Product: Vitruvian
Team: Vee

Abstract

Our project aims to help improve people's posture and health while working from home by monitoring sedentary activity and posture, notifying them via an app or by tactile feedback on a worn device.

Since last demo, we have achieved functionality on all key areas of the system (slouch detection, sedentary activity detection, hardware communication, simulation, testing and the Android app). We built tools to work around previous bottlenecks and have been able to make good progress without similar setbacks. We are able to verify our functionality in all areas. Going forward, we will continue to combine our systems together in real life and in simulation.

1. Project Plan Update

Task Name	Status
Combine all components	Achieved
Inactivity model trained w/ own data	Not necessary
Slouch detection	Achieved
Sedentary detection	Achieved
Slouch notification	Achieved
Inactivity notification	Party achieved
Casing built	Not feasible

Table 1. Planned goals for up to 2nd Demo

Following our updated Gantt chart from last demo (see Appendix C), we are on track with all major goals.

Feedback from the last demo highlighted our need to use alternate avenues to achieve progress, as we were encountering difficulty with the virtual setup of SDP. Following this advice, we have fully simulated our system in Webots and are able to run tests. We have also created a mock connection to demonstrate and test hardware-app communication using . These tools allow us to make progress on all parts of the system without the need for the technicians.

After receiving advice from the technicians, we have decided not to go ahead with building a physical casing for our system due to safety restrictions. We weren't able to connect the inactivity detection system to our buzzer-based notification system as we were unable run TensorFlow on the Pi. However, we were able to demonstrate this in simulation.

We are continuing to handle work organisation at our weekly meeting. The assignment of group members to tasks was:

- **Alasdair**: Finalising the implementation of slouch detection and notification (5 hours). Testing the hardware/software systems (15 hours)
- **Andrew**: Working to ensure the functionality required by the software exists in both hardware (28hr) and simulation (9hr).
- **Mohamad**: Implementation of sedentary detection and evaluating findings for the ML model (15 hrs). Generating and labelling real test data (5hrs). Running sedentary tests on model after training with simulation and phone data (10hrs). Deploy model to TensorFlow Lite and run real-time classification tests (5hrs).
- **Vincent**: Adding greater movement complexity to the simulation model and incorporating the necessary accelerometers into the code. (20hr)
- **Anelise**: Developed the Android Application - connection with the Web server, design, charts and settings (25 hrs).
- **Morgan**: Study of better demo video software and creation of demo video (10hrs), met with Ryan Bowler to discuss social inclusion (2hrs), further accessibility study, accessibility feature planning, and consultations with potential users with accessibility needs (4 hrs).
- **Jake**: Built web server for mock connection (10hrs), and Python code to connect to web server from Pi (2hrs). Met with Ryan Bowler to discuss social inclusion (2hrs). Created slide graphics for demo video. (2hrs)
- **Yining**: Learning how to use AutoCAD (23 hours) and created a basic 3D model case using it (3 hours). Spoke to Garry Ellard about ordering sportswear and further design details.

2. Technical details

2.1. Simulation

We have increased the complexity of movement in our human body model and programmed a pair of accelerometers into our simulation to act in the same manner as on our physical device. This has allowed us to create slouch and activity detection tests.

Some "creative" implementation was required to emulate our physical setup for slouch detection. The lack of buzzer means slouch detection is simply printed to the terminal. The two accelerometers also required emitters and receivers in order to communicate their readings. In addition, the timing of the readings had to account for the functionality of the simulation's time-step. Aside from these interfacing changes, the core slouch detection algorithm remains the same and functions as it would for a real person.

With physics enabled in the simulation, we can realistically mimic the behaviour of our device and carry out both simulated and physical tests in-tandem. In-tandem testing means that we can both prove the real-world functionality of our system, and carry out more complex tests that would be difficult to set up via the technicians.

2.2. Hardware

We have made a substantial change in our approach to hardware since last demo. In the past 2 weeks we've held 3 separate calls with Garry Ellard, and communicated with lab techs near daily in the labs. This has enabled us to keep on top of what is feasible to be accomplished within the labs and what we need to work on outside of them. We and Garry are confident we can have a fully functional version of the device built and tested in labs, if not the full design built and worn (due to safety concerns). Regarding the physical setup, we've made the following progress:

Multiplexer

Shortly after last demo we got a multiplexer board hooked up to our pi. This allows us to read from 2 IMUs on the same pi. The multiplexer claims a single I2C address, but by writing formatted bytes to it, we can select which device connected to it we are currently reading from. This functions quickly enough to not noticeably impact on our read times. We can read from both IMUs in what is, for our purposes, essentially simultaneously.

Battery Discussions

The technicians have acquired a number of lightweight 3.7V rechargeable LiPo batteries at both 1500mAh and 2000mAh with some appropriate chargers. Although this will require some testing with our device to ensure they are both suitable and safe, on paper these should work perfectly. We're confident we can get these hooked up and working when we make the final move to miniaturise our device.

Grove hat

We have also included a Grove Hat for the raspberry pi zero in our design. This allows the flex sensors to be hooked up (as they are now), since they require analogue pins to be read. The additional pins provided by the Grove also allow us to circumvent the need for designing additional PCB hardware. This may slightly increase our size profile, but it should decrease costs and maintain function.

Further Features

Discussion has also been held regarding further potential hardware components, such as regarding communication with the app (a button to establish connection), and accessibility (a sound feedback device as well). These are covered more in other sections but based on communication with Garry should be feasible given our inclusion of the Grove Hat.

2.3. Communication

Due to constraints with sourcing the necessary components, we haven't been able to build a Bluetooth connection yet. However, to enable communication between different components of our system, we built a web server to send data using an API. The web server uses **Vercel** to deploy a serverless API and **Firebase's Realtime Database** for persistence. The code for a Bluetooth connection and an HTTP connection can be swapped interchangeably on both Pi and Android, while the remaining detection algorithms remain the same.

The web server contains endpoints to fetch accelerometer data and Pi to Android communication data in a custom format. The web server is hosted at vitruvian.jakeryan.co.uk. Both accelerometer and communication data are generated using either mathematical calculations or using pre-generated data.

The accelerometer endpoint is at `/api/pi/[mpu_id]` where the id can either be the name of a predefined test (e.g. `test1_mpu1` and `test2_mpu2` in the demo video) or the name of a file generated using our iOS accelerometer tool (see Generating Data).

Our Pi to Android communication is available at `/api/app/[id]` and uses a custom format. It is a list of colon separated event in the format `[ACTIVITY_TYPE] [timestamp] ([value])`. The activity type is a number from a predefined enum (see Appendix ?), timestamp is a UNIX timestamp in milliseconds and value is an optional score parameter only where activity type is equal to 0 (`POSTURE` event). The `id` parameter can be selected in the app to change connection.

2.4. Android App

Great progress has been achieved with the Android application. We have integrated our RESTful Web API to communicate with the Pi. In the Device Settings screen, the user can enter a `test_id` which represents a mock connection to the Pi through our web server. We are displaying the results in a user-friendly way mking use of the `MPAndroidChart` and `CalendarView` libraries from GitHub. Results are stored so that daily results are recorded and saved for the past 30 days and an average score is computed instead for the months before that as data becomes older.

2.5. Generating Data

We used our iOS accelerometer tool (shown in the last demo) to generate a large body of test data for use in accelerometer testing. Group members used their phones to generate data in their own time without going through the technicians. This body of test data proved immensely useful for demonstrating the proper function of our inactivity detector.

CSV files from the tool can be placed in the **ios_tests** folder of our hardware repository and will automatically be available at the endpoint `/api/pi/[file_name]` where the file name matches the one in GitHub.

2.6. Design

Yining spoke to Garry about how the construction of the robot might affect the design of the case. From the discussion, the further implementation and improvement of the design would depend on detailed information about the adding of bump and IR sensors as well as the way all the cabling are routed.

We have also asked the technician team to order the "manbra" component from PlayerLayer and are awaiting delivery.

However, we have decided not to go ahead with building a physical casing for our system, as the technicians would not actually be able to wear it due to COVID safety restrictions - they are not able to pass around the clothing it would be attached to and put it on.

Morgan and Jake had a meeting with Ryan Bowler to discuss the design of our product and app with a focus on social inclusion. We discussed a number of potential tasks to increase inclusivity accessibility of our product including adapting the products to support visual/audio/motor disabilities, methods for behavioural training to help build a relationship with the product, detailed visual cues to enhance alert feedback and other design ideas to promote social inclusion through product features. Some key features on our roadmap are: - High Contrast and Dark Theme options, for those with visual impairments, or those using the system while working late at night, to reduce eye strain.

- Font size options and dyslexia-friendly fonts.
- Ensuring all text is either screen-reader compatible or has a recorded voice-over reading.
- We will also work to ensure the app is WCAG 2.1 compliant.

Implementing these changes will be a big focus towards the next demo.

2.7. Software Implementation

Slouch Detection & Notification We now have a functional slouch detection system, and have demonstrated it working alongside the buzzer to alert the users of slouching, both in Webots and in real life with the technicians.

The algorithm implemented was mostly similar to our system state diagram from Demo 1. We first perform calibration, where default values for back angle and back curve are set. The code then enters a loop (one iteration every 0.25s), and on each loop we use the sensors to detect slouching as described in Demo 2. If slouching is detected, a counter variable increases, and if no slouching is detected, the counter decreases (to a minimum of zero). If this counter indicates that the user has been bent over for at least 4s, then we decide that slouching has occurred and notify the user. We refer to this general system as our "loop and count" system.

One of the key elements of this loop and count system is that the counter decrements slower than it increments. Our system accounts if the user is repeatedly entering and exiting slouching, which would normally fail to classify as bad posture due to the speed of the counter.

Notification is currently handled via a buzzer attached to the Pi. We bypassed our previous library issues for the buzzer (accelerometer only works on Python 2, whereas the buzzer only works on Python 3) by simply executing a Python 3 program to activate the buzzer from our Python 2 detection code. As mentioned in the accessibility section, we are planning to add sound-based notification to our system as well. The technicians have indicated that we will probably be able to add this to our real life model, but as it is a last-minute issue we are also looking at adding it to Webots.

Posture Scoring The most significant deviation from the original system state diagram was due to changes in our method for hardware-app communication. Instead of communicating with the app every time slouching is detected, the software maintains a buffer file containing posture scores over a specified time period (e.g score between 13:00 and 13:01 if we set minute-long score periods).

In order to do this, our Python code keeps track of how often the user is in bad posture over a time period. Whenever it reaches the end of a period, it records the score value for that time period to a text file (the buffer file) stored on the Pi. When the Android connects to the device and asks for updated data, the Pi sends this buffer file and resets the buffer file to empty.

Sedentary Detection Following on from our previous Demo, we chose to continue with CNN model rather than LSTM because it had a better initial accuracy as far as our previous Demo findings were concerned. We have trained our model on the dataset from (UCI) because the phones used to collect accelerometer data were placed at the waist level which matches where we want to place ours. Unlike with the (Kwapisz et al., 2011) dataset where they place phones in the front pocket. UCI dataset contains around 4.5 hours of accelerometer and gyroscope data captured at a rate 50Hz. With that being said, the dataset contains

815,614 samples with 6 features: 3 features for accelerometer (x,y,z) and 3 features for gyroscope. We only used accelerometer features when training our model. The list of the labels and corresponding samples number that was found in the dataset is here (F)

We noticed that the dataset is quite balanced when we look at the first 6 labels. However, there are very few samples for the last 6 labels which may result in poor classification for those labels.

Before we gave this data to our model, we first normalised the data and segmented it into time-frames of 4 seconds each with 50% overlapping, such that each sample has a shape of (200,3). This will help our model in detecting consistent patterns in 4s time-frames. We split our data to 80% training set, 10% validation set and 10% testing set. We then trained our model on the training set, deployed and converted to TenseorFlow Lite format to be compatible with TensorFlow Lite interpreter on the Pi. Please see Evaluation for our findings.

3. Evaluation

Physical testing We firstly carried out basic real-life testing of our system with the technicians during lab time (a recording of one of these tests can be found in the Demo video). While the outcomes of these tests is not as easy to precisely analyse as Webots testing (as it is difficult to attach exact outputs to the console to the same point in a provided video to prove when detection takes place), it still serves as a useful demonstration. We can, at least, show that some detection occurs and the buzzer goes off when slouching is done.

Physical testing also allows us to get a better idea of how the angle and curve of a human back changes during slouching. This means we can further refine our webots tests, and narrow down the exact values we should be looking for when carrying out slouch detection

Webots slouch testing As previously mentioned, we carried out testing in webots as well as with the technicians. Doing testing in webots allowed us to more precisely prescribe the movements of the person (or simulated person, in this case) wearing the system, as well as to read the system output at stablet pecific time periods during the test. We've included a table I of 49 tests, of varying movements and detection thresholds, and recorded their outputs for both our individual detection of a curves and angles, aswell as our relevant outputs from our AND, and OR algorithms.

As can be seen the OR algorithm is substantially more sensitive than the AND. These both have advantages depending on how small a slouch we wish to detect. In addition we can see that even small differences in threshold can have large impacts on results. This reinforces that we need to include calibration per-individual user for what they consider "poor" posture. We also require testing on humans to establish how "hard" our threshold should be for comfort. But the main take-away is that our methodology certainly

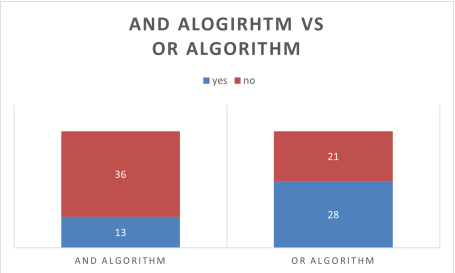


Figure 1. Results on simulation tests of each detection algorithm

works, and hopefully curve detection will become more accurate once flex sensors are included.

One of our tests confirmed the efficacy of our "loop and count" system. We had the model rock back and forth out of slouching, and observed that the system was still able to pick up slouching.

Inactivity detection testing After we trained our model. we tested it on the 10% testing set and we had an accuracy of 87% and this confusion matrix (D). From the confusion matrix we see that our model recognises Laying with 99% accuracy, walking with 89%, Walking upstairs with 91% walking downstairs with 96%. although that the accuracies for detecting Sitting and Standing are 85% and 81% respectively, we can see that there is about 14% miss-classification rate between them. and that makes sense since accelerometer data will be very similar when sitting and when standing still. for the rest of the classes the accuracy ranges between 10% and 70% since they only had 1% of the total dataset size each. having more data will help increasing those. For sedentary detection to work, we only need to map Sitting, Standing and Laying to Sedentary and the rest of the activities will be mapped to Dynamic. We had 3 testing phases for our algorithm. In the first two phases we only tested the final mapped results (black box testing style). In the third testing phase however, we evaluated the classification results of our model. As we showed in our video we extracted data from accelerometers placed on the back of simulated seated models in Webots and have achieved a fantastic accuracy of 100% for sedentary (more details in our video). In the second testing phase, did the same test on real data we have generated from accelerometers on phones that we sticked to our backs (more details about the testing data in the video) and also achieved a 100%. these two tests allow us to confidently say that our algorithm works perfectly for detecting sedentary behaviour. In our final testing phase we approach our model we looked at the CNN model and evaluated its results. The test set we generated and labelled included 2 to 3 minutes for each of the 3 labels, Walking, Sitting and standing. And this is the resulted confusion matrix (E). From looking at the accuracies on our matrix we can see that it classified Walking with 96% accuracy, Sitting with 100% accuracy. However, standing was almost completely classified as Laying which was unexpected, mainly because Standing had a great accuracy on the test-set from (UCI). To solve this, we're planning to add more features

to our model to make it more accurate in distinguishing between sedentary activities and we are also planning to train or model on Jogging. so we can potentially add more human activity features to our device.

4. Budget

Appendix B shows a current estimated total cost to make our product.

Appendix A shows an estimated breakdown of how many technician hours have been used up to this point. It is difficult to quantify our technician time usage due to the format of the labs.

5. Demo Video

Watch here.

Appendix

References

Kwapisz, Jennifer R, Weiss, Gary M, and Moore, Samuel A. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12(2):74–82, 2011.

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

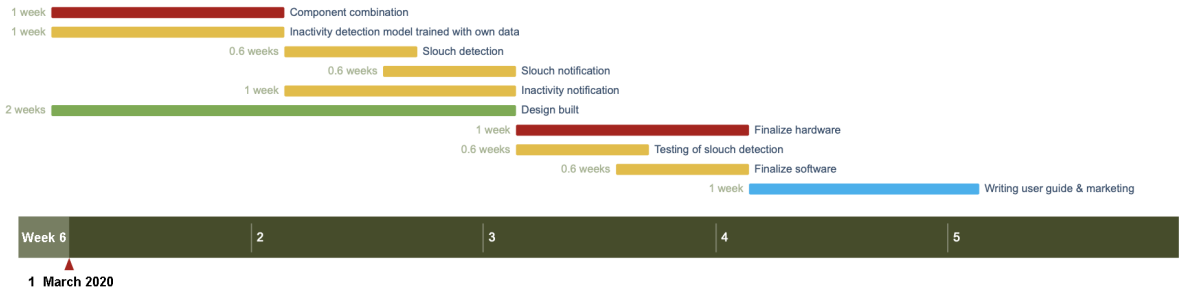
A. Estimated technician time usage

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 (29 Feb - 4 Mar)	3
Total	12 / 15 hours

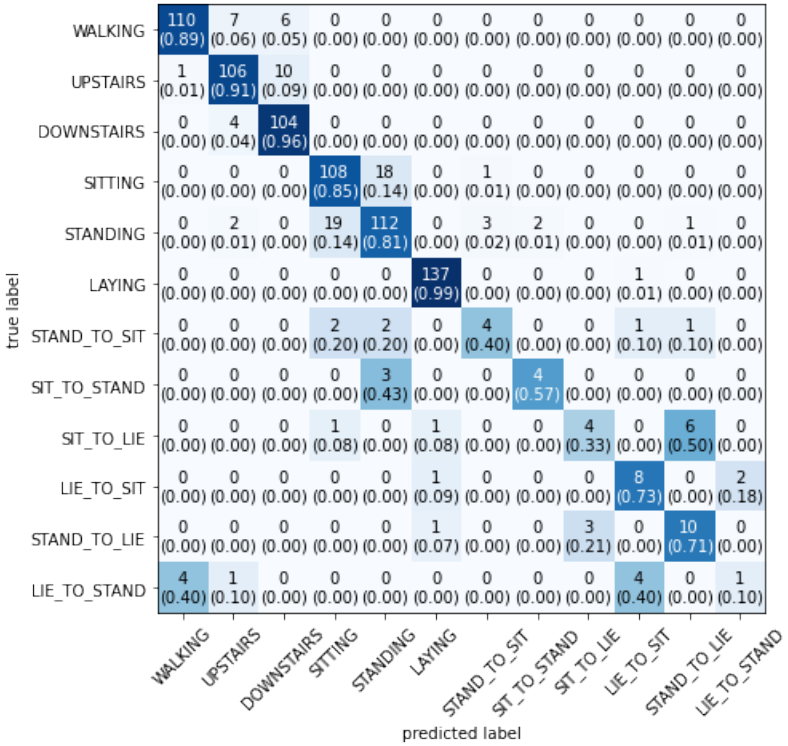
B. Estimated cost for components

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

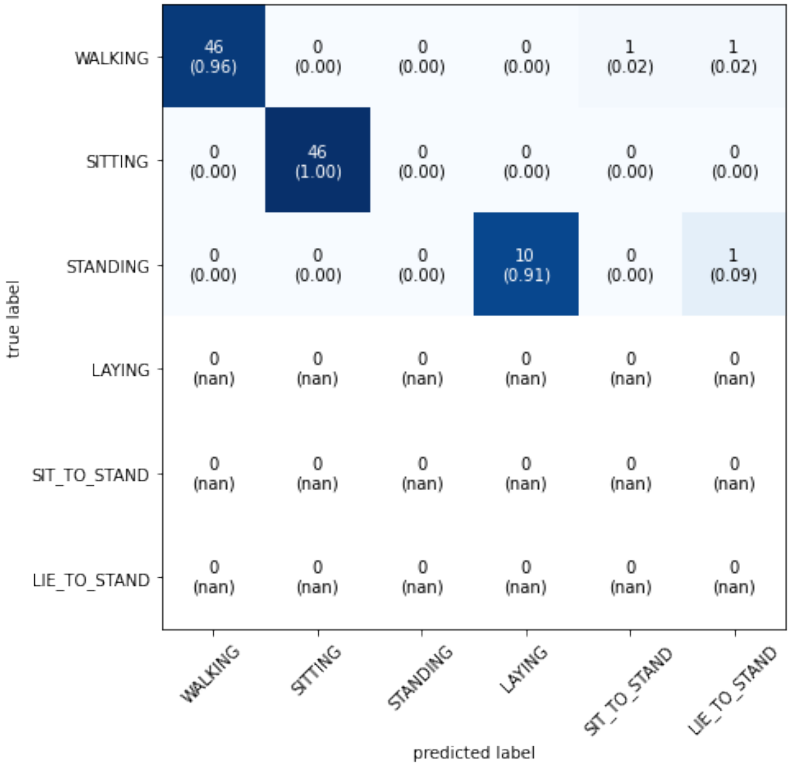
C. Gantt Chart



D. Confusion matrix for test data from (UCI) on CNN model



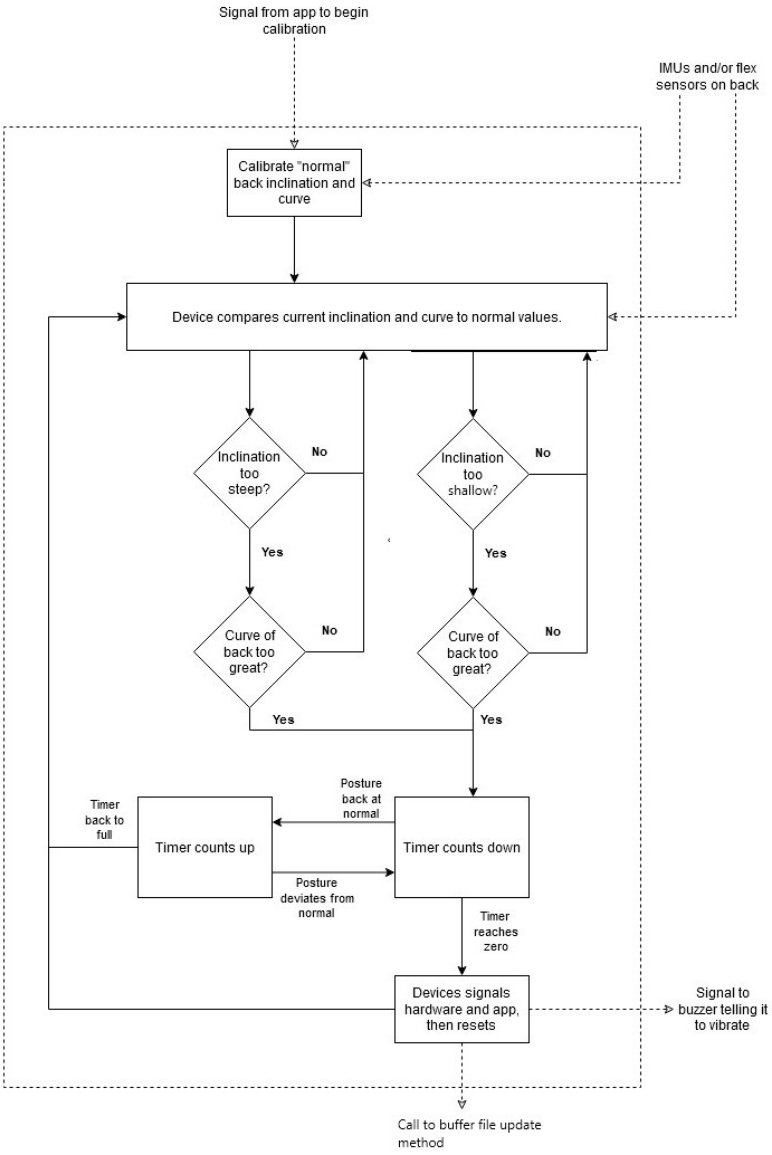
E. Confusion matrix for own test data CNN model



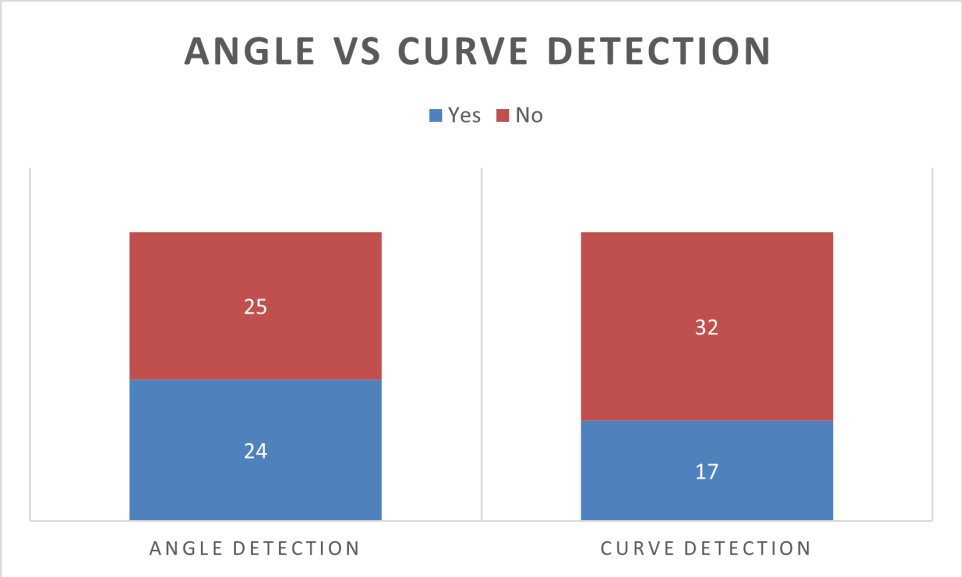
F. Labels and Data count from (UCI)

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%

G. Posture Detection State Diagram



H. Angle vs Curve Detection in Webots



I. Slouch Detection in Webots

Position	Thresholds	Angle detect	Curve Detect	Or Alg	And Alg	expected output
slight slouch	5 degrees	yes	yes	yes	yes	user dependant
(lower and upper back same angle)	10 degrees	no	no	no	no	user dependant
	15 degrees	no	no	no	no	user dependant
medium slouch	5 degrees	yes	yes	yes	yes	yes
	10 degrees	yes	yes	yes	yes	yes
	15 degrees	yes	no	yes	no	yes
large slouch	5 degrees	yes	yes	yes	yes	yes
	10 degrees	yes	yes	yes	yes	yes
	15 degrees	yes	yes	yes	yes	yes
slight curve	5 degrees	no	no	no	no	user dependant
(only upper back bent)	10 degrees	no	no	no	no	user dependant
	15 degrees	no	no	no	no	user dependant
medium curve	5 degrees	no	no	no	no	user dependant
	10 degrees	no	no	no	no	user dependant
	15 degrees	no	no	no	no	user dependant
large curve	5 degrees	no	yes	yes	no	yes
	10 degrees	no	yes	yes	no	yes
	15 degrees	no	yes	yes	no	yes
slight lean	5 degrees	yes	no	yes	no	user dependant
(only lower back bent)	10 degrees	no	no	no	no	user dependant
	15 degrees	no	no	no	no	user dependant
medium lean	5 degrees	yes	no	yes	no	yes
	10 degrees	yes	no	yes	no	yes
	15 degrees	yes	no	yes	no	yes
large lean	5 degrees	yes	no	yes	no	yes
	10 degrees	yes	no	yes	no	yes
	15 degrees	yes	no	yes	no	yes
in and out of slouch	5 degrees	no	no	no	no	no
(slight slouch)	10 degrees	no	no	no	no	no
	15 degrees	no	no	no	no	no
in and out of slouch	5 degrees	yes (20s)	yes (20s)	yes	yes	yes
(medium slouch)	10 degrees	yes (100s)	no	yes	no	yes
	15 degrees	no	no	no	no	yes
in and out of slouch	5 degrees	yes	yes	yes	yes	yes
(large slouch)	10 degrees	yes	yes	yes	yes	yes
	15 degrees	yes	no	yes	no	yes
mostly straight with small movements	5 degrees	no	no	no	no	no
	10 degrees	no	no	no	no	no
	15 degrees	no	no	no	no	no
occasional large slouch	5 degrees	no	no	no	no	no
(eg pick something up)	10 degrees	no	no	no	no	no
(c. once every 6 seconds)	15 degrees	no	no	no	no	no
slouch backwards	5 degrees	yes	yes	yes	yes	yes
(medium)	10 degrees	yes	yes	yes	yes	yes
	15 degrees	yes	no	yes	no	yes
Turn to side	5 degrees	yes	yes	yes	yes	no
(involves medium curve forward)	10 degrees	no	yes	yes	no	no
	15 degrees	no	no	no	no	no
stretch	5 degrees	yes	yes	yes	yes	no
(slight backward with small movements)	10 degrees	no	no	no	no	no
	15 degrees	no	no	no	no	no
Total		24/49	17/49	28/49	13/49	