition-with-lstms-using-tensorflow-2-and-keras-b816431afdff

4.2. Posture Detection

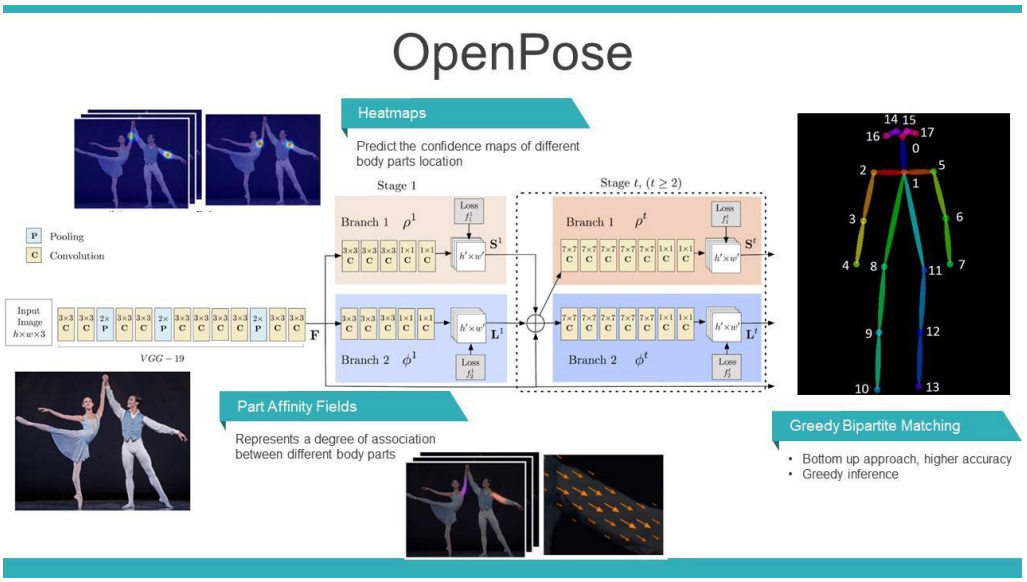


Fig 4.2.1: OpenPose Model

Posture detection is carried out using the pre-trained OpenPose model developed by Carnegie Mellon University. It carries out classification in a three-step process:

- Extraction of feature maps using a baseline CNN
- Generation of Part Affinity Maps and Part Confidence Maps using a multi-stage CNN-based pipeline
- Use of a greedy bipartite algorithm to assign coordinates to each of several “key points” on the body.

5. Power Management

Some of the steps taken to reduce power consumption are:

5.1. Sending IMU Data in Batches

Instead of sending the IMU data to the MQTT server each time a reading is sampled from the IMU, the IMU’s data is buffered and sent in sequences of 40 samples. This reduces bandwidth as well as power consumption.

5.2. Posture Analysis Activated Only When User is Seated

The camera application used to take images of the seated user is activated by the User Activity Detection program only once the user is detected to be in a seated position. If the user is detected to be standing or walking, the application is turned off.

5.3. Side-view Camera is Powered Down Between Image Captures

Upon sending a snapshot of the seated user to the Posture Detection program, the camera application turns off the camera and waits for an acknowledgement from it (or for a timeout from it, in case the sent image was not received due to packet loss) before sending the next one.

6. Power Consumption:

6.1. WeMos Device (Theoretical)

6.1.1. WeMos 1 (Activity Alert System)

WeMos1 receives highly infrequent MQTT messages such as “Inactive” (every hour) or “Flash LED” when the IMU is turned on or off.

WeMos1 transmit MQTT messages such as “IMU Activation Signal On” (only once, at the start of use) or “IMU Activation Signal Off” (at the end of use).

Therefore, the time WeMos1 spends in Active Mode is taken to be negligible, and WeMos1 is assumed to spend all of its time in Modem Sleep mode.

The power consumption for WeMos 1 calculated does not include the power used to play the buzzer or flash the LED.

6.1.2. WeMos 2 (Wearable Device)

WeMos 2 receives MQTT messages such as “IMU Activation Signal On” or “IMU Activation Signal Off” to turn it on or off. WeMos 2 also transmits MQTT messages such as “Flash LED” when it is turned on or off. However, as the time WeMos2 spends in Active Mode for these scenarios is negligible, they are not considered when calculating the total power consumption due to WeMos2 being in Active Mode.

- Time interval between sending two sequences of Acceleration Data to User Activity Detection Program = 2159ms.
- Time taken to send out a sequence of 80 samples of Acceleration Data to the Activity Detection Program = 98ms

Power Consumption of WeMos 2 = Active Mode Power Consumption + Modem Sleep Power Consumption + MPU9250 IMU Power Consumption

$$= \left(\frac{98}{2159} \times 120mA\right) + \left(\frac{2061}{2159} \times 15mA\right) + 450uA = 20.45mA$$

	WeMos 1	WeMos 2	Battery Capacity
StandBy	15 mA (Modem Sleep)	15 mA (Modem Sleep)	10 000 mAh battery
In Use	15mA (Modem Sleep)	20.45 mA	10 000 mAh battery

Estimated Power Life (Continuous Usage): WeMos1 = ~666 hours and WeMos2 = ~488 hours.

6.2. Camera Application

The camera application's power usage was tracked using Android's power-tracking feature. An unoptimized version of the application developed around Check-In 2 was found to use 333 mAh in 13m 46s, for a power consumption of 1456 mA.

The final version of the application (optimized to turn off the phone camera between taking snapshots) was found to use 613 mAh in 26m 21s, for a power consumption of 1395 mA.

7. Challenges Faced

7.1. Differentiating between "Walking" and "Standing".

Initially, we used the gyroscope's x-, y- and z-axis readings to determine the user's activity. However even though this produced a high training and testing accuracy, it performed poorly when tested with the wearable device, being unable to differentiate between standing and walking. Likewise, using the accelerometer's x-, y- and z-axis readings resulted in the same issue.

We ultimately discovered that using the accelerometer's x-, y- and z-axis angles produced stable prediction even when tested with the wearable device. Additionally, it was determined that the change in z-axis angles when the device was in use was so marginal that removing it from the dataset (and therefore reducing packet size for MQTT transmissions) did not have any negative effect on accuracy.

7.2. Camera Power Consumption

An initial version of the camera application would send one image per second to the listener. This cause the used mobile devices to heat up quickly as the camera was in continuous use, sometimes even when no human was in the frame.

To improve upon this, we configured the camera to turn on only when the user is seated, as detected by the User Activity Detection program, and to turn off otherwise. Moreover, the camera turns off completely between taking snapshots, and snapshots are only taken when the previous image has already been classified and an acknowledgement of this is received.

8. Limitations and Possible Improvements

The current setup utilized is somewhat inefficient, as the user is forced to use a dedicated smartphone as part of the FineSpine system. Additionally, if the phone runs background applications, it can be seen that its battery drains fast. A possible improvement would be to use an IP camera connected to a WeMos. The IP camera would then be controlled via the WeMos instead of a camera application. This would reduce the power usage of the camera component significantly.

9. Individual Contributions

Member	**Contribution**
Charisma Kausar	Mobile app to capture and send user snapshots, Desktop app as pop-up notification for bad posture, Desktop Listener to activate notification, Web dashboard for long-term analysis, Servers to retrieve data files and process data
Nicole Joseph	Mobile app to capture and send user snapshots through MQTT, Retrieval of user activity and posture data files from server for the web dashboard.
Pratyush Ghosh	Implementing OpenPose-based algorithm for posture classification. Configuring Google Cloud Platform server, building OpenPose binaries, and configuring MQTT broker. Helping improve classification accuracy for user activity model.
Siew Yang Zhi	Wearable, User Activity Detection Program, WeMos1 and WeMos2 Arduino Programs, User Activity Detection ML Model

10. Citations

1. Hidalgo, G., Cao. Z., Simon, T., Shih-En, W., Raaj, Y., Joo, H., Sheikh Y. (2017) *CMU-Perceptual-Computing-Lab/openpose: OpenPose: Real-time multi-person keypoint detection library for body, face, hands, and foot estimation*. GitHub. Retrieved November 18, 2022, from https://github.com/CMU-Perceptual-Computing-Lab/openpose

2. Khusainov, R., Azzi, D., Achumba, I., & Bersch, S. (2013). Real-time human ambulation, activity, and physiological monitoring: Taxonomy of issues, techniques, applications, challenges and limitations. *Sensors*, *13*(10), 12852–12902. https://doi.org/10.3390/s131012852

3. *Time to move*. Human Resources University of Michigan. (2020, March 12). Retrieved November 18, 2022, from https://hr.umich.edu/benefits-wellness/health-well-being/mhealthy/faculty-staff-well-being/physical-activity/time-move#:~:text=Remember%20to%20move%20for%20approximately,helps%20us%20feel%20our%20best

4. Choi, K.-H., Cho, M.-U., Park, C.-W., Kim, S.-Y., Kim, M.-J., Hong, B., & Kong, Y.-K. (2020). A comparison study of posture and fatigue of neck according to monitor types (moving and fixed monitor) by using flexion relaxation phenomenon (FRP) and craniovertebral angle (CVA). *International Journal of Environmental Research and Public Health*, *17*(17), 6345. https://doi.org/10.3390/ijerph17176345

5. Torres, A. (2021, November 9). *Body position sensing with an accelerometer/gyroscope*. Hackster.io. Retrieved November 18, 2022, from https://www.hackster.io/431666/body-position-sensing-with-an-accelerometer-gyroscope-5139e5

6.Valkov, V (2020, April 25). *Time Series classification for human activity recognition with lstms ...*. Retrieved November 18, 2022, from https://towardsdatascience.com/time-series-classification-for-human-activity-recognition-with-lstms-using-tensorflow-2-and-keras-b816431afdff

Back to Top