Product: Vitruvian
Team: Vee

Abstract

Vitruvian aims to help improve people's posture and health while working from home. Students like us and 24 million other Brits are working at home due to the COVID-19 pandemic. Working from home is having a huge impact on our spine's health and many do not have the recommended desk, ergonomic chair, monitor, mouse and keyboard for home working.

Vitruvian is a compact and lightweight device that tracks your posture and activity, provides real-time biofeedback and sends insights to the accompanying Android App. The device is lightweight and comfortable and attaches to your back with the included vest.

1. Project Plan Update

TASK NAME	STATUS
FINALISE HARDWARE	ACHIEVED
TESTING OF SLOUCH DETECTION	ACHIEVED
FINALISE AND INTEGRATE SOFTWARE	ACHIEVED
WRITE USER GUIDE	ACHIEVED
CREATE WEBSITE	ACHIEVED
CREATE MARKETING VIDEO	ACHIEVED

Table 1. Planned goals for up to 4th Demo

We have spent the last two weeks finalising our product, integrating code modules, and creating marketing items such as the website and user guide. We have successfully met all major goals set for this demo and thus have achieved all adjusted goals for the project.

This demo report will serve as both a summary of the work we have done since last demo, as well as a final reflection on the state of the project and product.

Collaborative work periods were held daily over over the this week, and on Monday, Wednesday, and Friday on the previous week, totalling 19 hours. This allowed us to carry out code integration and write-ups quickly and effectively. We also feel that working together in this way has improved the quality of our reports, website, and user guide.

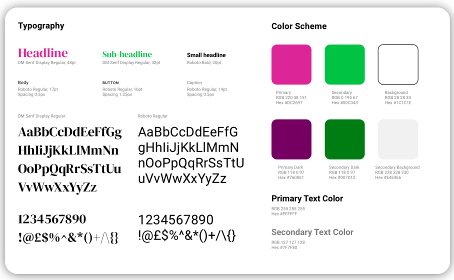
Aside from this, we also carried out individual work. Assignment of group members to tasks was:

- **Alasdair**: Integrating code together (11hrs). Writing for the guides and video scripts (6hrs). Planning evaluation for the slouch detection (4hrs).
- **Andrew**: Adding hardware functionality for the grove led button and grove buzzer (10hrs), working on evaluation tests in webots (12hrs).
- **Mohamad**: Creating the high level sedentary detection algorithm that utilises the machine learning, connecting it to the model as well as running unit tests on the algorithm (16hrs).
- **Vincent**: Created new environment of simulation (7hrs); Added LEDs and changed textures for the sensors to provide better readability (5hrs); Added randomness to the simulation-based tests (6hrs).
- **Anelise**: Created user-guide (3hrs); finished Android Application, changed calibration screens, added graphs for the results and new settings (25hrs).
- **Morgan**: Creation of app graphics (8hrs), additional documentation (3hrs), thorough accessibility research and assessment (4hrs).
- **Jake**: Created Live iOS to Pi "mock" communication (5hrs); Created website (20hrs); Created video (25hrs); Created graphics for reports (10hrs)
- **Yining**: Created sketch and model.(15hrs) Send dwg file to technician for printing. Finished further improvements on the model.(5hrs)

2. Marketing Materials

2.1. Style Guide

In the first week of the project, we created a style guide for our visual designs. We have made sure to implement it for the marketing materials we created during this demo in order to promote brand cohesion, and because it was chosen to improve visibility and colour-blind accessibility.



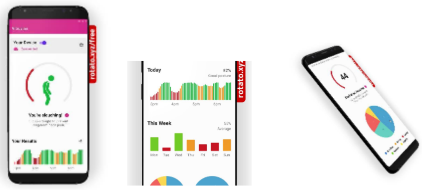
2.2. User Guide

Our User Guide was created using canva.com.We stuck to our style guide throughout its design.

2.3. Website

Our website has been built since the last demo. Jake has experience in web development, so we opted not to follow the provided template and create our own, keeping in line with the branding we created in Week 1. The website and API was built using NextJS for React and hosted using Vercel. The website also uses FontAwesome library and Icons8 for icons; and `react-reveal` and `react-scrollmagic` libraries for animations. We also added a comment functionality using Disqus.

Animations were created with Rotato. The image of the phone moves between the following positions as you scroll.



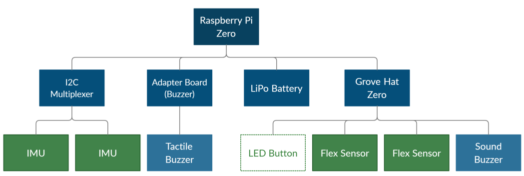
3. Technical details

3.1. Simulation

The background of the simulation has been changed to an office, which can simulate a more realistic operating environment of the device. We have also updated the readability of the information in the model. The sensor on the back now can light up when the system detects slouching.
In simulation-based testing, we have added some randomness to the model. The program can now simulate some random actions while performing the main one.

3.2. Hardware

We incorporated the button for calibration and the sound buzzer for extra accessibility as we intended. This leaves our final design as below, where blue boxes are within the central casing, green are external, and the button is built into the casing itself. While we have a model design created, it was not feasible to actually print it with the technician time we had. Flex senors were included for the purposes of future experimentation.



3.3. Communication

Since last demo, we have finalised our web server based communication mockup. It now supports live communication between the iOS sensor app (which we use to record data in lieu of having hardware access) and our detection code, meaning we demonstrate our system outside of simulation.

Other communication details remain unchanged since our last report, as we were able to integrate the communication module into the detection loop without issue. The web server is hosted on Vercel, and uses Firebase for the database system. The web server also handles communication between the Pi and the Android using our custom buffer file format, discussed in the last demo.

3.4. Android App

The calibration screens have been changed to include more useful information and a video showing the required back positions. The button on the home page showing the connection with the web server was replaced by an overview of the current day results. Instead of having a test ID for the mock connection we now have a device ID that is meant to identify the physical device. The features in "Configure alerts" such as disabling the sound and the vibration when slouching are for now hardcoded to showcase the future functionalities because we do not have a Bluetooth connection in place.

Data is passed to the app from the device only when the app is active and connects. Data is stored in an extremely compact form on the device's internal storage, so will not be lost if the system loses power.

3.5. Usability and Acessibility

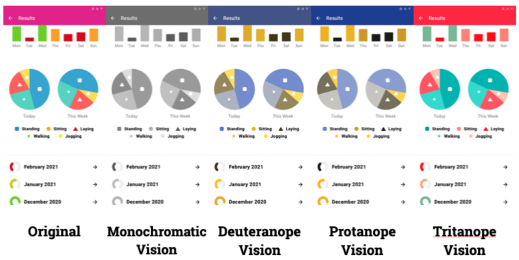
Social inclusion was one of our top priorities when building the app, hence we have added the "Change Character" screen that lets the user choose the character they identify most with. We had a review of our designs with Ryan Bowler, who suggested we make several changes to improve the emotional connection between the user and the product.

We realise that people may want to turn off the device's sound at work, or the vibrations may get annoying at times. Therefore, we wanted allow the user to configure their alerts with options: Visual notifications, Haptic notifications, via a vibration motor, Audible notifications, via a buzzer.

The design guidelines we have followed are: WCAG AAA compliant for readability. Using Material Design best practices. This creates Android-esque styling, ensuring the design is naturally familiar to the user. We have also made sure our system is colour deficiency and colour-blind friendly, with several options.

Following these guidelines has improved out usability have made our website accessible to those with disabilities. We made sure to use a colourblind-friendly design. We use to Aria-label to provide a screen-reader functionality, so our website is usable by those with visual disabilities. We also added alt-tags for all our images.

The app has been fully tested against 4 different types of visual deficiency, with particular focus on data visualisation. As seen below:



3.6. Software Implementation

Slouch Detection & Notification Since last demo we have performed finishing touches to the slouch detection system, and integrated it with the rest of the code for the system including that for sedentary detection, button inputs for calibration, and wireless connectivity to the app. What follows is a full summary of the system's functionality.

Firstly, the system calculates the angle relative to gravity of both accelerometers placed on the back. We use this to calculate average back angle, as well as overall spinal curvature of the user.

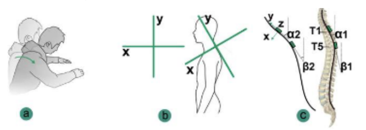


Figure 2. Angle Calculations. (a) Compensation Movement. (b) Zero Degree Calibration. (c) Calibration Model

(Via (Wang et al., 2015))

We can calibrate normal and threshold values. When activated by a button, the system gathers a set of readings over a 4 second period, and averages these to for the normal. We added a grove LED button to our physical system, which activates our calibration procedure.

We expect most users not to need to calculate threshold values, however this feature has been included for users who have different posture needs - we wanted our product to be accessible to everyone!

Detection of poor posture is done by comparing the absolute difference between current back angle and posture to the calibrated normals. We use the "and" variant of our algorithm (where both angle and posture need to deviate significantly from the expected normal). We were worried that with the or algorithm (where only one of the two needs to be over) someone who is naturally moving around to keep comfortable might accidentally trigger the system.

To prevent the system from alerting the user every time they bend over to pick something up, we only alerts the user if they have been slouching for 4 seconds. We implemented a counter system that ticks up when the user's posture is poor, and ticks back down otherwise. The system ticks down slower than it ticks up (i.e it increments the counter variable less), to so someone repeatedly entering poor posture will still be notified.

Notification is handled using a haptic motor, a noisemaker, and app notifications.

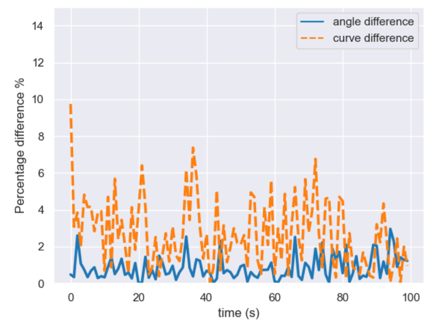
Sedentary Detection We have also integrated our sedentary detection model into the rest of our code. This means that we can run the full system as-intended in the real world, using the iOS mock-input tool.

On a high-level Sedentary Detection utilises our machine learning model to alert the user if they were sitting, standing or laying in a sedentary behaviour for longer than an hour. It then requires the user to stand up and walk or exercise for at least one minute. This is done via a counter that increments every 4 seconds when it detects sedentary behaviour and once it reaches the time limit (1 hour) it will activate the buzzer and alert the user. Another counter will be incremented for the activity time and once it reaches the time limit (1 minute) it will reset the sedentary counter.

4. Evaluation

Slouch System We carried out testing in webots to confirm the effectiveness of our system. We added accelerometer noise in simulation based on the datasheet for the accelerometers used on our physical mode. The datasheet (InvenSense, 2016) reports a maximum root mean square error of 8 milligravities.

Below is a graph (generated within our simulation) of the errors in our system (resulting from our introduced noise) when our model is relatively still, with only small movements to simulate a real human.

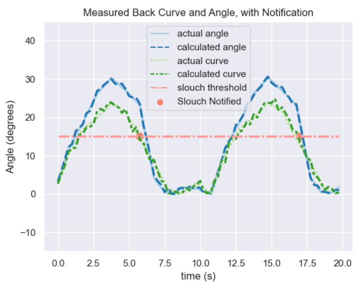


Angle error is almost always <5%, and curve error is almost always <10%. Either of these values being below 5% for any given reading is sufficient to ensure high accuracy, as the use of our "and" checker means that both values need to erroneously be over threshold to cause a 'bad' slouch detection.

Due to the use of accelerometers, the system suffers reading spikes during quick random movement but this is not an issue due to our counter system (i.e it won't register slouching during these spikes). The user will still be alerted if long-term slouching occurs, which was our original goal.

We ran tests with a 6 or more second slouch, this is a typical use-case, and over 40 tests we had success in every instance. After a short period of slouching, the system notifies the user, as expected. THis is the ideal result for our system.

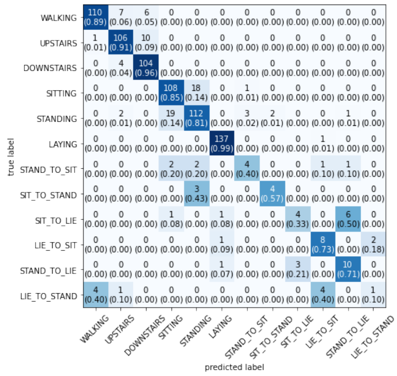
We also ran some tests on a specific edge case scenario. With noise and random movement included, the webots model slouched for just over 4 seconds (the length of our counter), returns to normal, then performs the same movement again. We expect that just before the end of each slouch period we will detect a slouch. The purpose of this test is to check that noise and random movements will not overly disturb accuracy, even on slouch periods just long enough to count. Our results are graphed like this, with detections marked on the threshold line.



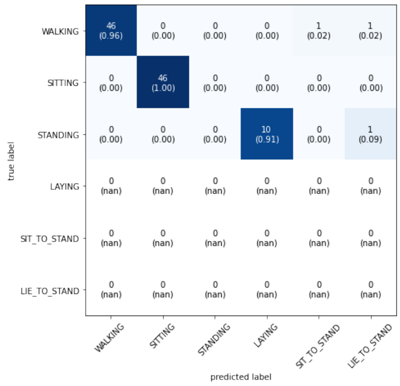
We consider a success to be any detection after 4 seconds of slouching, and withing 0.5 seconds of slouching ending (to account for potential noise distortion). A partial success is when only one slouch is correctly detected. A failure is if any detections are outside the acceptable periods (misdetection), or both slouches go undetected (no detection).

Even under this edge case scenario our system functions acceptably, with partial success in 38% of tests an and full success in 34% of our 50 tests. No misdetections occurred, which we are very happy with as preventing user annoyance due to notifications when not slouching is a priority.

Sedentary Detection Our first round of evaluation was to test our machine learning model on the test data from the dataset we trained on. We achieved high accuracy.



The next step was to test using our own data, generated using the IPhone app. The purpose of this was to confirm that accelerometer placement on the body was not going to cause issues, as our device has accelerometers in different locations to the system that generated the original dataset. Again we achieved high accuracy, telling us that our system will work regardless.



Future Development We have a full product with demonstrated slouch and sedentary detection. If given further time to develop the product, we would further develop our tracking algorithms to accurately track more activities for different body-types.
We would also build out the features of the Android application to provide better insights to the user, and build an equivalent iOS application.
While we believe the vest is a comfortable method to attach Vitruvian to your body, we believe we can design a solution which is even less obtrusive and easy to quickly take on and off.

5. Budget

A. Estimated technician time usage

Week	Technician Time (Hours)
Week 3 (25 - 31 Jan)	1
Week 4 (1 - 7 Feb)	2
Week 5 (8 - 14 Feb)	3
Reading Week	
Week 6 (22 - 28 Feb)	3
Week 7 (1st Mar - 7th Mar)	3
Week 8 (8th Mar - 14th Mar)	3
Week 9 (15th Mar - 21st Mar)	3
Week 10 (22nd Mar - 28th Mar)	1
Total	19 / 24 hours

B. Estimated cost for components

Item	Cost
3D printing (TPU)	~ £35
Vest with pouch	~ £25
Raspberry Pi Zero W	£9.30
2x MPU-9250	£11.40
Adafruit TCA9548A I2C multiplexer	£5
Seeed Studio mini vibration motor	£1.09
Adafruit 2305 controller board	£7.28
LiPo Battery	£17.88
Charger	£5.66
LED Button	£1.80
Sound buzzer	£1.40
Grove Base Hat for Raspberry Pi Zero	£6.40
Total	~ £130

6. Demo Video

Watch here.

Appendix

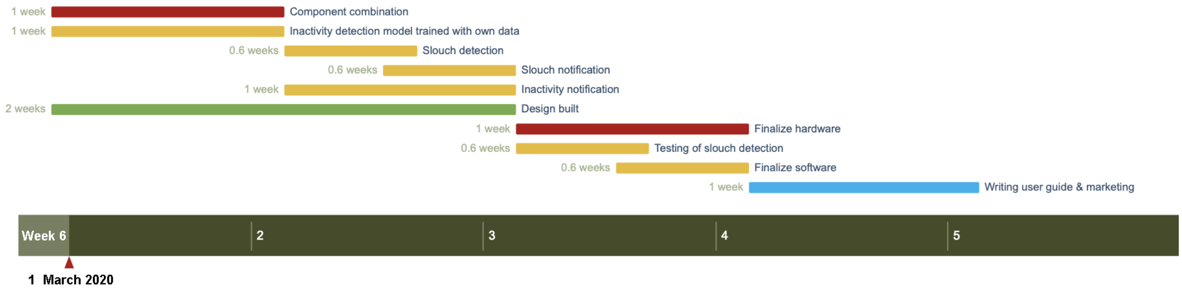
References

InvenSense. Mpu-9250 datasheet, 2016. URL https://invensense.tdk.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf.

UCI. Human activity recognition using smartphones data set, 2015. URL http://archive.ics.uci.edu/ml/datasets/Smartphone-Based+Recognition+of+Human+Activities+and+Postural+Transitions.

Wang, Qi, Chen, Wei, Timmermans, Annick AA, Karachristos, Christoforos, Martens, Jean-Bernard, and Markopoulos, Panos. Smart rehabilitation garment for posture monitoring. In *2015 37th annual International Conference of the IEEE engineering in medicine and biology Society (EmbC)*, pp. 5736–5739. IEEE, 2015.

A. Gantt Chart



B. Labels and Data count from (UCI, 2015)

Label	Samples Count	Percentage
STANDING	138105	16.9%
LAYING	136865	16.8%
SITTING	126677	15.5%
WALKING	122091	14.9%
WALKING_UPSTAIRS	116707	14.3%
WALKING_DOWNSTAIRS	107961	13.2%
STAND_TO_LIE	14418	1.8%
SIT_TO_LIE	12428	1.5%
LIE_TO_SIT	11150	1.4%
LIE_TO_STAND	10867	1.3%
STAND_TO_SIT	10316	1.3%
SIT_TO_STAND	8029	0.9%