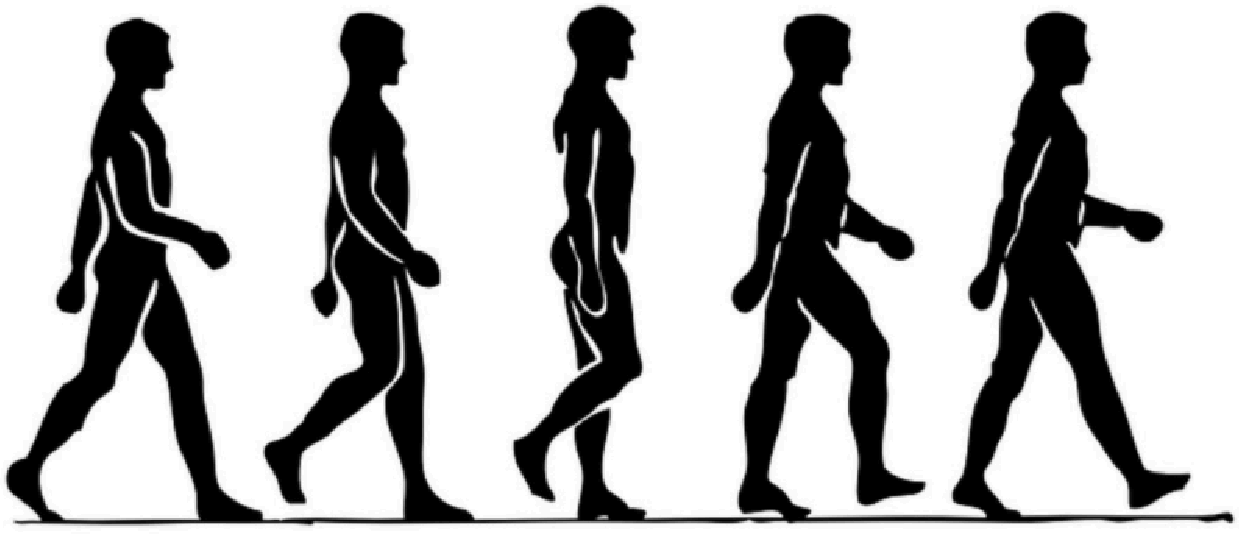
COMP6733 Final Report



Biometric Authentication using Gait Analysis in Wearable Devices

Otto Flaherty z5116011

Michael Gysel z5251938

Stephen Comino z3470429

Hasaru Kariyawasam z5232917

Introduction

The COVID-19 pandemic has dramatically changed normal life for billions of people globally. As such, governments, organisations, and individuals need better tools to help them adapt and overcome this hardship.

Being in a position to help in the efforts against COVID-19, Jim's IoT aims to produce an Internet of Things solution to aid people through this challenging time.

The Problem

Through the Australian Defence Force's door knocking in Melbourne, the Victorian government discovered that more than 25% of people with Coronavirus were not at home during their mandatory quarantine period [6]. This kind of behaviour increases the risk of community transmission and further complicates work for contact tracers, which has detrimental health and economic effects for everyone.

In order to reduce the number of people breaking self quarantine we need a cost-effective way to ensure that those under lockdown are staying home and are accounted for. Currently, almost everyone carries a smartphone; moreover, these smart phones are able to track our locations and stream this data to the many services we all subscribe to, such as Google Maps. While a government could use location data from a smartphone or smartwatch to track whether or not a user is staying at home, the user could easily leave their device at home. This would allow users to break quarantine without the government's knowledge, making the location data useless. Thus, we need a way to improve the government's ability to ensure users remain at home during quarantine.

One way to improve the government's ability to ensure that users remain at home is to ensure that location data represents the true location of an individual.

Solution Overview

In this document, we propose a user authentication method that would work in tandem with other projects being developed by other UNSW teams, to provide an extremely accurate way to track and trace individuals. We are proposing a biometric authentication system that is resilient to attacks, such as a user providing their password to someone else. We are targeting authentication when people are on the move, like when they are walking to shops.

Due to the IoT based nature of the team, our solution comes from the IoT world, so we propose the use of a wearable that is able to provide the biometric authentication we require. The form of biometric authentication that we aim to target is gait based authentication due to the unique form of a person's gait. Due to our varying physical structures, no two gaits are exactly alike. Furthermore, the prevalence of smart watches provides a perfect opportunity for accurately sensing a person's gait, due to many already containing the sensors needed. Thus, we will develop a proof of concept that could be integrated into a smart watch.

The wearable measures the gait through onboard sensors such as accelerometers and gyroscopes, and it uses the data received and compares it against previous gait data on the user known to be accurate. We use machine learning to develop the authentication algorithm for our program.

Our solution provides a central authority such as a local government with the ability to monitor individual devices through a web application that would assess if the wearables have been in possession by their owners or a third party.

Networking

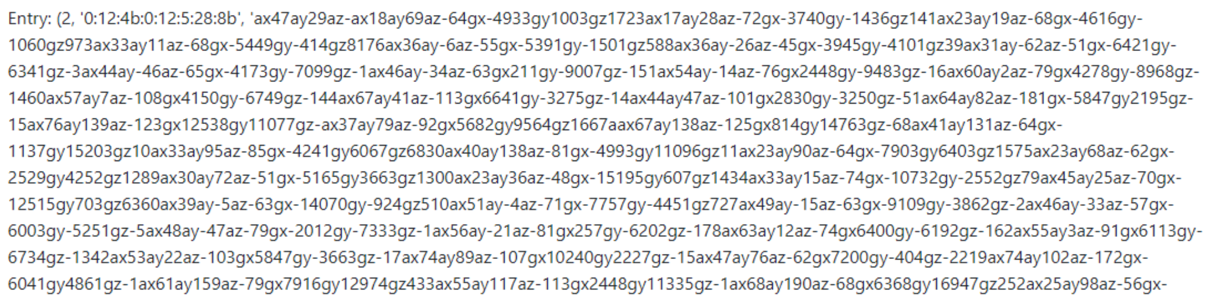
We attempted various methods for transferring accelerometer gait data at 50Hz from the SensorTag to the web application using the Constrained Application Protocol (CoAP), including block transfer with a periodic resource, block transfer with a timer, block transfer with multiple buffers. We also tested using a simple TCP server and client which was not suitable for a IoT device like the SimpleLink. For all of these methods, the fastest rate data could be transferred without significant packet loss was approximately 30Hz.

When testing these methods, we validated the TI SimpleLink SensorTag's ability to both collect accelerometer data and store it in a buffer variable. The SensorTag showed no issues printing results at the rate of 50Hz. Next, we validated the data transfer from the RPL Border Router to the web application, which did not show any data loss. Upon transferring the accelerometer data at the rate of 50Hz from the SensorTag to the RPL border router however, there was an approximate 20% to 34% data loss rate, depending on the method used. We found that the size of our data affects the packet loss, with larger data per block we found that we lost more data. When transferring fixed data that did not require use of the SensorTag's sensor readings, these data loss rates held. Previous research has shown that CoAP has issues handling rapid, frequent requests, as this results in buffer overflow and packet loss [7]. Thus, we attempted the use of multiple buffers when transferring data, but these data loss rates still held. The Internet Engineering Task Force (IETF) is currently developing Block-Wise transfer options Block3 and Block4 that enable faster transmissions of sets of blocks of data, but these transfer options are currently unavailable using the Python CoAP library, aiocoap [8].

Thus, it appears CoAP blockwise transfers have current limitations to the speed at which data can be sent, with our research suggesting the upper limit to be approximately 30Hz.

Data Cleaning

When a packet is received from the server it is in a large string containing the sequential readings in format like:



```
Entry: (2, '0:12:4b:0:12:5:28:8b', 'ax47ay29az-ax18ay69az-64gx-4933gy1003gz1723ax17ay28az-72gx-3740gy-1436gz141ax23ay19az-68gx-4616gy-
1060gz973ax33ay11az-68gx-5449gy-414gz8176ax36ay-6az-55gx-5391gy-1501gz588ax36ay-26az-45gx-3945gy-4101gz39ax31ay-62az-51gx-6421gy-
6341gz-3ax44ay-46az-65gx-4173gy-7099gz-1ax46ay-34az-63gx211gy-9007gz-151ax54ay-14az-76gx2448gy-9483gz-16ax60ay2az-79gx4278gy-8968gz-
1460ax57ay7az-108gx4150gy-6749gz-144ax67ay41az-113gx6641gy-3275gz-14ax44ay47az-101gx2830gy-3250gz-51ax64ay82az-181gx-5847gy2195gz-
15ax76ay139az-123gx12538gy11077gz-ax37ay79az-92gx5682gy9564gz1667aax67ay138az-125gx814gy14763gz-68ax41ay131az-64gx-
1137gy15203gz10ax33ay95az-85gx-4241gy6067gz6830ax40ay138az-81gx-4993gy11096gz11ax23ay90az-64gx-7903gy6403gz1575ax23ay68az-62gx-
2529gy4252gz1289ax30ay72az-51gx-5165gy3663gz1300ax23ay36az-48gx-15195gy607gz1434ax33ay15az-74gx-10732gy-2552gz79ax45ay25az-70gx-
12515gy703gz6360ax39ay-5az-63gx-14070gy-924gz510ax51ay-4az-71gx-7757gy-4451gz727ax49ay-15az-63gx-9109gy-3862gz-2ax46ay-33az-57gx-
6003gy-5251gz-5ax48ay-47az-79gx-2012gy-7333gz-1ax56ay-21az-81gx257gy-6202gz-178ax63ay12az-74gx6400gy-6192gz-162ax55ay3az-91gx6113gy-
6734gz-1342ax53ay22az-103gx5847gy-3663gz-17ax74ay89az-107gx10240gy2227gz-15ax47ay76az-62gx7200gy-404gz-2219ax74ay102az-172gx-
6041gy4861gz-1ax61ay159az-79gx7916gy12974gz433ax55ay117az-113gx2448gy11335gz-1ax68ay190az-68gx6368gy16947gz252ax25ay98az-56gx-
```

Our server then runs a python script to process and clean this data. It uses Regex to pattern match the readings, discarding any readings that are not viable such as a reading containing an AX and AY reading but no AZ reading as the machine learning program is unable to deal with improper data.

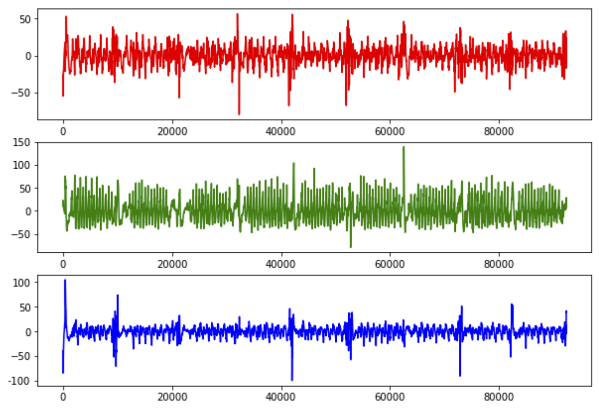
It converts these readings into three arrays ax_array, ay_array and az_array which are then able to be fed into our machine learning pipeline.

Machine Learning Pipeline

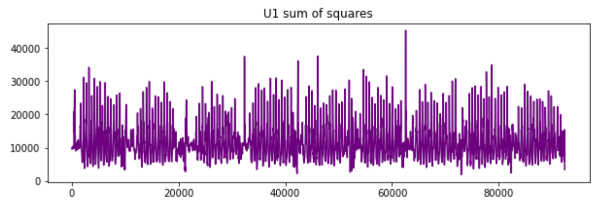
In our machine learning pipeline, the class label U1 represents Otto who is the intended wearer of the tag, whereas U2 represents an imposter. The goal of our binary classifier is to predict whether a given gait sample belongs to Otto or an imposter. We collected gait data from Otto and Hasaru to generate U1 and U2 instances respectively with wrist placement on the right hand.

The concatenated accelerometer gait signals for model training are shown below for U1 and U2. Regular gait patterns are visible; regular cyclical peaks in the y-axis were used to segment the signal, as using peaks in the y-axis gave optimal accuracy compared to the other axes. A bandpass filter in the range [0.25hz, 15hz] was applied to the gait signal. A separate "SquareSum" signal was collected to represent the net acceleration; it is simply the sum of the X, Y and Z axes readings which have each been squared.

]

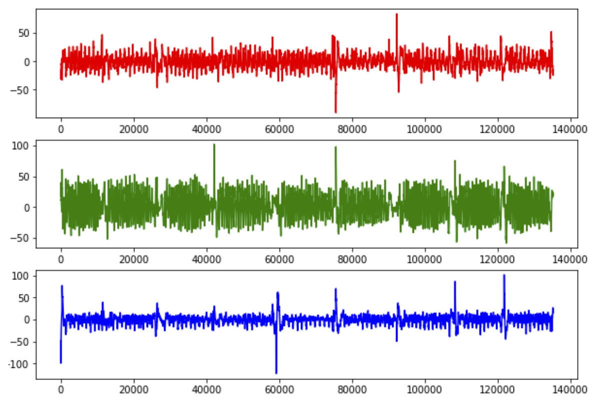


(a): Accelerometer X, Y and Z axes for U1

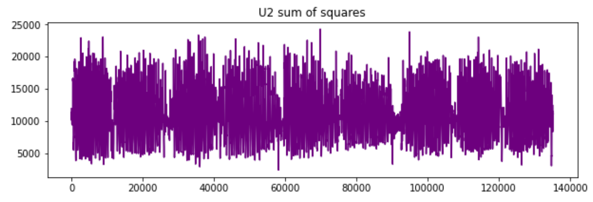


(b) Sum of Squares for U1

Figure 1: U1 Signals



(a) Accelerometer X, Y and Z axes for U2



(b) Sum of Squares for U2

Figure 2: Accelerometer X, Y and Z axes for U2

Feature Extraction

Approach (i): Full Windows

We located peaks in the accelerometer y-axis using scipy.signal.find_peaks using a threshold distance of 600ms between peaks. The goal is to isolate the strike points to create individual gait cycles. We then create windows centred around each peak, the total window length is 400ms, which translates to 20 readings on either side of the peak, with 40 readings in total. The corresponding x and y axis windows are concatenated onto this to form an instance. Gyroscope axis windows were optionally concatenated onto each instance as well (discussed later).

Approach (ii): Squashed Windows

As an alternative to using the full window of 40 readings for each signal to generate an instance centred around a peak, we took the mean and median of the window and added this to the instance. We refer to this as the squashed window feature set. In both approaches, 87 U1 instances and 153 U2 instances were extracted to form a dataset of 240 instances in total.

Performance

Table 1: Full Window Accuracy

Model || Feature Sets	Accel.	Norm. accel	Norm accel. + Gyro + SquareSum	Gyro + SquareSum	SquareSum
SVM	0.909	0.927	0.905	0.909	0.901
kNN	0.930	0.909	0.935	0.935	0.914
LR	0.830	0.884	0.875	0.875	0.815
RF	0.904	0.905	0.927	0.931	0.922

Table 2: Squashed Window Accuracy

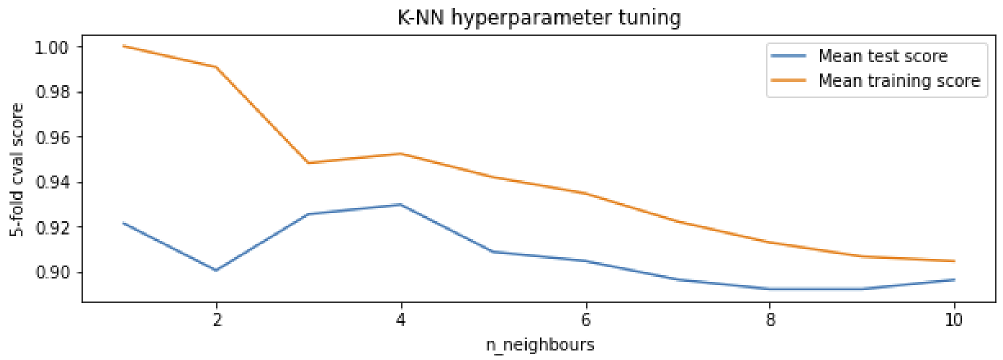
Model || Feature Sets	Accel.	Norm. accel	Norm accel. + Gyro + SquareSum	Gyro + SquareSum	SquareSum
SVM	0.767	0.862	0.810	0.8232	0.780
kNN	0.743	0.832	0.815	0.836	0.741
LR	0.759	0.733	0.866	0.810	0.759
RF	0.780	0.879	0.948	0.876	0.754

Five separate feature sets were generated to evaluate their performance across various classifiers, shown in the columns of Table 1 and 2. Norm accel. refers to the accelerometer data which has been normalised to obtain a mean of 0 in order to control for changes in pace.

5-fold cross validation was used to measure the performance of four different classifiers: a linear kernel SVM, kNN with n=4, logistic regression and a random forest. We found that the full window feature sets generally achieved better accuracy overall compared to the squashed window. When using squashed windows, we found that normalising accelerometer signals to give a mean of 0 delivered a large average increase in accuracy of 6.4% across all four classifiers. When using the full windows, normalising the accelerometer signal had a small average impact on accuracy (1.3%). The highest accuracy (94.8%) was achieved by using a squashed window feature set consisting of the normalised acceleration data, gyroscope signals and the SquareSum signal, and a random forest classifier (highlighted in green). However, we were unable to implement obtaining gyroscope data in a live setting due to issues in CoAP data transfer. The next best feature set and classifier combination (achieving 93% accuracy, highlighted in blue) used full windows, accelerometer data only and the K-nearest neighbours classifier (with n=4). This is the implementation used in our gait identification Flask application, with the training feature set saved locally, ready to be loaded for online fitting and classification.

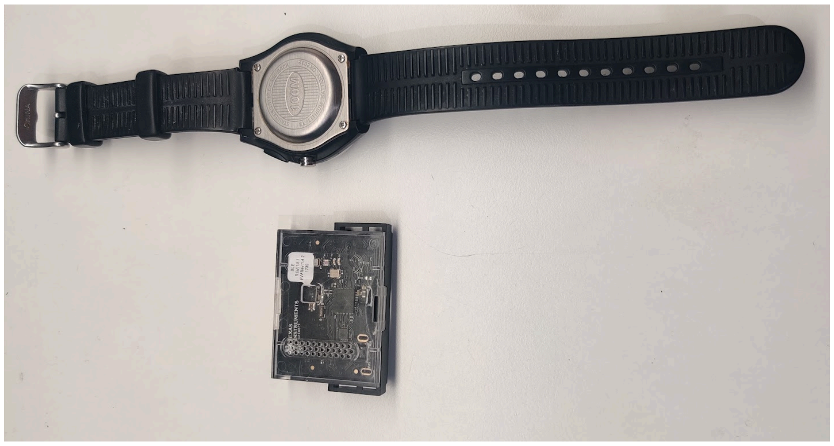
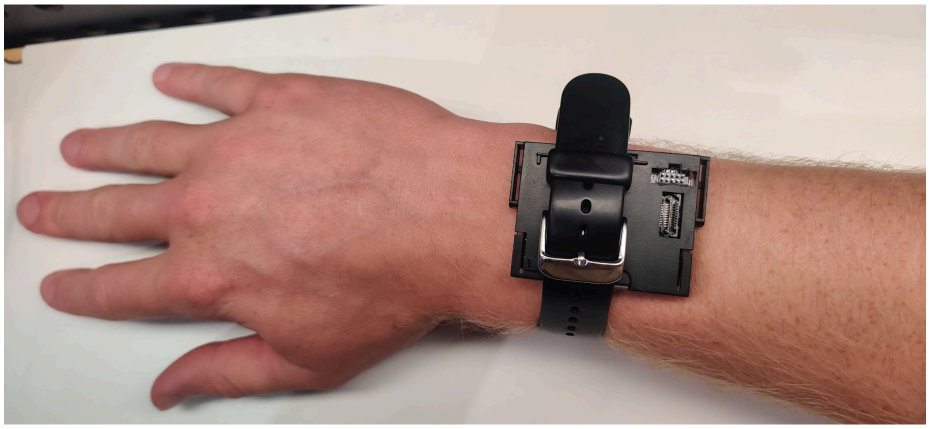
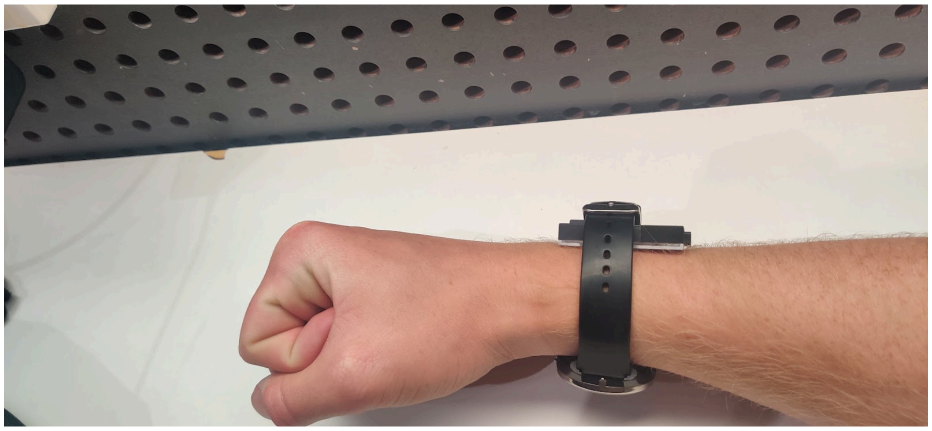
Hyperparameter Tuning

After choosing kNN as our classifier, we utilised 5-fold cross-validation grid search to optimise the number of neighbours. The best value was indeed n=4, achieving 93% accuracy on average on the test set. We can see that increasing n further leads to an overall decline in accuracy. Although n=1 delivers perfect accuracy on the training set, it is clear that this is due to overfitting, as each instance being evaluated would already exist in the training set - its nearest neighbour is itself.



Physical Setup

Our implementation requires the TI Simplelink SensorTag Development Kit - CC2650STK, and a wrist strap. For the wrist strap we made use of an old wristwatch which was able to securely hold our sensor tag in our testing.

How to Use

In order to make use of our solution we have developed a series of steps for use, that cover registering with the site, how to gather initial user training data, and how to monitor users and verify their gait data.

1. **Using the Web Application:**
    a. User training data can be found at: http://flask-env-testing.eba-4kjraqyp.us-east-2.elasticbeanstalk.com/
    b. User authentication can be found at: http://flask-env-testing.eba-4kjraqyp.us-east-2.elasticbeanstalk.com/test
2. **Gathering Training Data:**
    a. Data collection must be conducted by the authority personnel with the user present. Otherwise, the user could provide false or misleading data.
    b. The authority personnel must attach the SensorTag to the user's wrist in the orientation shown in the 'The Physical Setup' section; the orientation is significant as it will impact the program's ability to identify a user.
    c. The authority personnel must then run our data collection program to collect the user's accelerometer data while the user walks in a straight line at a relaxed pace for 2 minutes.
    d. This program then processes the user data and stores the resulting training data in our hosted database.
3. **Monitoring Users**
    a. When the user is wearing the SensorTag with our user authentication program, the SensorTag sends the user accelerometer data to the web application, where it is authenticated.
    b. The web application then displays whether the user's most recent accelerometer data was authenticated, and can be used to monitor each individual by the authority personnel.

Conclusion

Our research suggests that CoAP has issues handling rapid, frequent requests; furthermore, these issues are not resolved when using blockwise transfer or multiple buffers. When using CoAP blockwise transfer with multiple buffers, our research suggests the upper limit at which data can be sent without significant data loss is approximately 30Hz. In order to send data using CoAP above 30Hz with low-energy devices, we believe new Block-Wise transfer options are necessary, such as the Block3 and Block4 Options being developed by IETF. While data can be sent without data loss using HTTP and TCP, the energy constraints of the SensorTag disallow its use for energy intensive use cases, such as constant user authentication.

We have presented a data pipeline for processing TI SensorTag accelerometer and gyroscope data and performing gait authentication using instance-based machine learning methods. After exploring numerous feature sets, it was established that a combination of compressed sliding windows, normalised accelerometer data, gyroscopic data and a net acceleration feature achieved optimum accuracy (95%) using a random forest classifier. Due to data transfer constraints, a slightly lower accuracy (93%) proof-of-concept k-NN algorithm was implemented using full sliding windows and accelerometer data only, to perform live gait authentication.

Future work should incorporate larger gait datasets to ensure good accuracy when testing on random individuals.

Bibliography

[1] H. Lu, J. Huang, T. Saha, and L. Nachman, “Unobtrusive gait verification for mobile phones,” in Proceedings of the 2014 ACM International Symposium on Wearable Computers, Seattle, Washington, Sep. 2014, pp. 91–98, doi: 10.1145/2634317.2642868.

[2] F. Juefei-Xu, C. Bhagavatula, A. Jaech, U. Prasad, and M. Savvides, “Gait-ID on the move: Pace independent human identification using cell phone accelerometer dynamics,” in 2012 IEEE Fifth International Conference on Biometrics: Theory, Applications and Systems (BTAS), Sep. 2012, pp. 8–15, doi: 10.1109/BTAS.2012.6374552.

[3] M. B. Crouse, K. Chen, and H. T. Kung, “Gait Recognition Using Encodings With Flexible Similarity Measures,” undefined, 2014. https://www.semanticscholar.org/paper/Gait-Recognition-Using-Encodings-With-Flexible-Crouse-Chen/435e1cde64de351f0e169cd0f68c6a1e26c5a589 (accessed Oct. 11, 2020).

[4] P. Musale, D. Baek and B. J. Choi, "Lightweight gait based authentication technique for IoT using subconscious level activities," 2018 IEEE 4th World Forum on Internet of Things (WF-IoT), Singapore, 2018, pp. 564-567, doi: 10.1109/WF-IoT.2018.8355210.

[5] W. Xu, Y. Shen, Y. Zhang, N. Bergmann and W. Hu, "Gait-Watch: A Context-Aware Authentication System for Smart Watch Based on Gait Recognition," 2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI), Pittsburgh, PA, 2017, pp. 59-70.

[6] Handley, E., 2020. More than a quarter of Victoria coronavirus patients not at home when doorknocked by ADF. ABC, [online] Available at: <https://www.abc.net.au/news/2020-07-31/one-in-four-not-home-covid19-positive-adf-door-knock/12511682> [Accessed 15 October 2020].

[7] Suwannapong C, Khunboa C. Congestion Control in CoAP Observe Group Communication. Sensors (Basel). 2019 Aug 5;19(15):3433. doi: 10.3390/s19153433. Erratum in: Sensors (Basel). 2019 Oct 11;19(20): PMID: 31387304; PMCID: PMC6696228.

[8] Boucadair M, Shallow J. Constrained Application Protocol (CoAP) Block-Wise Transfer Options for Faster Transmission. 2020 Internet Engineering Task Force. <https://tools.ietf.org/id/draft-bosh-core-new-block-03.html >