G54MRT Coursework 2 Final Report

Title: A motion type recognition and counting system based on accelerometer and gyroscope sensors

Student ID: 20306228

Date: 31st May 2021

Word count (Excluding appendices and references): 1980 words

Summary

A system based on accelerometer and gyroscope sensors that identifies the type of movement of the athlete and calculates each movement. The system is strapped to the outermost part of the athlete's right thigh. The system uses high-pass filters, vector angles of acceleration and averaging of historical angular velocities to determine the type of movement the athlete is performing. The system counts events by setting thresholds and cancelling proximity events. The system was tested and its results had 14 true positives, 0 true negatives, 5 false positives and 2 false negatives. The system is effective in preventing incidents caused by intentional shaking of athletes.

Background and motivation

With the rapid development of smart mobile devices, various sensors (acceleration sensors, light sensors, gyroscopes, etc.) are increasingly being integrated into mobile terminals, allowing for functions that tend to be intelligent and facilitating the further development of ubiquitous computing. Caceres, R. and Friday, A. (2011) argue that ubiquitous computing can use sensors to collect contextual information about people including their state, location, environment, etc. This information can give ubiquitous computing the ability to recognise and understand people. Meanwhile, according to Chi, E.H. and et al. (2005), ubiquitous computing techniques have been widely applied to sport research and the results of these studies are encouraging. Therefore, the aim of this project is therefore to use ubiquitous computing techniques to collect data on athletes' leg movements to count the number of times an athlete performs each of the four different behavioural patterns (walking, running, turning, and star-jumping) during a training session. In response to this aim, the project has designed a motion pattern recognition system based on a combination of accelerometer and gyroscope sensors, which are fitted to the outside of the right thigh. The system allows athletes' movement patterns during training to be quantified so that coaches can help athletes plan their sports training or help athletes understand their own movement.

Related work

Rogers, Y. (2006) discusses the direction of universal computing and a statement on the future of development. This has a great impact on the design purpose of the system. Meanwhile, the design of the system is informed by the framework for sensor interaction design provided by Benford, S. and et al. (2005). The seamful and seamless design referred to by Chalmers, M. and MacColl, I. (2003)

also makes the system designed from the outset with security, uncertainty and what signals the system should collect in mind. The system was also designed to avoid ethical issues by referring to the ethical considerations provided by Brown, B. and et al. (2016) and Greenfield, A. (2006). Finally, the entire system code implementation, filtering methods and testing relies on the Sensor Processing Interactive Teaching Stuff URL provided by COMP4036 on Moodle.

Design

The sensors selected for this system are accelerometer and gyroscope sensors and the selection method is based on a taxonomy (Benford, S. and et al., 2005) which classifies the input devices according to the type of sensor input and the number of dimensions sensed.

Figure 1 shows a basic algorithmic structure of the Ubi-comp system, which takes data from accelerometer and gyroscope sensors and filters it to identify the user's movement patterns.

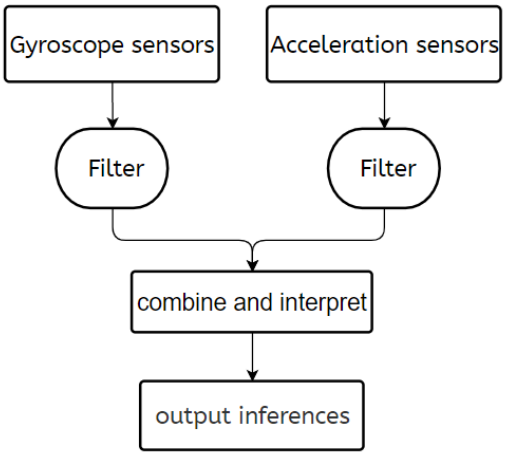


Figure 1: A basic multi-sensor algorithm structure

As shown in Figure 2, when the phone is placed face up and flat on a horizontal surface, the vertical lift of the phone affects the Z-axis of acceleration, the tilt to the left or right affects the Y-axis of the gyroscope, and so on. Therefore, the interface of the system is designed as a square and very small to fit on the leg, where the athlete is expected to fit the system and perform repetitive movements of the leg.

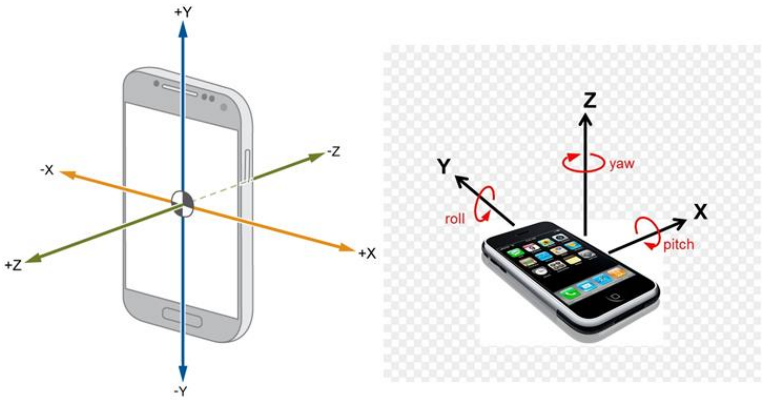


Figure 2: Acceleration sensor and gyroscope sensor

Figure 3 shows a detailed if-this-then-that logic structure for the system design. The code implementation of the system is based on this structure.

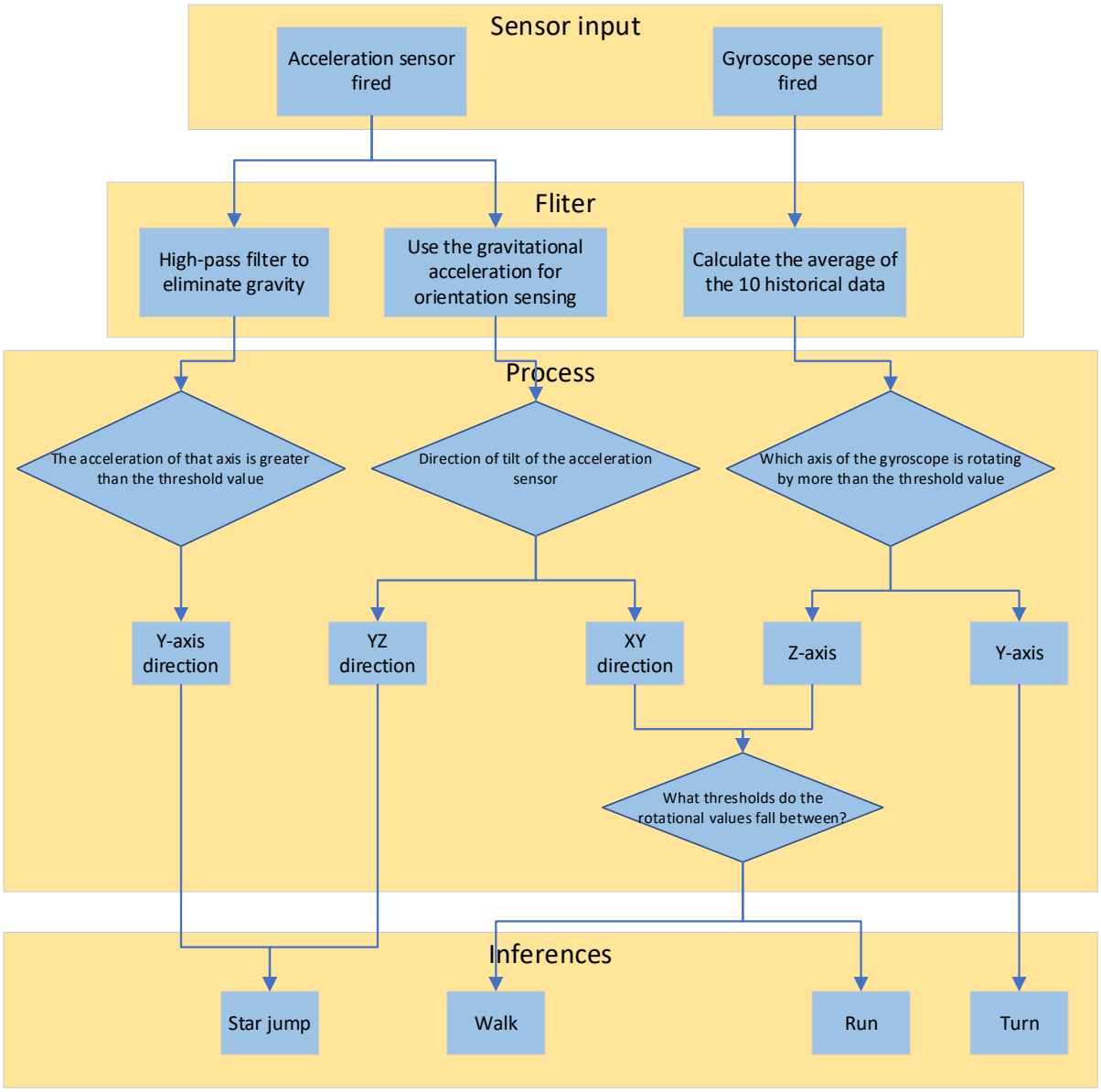


Figure 3: If-this-then-that logic

Implementation

Data reading and filtering

In figure 4, in order to read the instantaneous changes in acceleration and gyroscope during the athlete's movement, the system is set to sample 100 times per second. the aim of the project is to capture the number of times an athlete performs each movement pattern during training, so the system needs to use a high-pass filter which the filtering time is 20^{th} of a second to attenuate the low frequencies in the signal to capture the high frequencies. In addition, use math.atan2() to monitor the direction of the acceleration and if the acceleration changes, the vector angle of the change in acceleration will be calculated. Meanwhile, as the time of change in the amount of

rotation of the angular velocity is not at the same point as the change in acceleration, it is possible to judge whether rotation has occurred at the sensor by averaging the 10 historical data recorded.

```
if sensors.replayer.has_replay():
    ax, ay, az, amag, gx, gy, gz = sensors.replayer.get_level("ax","ay","az","amag","gx","gy","gz")
else:
    # get xyz values of acceleration
    (ax, ay, az)=sensors.accel.get_xyz()
    amag=sensors.accel.get_magnitude()
    # get xyz values of gyroscope
    (gx, gy, gz)=sensors.gyro.get_xyz()
# Direction of acceleration
axy_angle=math.atan2(ax, ay)
ayz_angle=math.atan2(ay, az)
axz_angle=math.atan2(ax, az)
# High-pass filter to reject gravity interference
amag_highpassed=hpFilter.on_value(amag)
ax_highpassed=hpFilter.on_value(ax)
ay_highpassed=hpFilter.on_value(ay)
az_highpassed=hpFilter.on_value(az)
gx_history.append(gx)
gy_history.append(gy)
gz_history.append(gz)
gx_mean=sum(gx_history)/len(gx_history)
gy_mean=sum(gy_history)/len(gy_history)
gz_mean=sum(gz_history)/len(gz_history)
```

Figure 4: Data reading and filtering

Event determination and threshold setting

When the phone is placed vertically and rotated in the Z axis, the data collected is shown in Figure 5. The first plot represents the angular vector of rotation of the acceleration in each direction and the second plot represents the angular velocity of the gyroscope. Based on the Figure 5 when shaken vertically, axy_angle and gz change in a regular and noticeable way. Therefore, a threshold can be set to differentiate between axy_angle and gz to determine whether the athlete is walking or running.

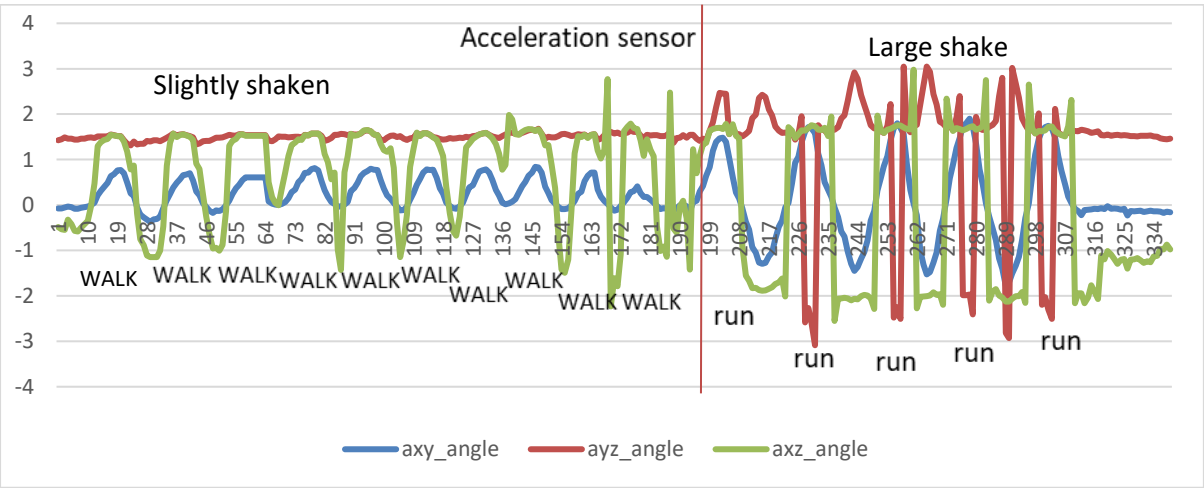


Figure 5: Vertical sway

The code snippet used to differentiate between walking and running is shown in Figure 6. When the value of axy_angle hits the acceleration threshold for walking but not for running, it then determines whether the average angular velocity of gz hits the angular velocity threshold for walking. If both are satisfied, a walk event is triggered at table time. In addition, the average angular velocity of gx and gy is used to prevent the user from intentionally jittering. The same is true for running.

```
# Walk
if axy_angle>WALK_ANGLE_THRESHOLD_UP and abs(gz_mean)>2.0 and axy_angle<RUN_ANGLE_THRESHOLD_UP and abs(gx_mean)<GYRO_SHAKE_THRESHOLD and abs(gy_mean)<GYRO_SHAKE_THRESHOLD:
    walk_threshold=1
elif axy_angle<WALK_ANGLE_THRESHOLD_DOWN:
    walk_threshold=0
# Run
if axy_angle>RUN_ANGLE_THRESHOLD_UP and abs(gz_mean)>2.0 and abs(gx_mean)<GYRO_SHAKE_THRESHOLD and abs(gy_mean)<GYRO_SHAKE_THRESHOLD :
    run_threshold=1
elif axy_angle<RUN_ANGLE_THRESHOLD_DOWN:
    run_threshold=0
```

Figure 6: Judging walk and run

As shown in Figure 7, there is a clear and regular change in ay, amag when a star-jump is performed. Meanwhile, the y-axis and z-axis of the gyroscope theoretically do not change when the leg is open and closed, and the direction of acceleration is theoretically along the y-axis and z-axis, so the star-jump action is recognised by setting thresholds for yz_angle,gy,gz,ay,amag, the code for which is shown in Figure 8.

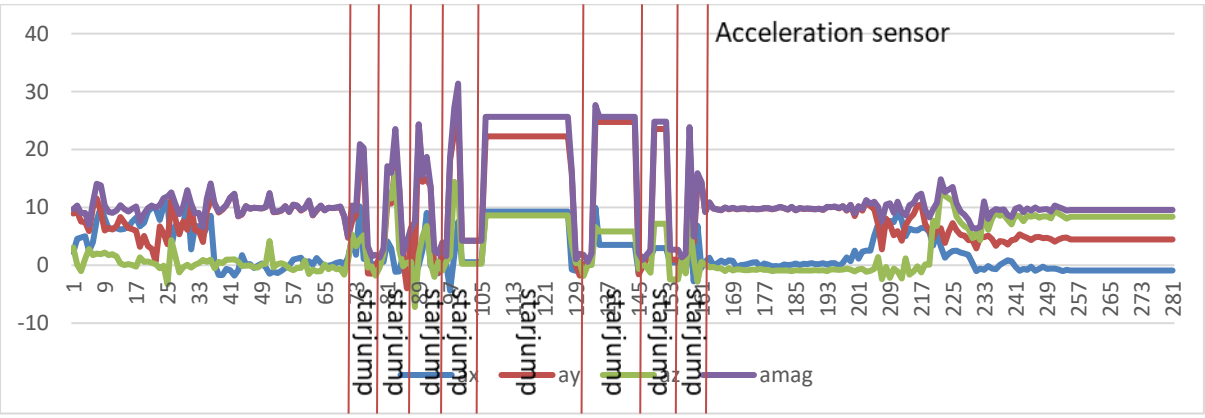


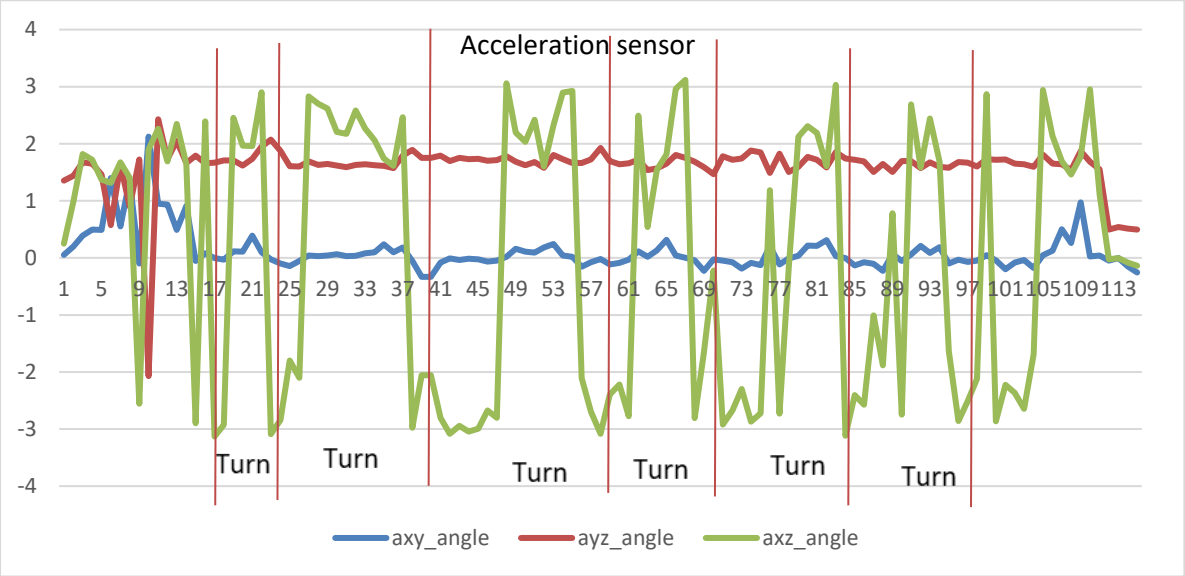
Figure 7: Star-jump sway

```
 # Star-jump
if ayz_angle<STARJUMP_ANGLE_THRESHOLD_DOWN:
    yz_up=0
elif ayz_angle>STARJUMP_ANGLE_THRESHOLD_UP:
    yz_up=1
if yz_up==1 and ay_highpassed>STARJUMP_AY_THRESHOLD and amag_highpassed>STARJUMO_AMAG_THRESHOLD and gy_mean<GYRO_SHAKE_THRESHOLD and gz_mean<GYRO_SHAKE_THRESHOLD:
    starjump_threshold=1
else:
    starjump_threshold=0
```

Figure 8: Judging star-jump

In Figure 9, the change in axz_angle and gy is most pronounced when the sensor is steered, so by specifying a threshold, when axz_angle and gy touch the threshold, the user is proven to have

performed the action of steering.



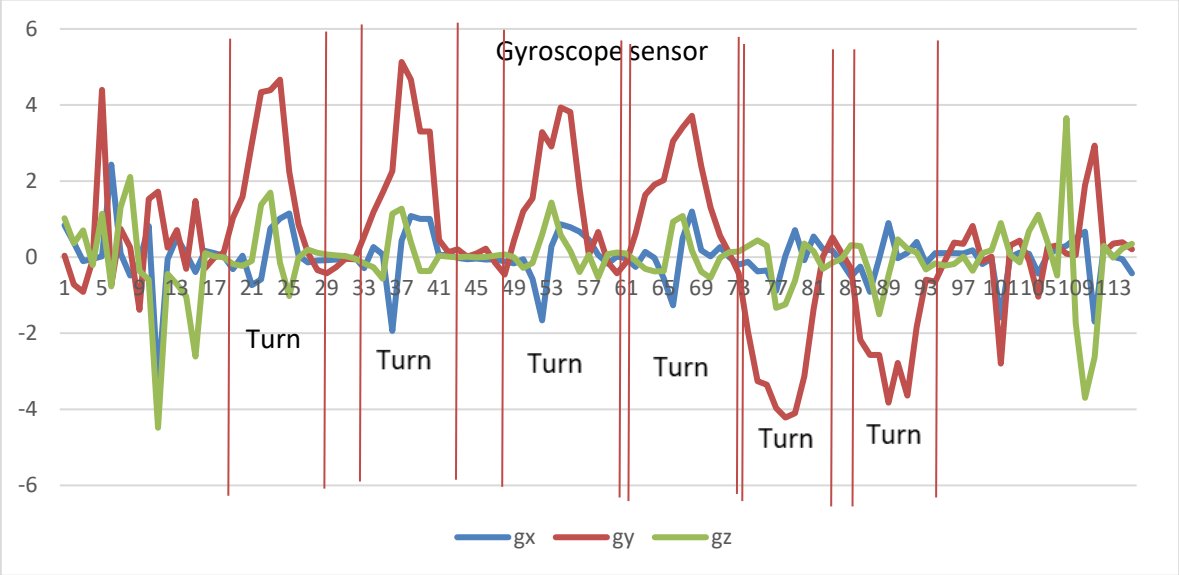


Figure 9: Turning sway

As shown in Figure 10, the magnitude of the change in gy and axz_angle is used to distinguish whether the user makes a turn or not. Also, the gx,gz and ayz_angle are used to avoid the user deliberately jittering to trigger the event.

```
# Turn
if ((axz_angle>ANGLE_TURN_THRESHOLD or axz_angle<-ANGLE_TURN_THRESHOLD) and abs(gy_mean)>GYRO_TURN_THRESHOLD and
    abs(gx_mean)<GYRO_SHAKE_THRESHOLD and abs(gz_mean)<GYRO_SHAKE_THRESHOLD and ayz_angle<TURN_YZ_ANGLE_UP and ayz_angle>TURN_YZ_ANGLE_DOWN):
    turn_threshold=1
else:
    turn_threshold=0
```

Figure 10: Judging turning

Event triggering and result output

To prevent events being triggered multiple times during a single action, the system uses a set threshold method and cancel close together method which is shown in Figure 11. The threshold is 1 when the user performs the action and 0 when the action ends. last_threshold is also used to record

the last threshold data. If last_threshold is equal to 0 and current threshold is equal to 1 the user has completed a complete action and the event is triggered. As second method, the program sets MIN_TIME_BETWEEN_EVENTS, and the event will not be triggered until at least MIN_TIME_BETWEEN_EVENTS samples are sampled.

```
# Fire event
if (walk_last_threshold==0 and walk_threshold==1):
    if time_since_last_walk>=MIN_TIME_BETWEEN_EVENTS:
        time_since_last_walk=0
        speech.say("walk")
        walk+=1
elif (run_last_threshold==0 and run_threshold==1):
    if time_since_last_run>=MIN_TIME_BETWEEN_EVENTS:
        time_since_last_run=0
        speech.say("run")
        run+=1
elif (starjump_last_threshold==0 and starjump_threshold==1):
    if time_since_last_starjump>=MIN_TIME_BETWEEN_EVENTS:
        time_since_last_starjump=0
        speech.say("star jump")
        starjump+=1
elif (turn_last_threshold==0 and turn_threshold==1):
    if time_since_last_turn>=MIN_TIME_BETWEEN_EVENTS:
        time_since_last_turn=0
        speech.say("turn")
        turn+=1
walk_last_threshold=walk_threshold
run_last_threshold=run_threshold
turn_last_threshold=turn_threshold
statjump_last_thrshold = starjump_threshold
print(ax, ay, ay_highpassed, az, amag, amag_highpassed, gx, gy, gz, axy_angle, ayz_angle, axz_angle,
WALK_ANGLE_THRESHOLD_UP, WALK_ANGLE_THRESHOLD_DOWN, RUN_ANGLE_THRESHOLD_UP, RUN_ANGLE_THRESHOLD_
DOWN, WALK_GZ_THRESHOLD,
RUN_GZ_THRESHOLD, STARJUMP_ANGLE_THRESHOLD_UP, STARJUMP_ANGLE_THRESHOLD_DOWN, STARJUMP_AY_THRESH
OLD, STARJUMP_AMAG_THRESHOLD,
GYRO_TURN_THRESHOLD, ANGLE_TURN_THRESHOLD, TURN_YZ_ANGLE_UP, TURN_YZ_ANGLE_DOWN, walk, run, turn, st
arjump, sep=",")
```

Figure 11: Fire events and output results

Testing

Before starting the test, place the system against the outside of your left thigh, as shown in Figure 12. After clicking start, the body should be stationary for 3 seconds before starting the movement. The four movements are performed in sequence: walking, running, star-jumping, and turning. Also, perform each movement. 5 times, pausing for 5 seconds between each movement.

Figure 12: The system position

Motion recognition test

According to Figure 13, it can be seen that one walk event was incorrectly triggered during the jitter and two walk events were incorrectly triggered during the run, the rest of the event triggers were accurate. During jitter, the walk event plus one is caused by the threshold value concerning the walk being met. Therefore, new thresholds need to be developed by looking at the waveforms and values of the other signals from the sensor to enable the walk events to be masked out during jitter. The reason for adding one to the walk event during a run is also that the threshold set does not fully distinguish between walking and running. It is necessary to update the threshold or to find a threshold from other signals to increase the condition.

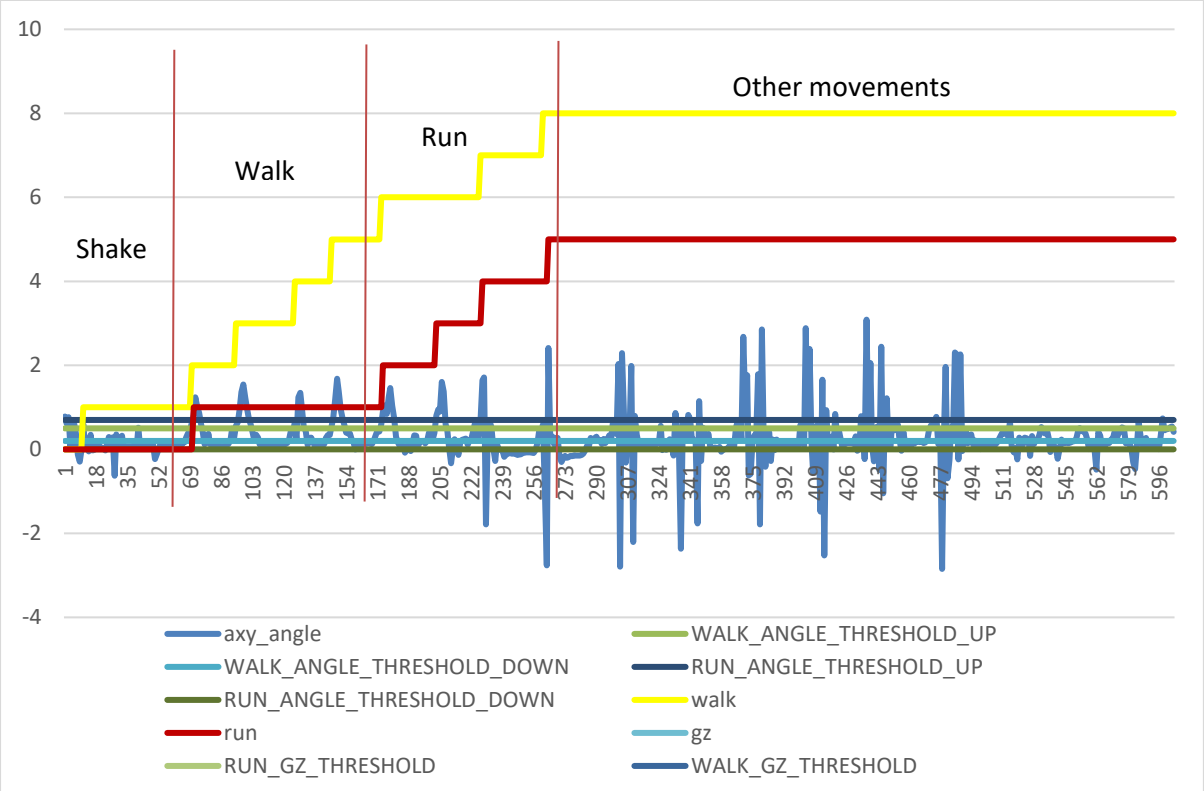


Figure 13: Walk and run events

As shown in Figure 14, the system only recognised two actions during the star-jump, with two actions being ignored. It is possible that the star-jump was ignored twice because a threshold was set too high and was not touched during the movement. It is also possible that the MIN_TIME_BETWEEN_EVENTS was set too large when using the cancel close together events method, causing the two events to be divided into similar events and ignored.

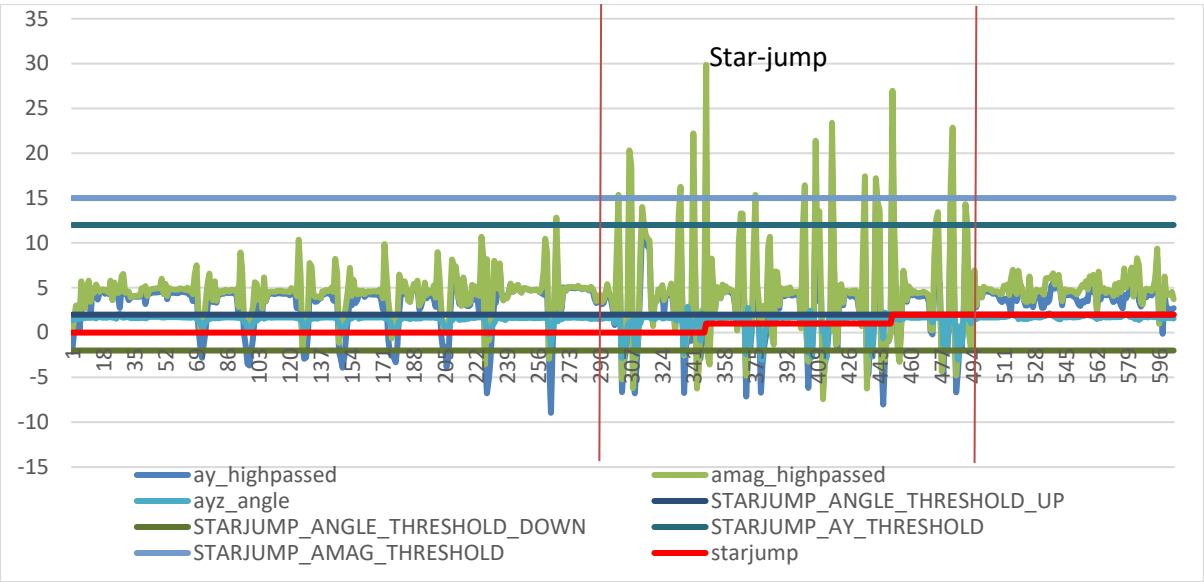


Figure 14: Star-jump events

According to Figure 15, the system accurately identifies four steering movements.

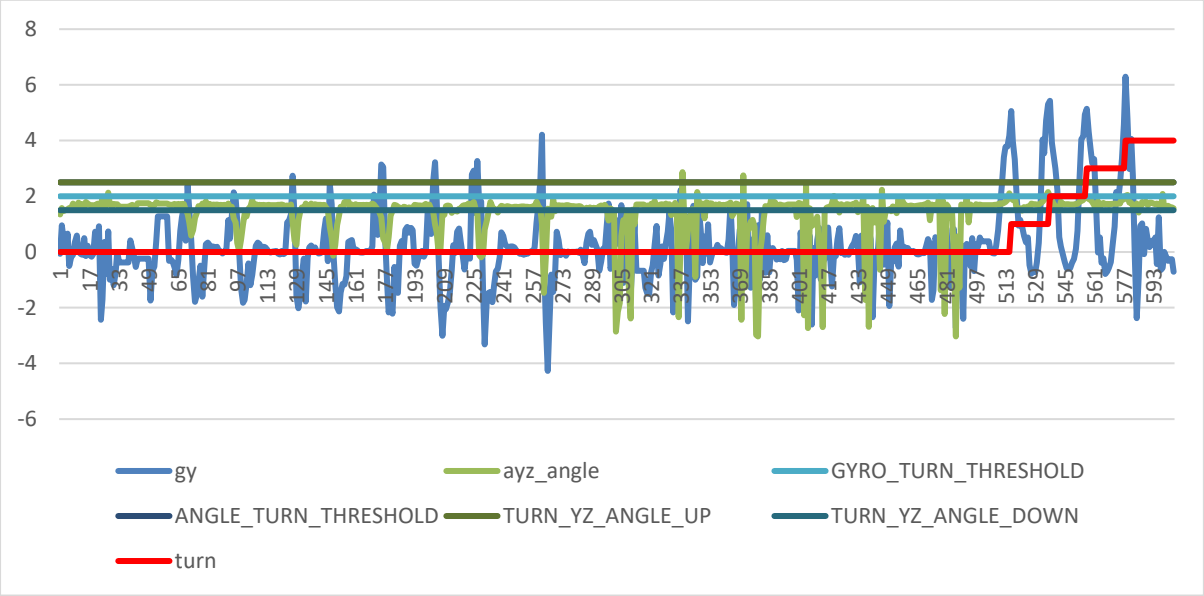


Figure 15: Turn events

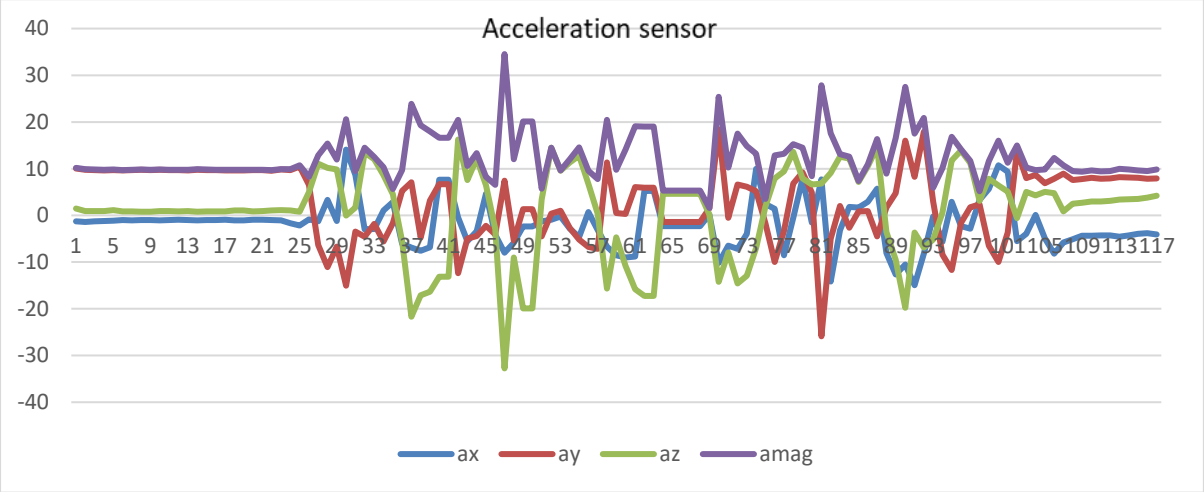
The results of the above test can be presented in the form of a truth table. False negative results occurred 2 times and false positive results occurred 5 times. The rest of the results were correct.

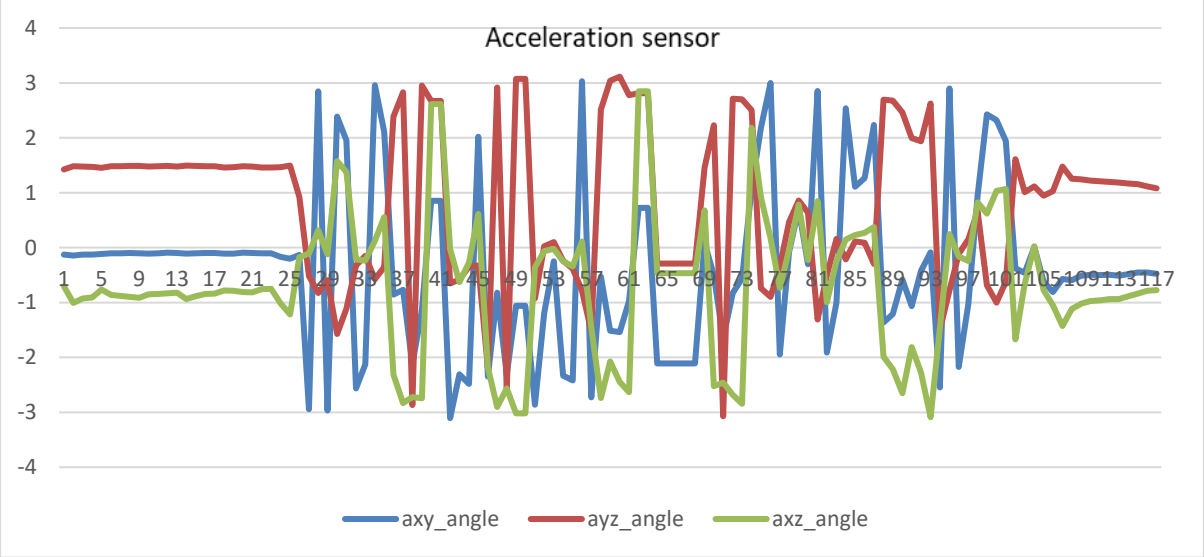
Sensor report	Real event	
	TRUE	FALSE
TRUE	14	5
FALSE	2	0

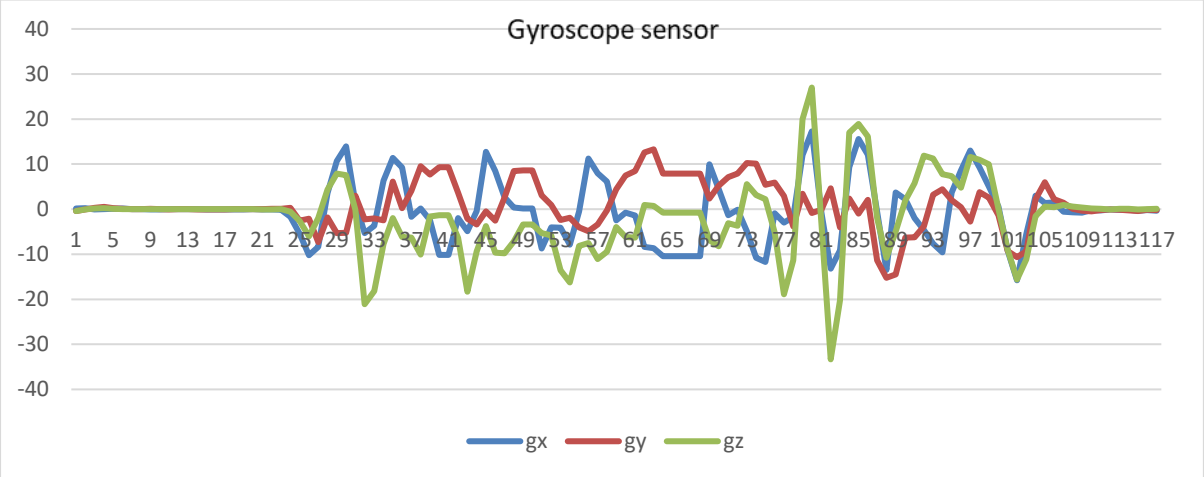
Figure 16: All events Trigger Truth Table

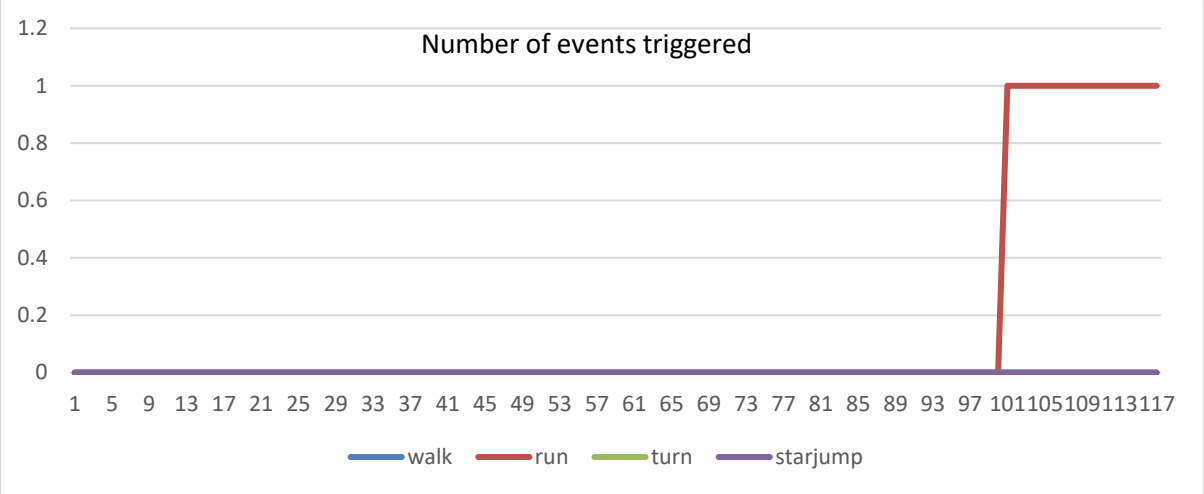
Intentional shake test

As can be seen in Figure 17, when the user deliberately jiggles the sensor, all events other than run not be triggered due to a judgment statement in the program, and run is only triggered once. The reasons for this are very similar to those in the previous section, where the walk was triggered by jitter.









Sensor report	Real event	
	TRUE	FALSE
TRUE	N/A	1
FALSE	N/A	N/A

Figure 17: Intentional shake

Critical reflection

Ethical consideration

The ethical issues involved in this system combine the views of Brown, B. and et al (2016) and Greenfield, A. (2006). Brown, B. and et al. argue that users' data should be protected, that users should have the right to know and that they should not be harmed. As a supplement, Greenfield, A. argues that the design of the software must also comply with the following conditions: the software is open and transparent, and the user software has the right to reject the software and specify the time of use of the software. The design of the system follows these ethics, the users are clear about the purpose of the system and how their data is collected, and the data collected is held by the users themselves. At the same time, users voluntarily use the system for testing, and the testing is limited to the time the users are in their sport exercises.

System design and testing

During the design phase, the system followed the design guidelines of Benford, S. and et al. (2005) for interactive sensors. The design of the system anticipates the possible actions of the user and the factors such as threshold, speed, accuracy, and stability of these actions. Consideration was also given to what signals from the sensor could distinguish these actions. However, during the code implementation phase, too much noise made it extremely difficult to set thresholds, which resulted in many actions not being accurately identified during testing. This requires iterative testing of the sensor to update the threshold or to increase the judgement by finding other sensor signals that can recognise the action.

References

- Benford, S., Schnädelbach, H., Koleva, B., Anastasi, R., Greenhalgh, C., Rodden, T., Green, J., Ghali, A., Pridmore, T., Gaver, B. and Boucher, A. (2005) Expected, sensed, and desired: A framework for designing sensing-based interaction. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *12*(1), pp.3-30.
- Brown, B., Weilenmann, A., McMillan, D. and Lampinen, A. (2016) Five provocations for ethical HCI research. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (pp. 852-863).
- Caceres, R. and Friday, A. (2011) Ubicomp systems at 20: Progress, opportunities, and challenges. *IEEE Pervasive Computing*, *11*(1), pp.14-21.
- Chalmers, M. and MacColl, I. (2003) Seamful and seamless design in ubiquitous computing. In *Workshop at the crossroads: The interaction of HCI and systems issues in UbiComp* (Vol. 8).
- Chi, E.H., Borriello, G., Hunt, G. and Davies, N. (2005) Guest editors' introduction: Pervasive computing in sports technologies. *IEEE Pervasive Computing*, *4*(3), pp.22-25.
- Greenfield, A. (2006) *Everyware*. Pearson Education Incorporated.
- Rogers, Y. (2006) Moving on from weiser's vision of calm computing: Engaging ubicomp experiences. In *International conference on Ubiquitous computing* (pp. 404-421). Springer, Berlin, Heidelberg.

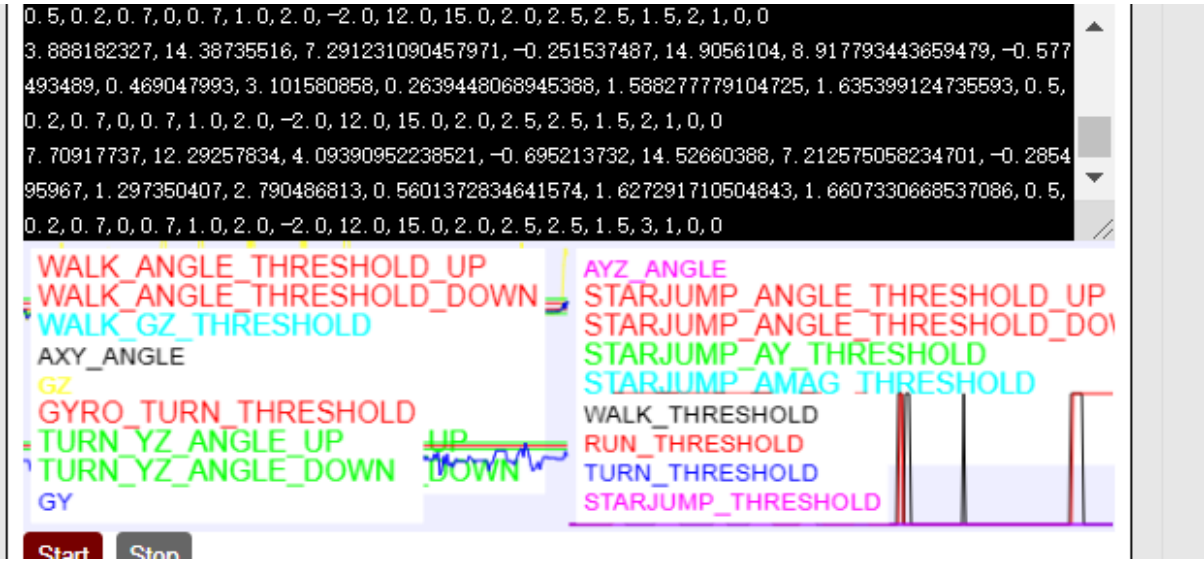
Appendix A – instructions

No data

If no data is available, simply load the system code with your mobile phone and strap it to the outside of your right thigh for leg movements which is shown in the figure below. After the user has heard start, the program has been run. Whenever the system recognises an action during the

program, a voice announcement is made. The system recognises a total of 4 movements: walk, run, turn and stat-jump.

At the end of the system, an image of the sensor signal, an image of the event trigger and a count of each movement can be seen on the phone, as shown in the figure below.



Data already available

If you already have data, just click on the load replay csv... button to select the dataset. Once you have done this, simply click on start to see the image of the sensor signal and the count of each movement, which is shown in the figure below.

[Appendix B – NOTE: You need to submit your coursework as zip file to Moodle – including this report and source code, test data files etc.]

The zip file contains 7 files:

- COMP4036_20306228_CHAO CUI.pdf
- Final_test.csv
- Shake_test.csv
- Starjump_sway.csv
- Turn_sway.csv
- Vertical_sway.csv
- 20306228_ChaoCui

The pdf file is a report of the system.

The csv files are the data used for the test.

The 20306228_Chao Cui file is the python script of the system.