CG4002 Embedded System Design Project

August 2021 semester

"Dance Dance"
Design Report

Group **16**	Name	Student #	Specific Contribution
Member #1	Xie Yaoyue	A0205294H	Hw Sensors
Member #2	Marcus Ng Yong	A0201743R	Hw FPGA
Member #3	Koh Rui Wen Tammy	A0202616R	Comms Internal
Member #4	Manuntag Manuel II Papa	A0202130J	Comms External
Member #5	Wang Zihao	A0204706M	Sw Machine Learning

For our group, since we only have 5 members, no one is assigned to do the dashboard design. For the design report, Section 1 System Functionalities, Section 2 Overall System Architecture and Section 6 Project Management Plan are written by all members together. Section 3.1 Hardware Sensors is written by Yaoyue. Section 3.2 Hardware FPGA is written by Marcus. Section 4.1 Internal Communications is written by Tammy. Section 4.2 External Communications is written by Manuel. Section 5.1 Software Machine Learning and Appendix are written by Zihao.

Section 1 System Functionalities [All members]

The ultimate design of the system is to classify 8 different dance moves and identify the relative positions of 3 different dancers. The details of the system functionalities are illustrated below in the format of feature lists and user stories.

Feature Lists

- The system will be able to differentiate the different dance moves.
- The system will be able to evaluate the level of synchronization.
- The system will be able to identify the position of the dancers.
- The system is user-friendly and easily operable.

User Stories

As a:	**I want:**	**So that:**
Dance Coordinator	To know how synchronised the dancers are	I can instruct the dancers on how to improve their timing
Dance Coordinator	To receive data on the synchronization of dancers as quickly as possibly	I can give prompt feedback to the dancers
Dancer Coordinator	To know the position of the dancers	I can give feedback to the correct dancers
Dancer Coordinator	To know the dance movement that is performed by the dancer	I can give feedback to the dancers regarding the specific dance movement
Dancer	The positions of sensors to be not in the way	I can do the dance without being restricted
Dancer	The sensors to be attached are secure	I don’t need to worry about sensors falling off

Dancer	To know if I’m doing the move right	I can be corrected if I’m wrong

Section 2 Overall System Architecture [All members]

2.1 High level system architecture

The overall system architecture and high level system architecture are shown in Figure 1 and 2 respectively.

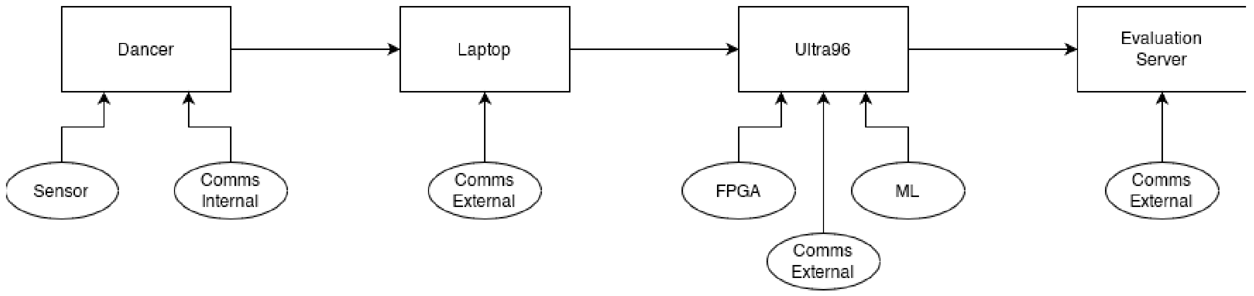


Figure 1. Overall system architecture

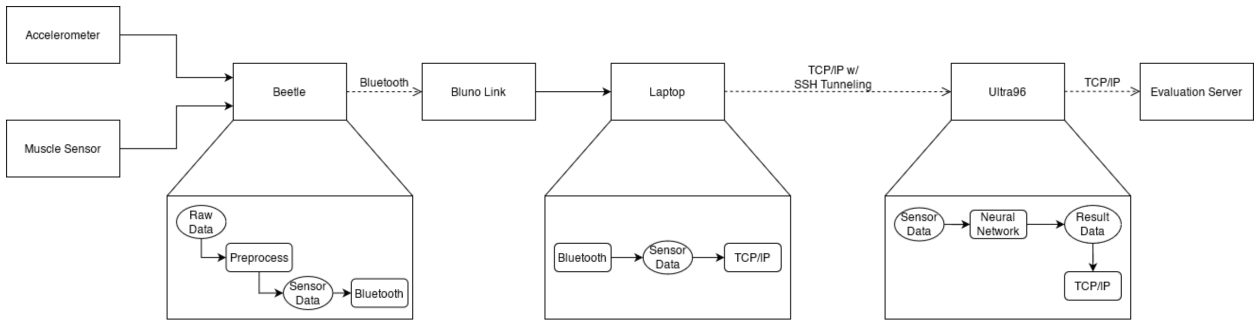


Figure 2. High level system architecture

The beetle will take in motion data from the accelerometer and gyroscope and preprocess the motion data. The beetle will establish a bluetooth connection with the laptop and send the motion data in packets following the BLE Protocol Stack to the laptops (Bluno link) along with timestamps (for synchronization).

The laptop will establish a TCP/IP connection with the Ultra96 with SSH Tunneling. The laptop will add delay information to the timestamps (for NTP), encrypt and encode the motion data and send them all to the Ultra96.

The Ultra96 will establish a TCP/IP connection with the evaluation server. The Ultra96 will run the Neural Network to evaluate the motion data and predict the dance move. The Ultra96 will calculate the synchronization delay from the information provided by the laptop and the beetle. The dance move will be encrypted and encoded along with the synchronization delay and then sent to the evaluation server where it will be evaluated.

2.2 Final form of the system

Since all dance movements are symmetrical on both left and right sides of the body, we shall detect the dance movements by capturing the data from the left side of the body. Hence, for each dancer, 3 x 1.5V batteries + 1 voltage regulator + 1 DFR0339 + 1 MPU6050 + 4 x 1.5V batteries will be attached onto the upper left arm of the dancer through an arm band, while 3 x 1.5V batteries + 1 voltage regulator + 1 DFR0339 + 1 MPU6050 + 4 x 1.5V batteries will be attached onto the left thigh of the dancer through a leg band.

One of the DFR0339 at the upper left arm of a dancer will be connected to AT-04-001 to detect the fitness level of the dancer. This is because all the dance moves require arm movements, hence the fatigue level of the dancer at the upper arm would be more noticeable. With the increased fatigue level of the dancer at the arms, the EMG signal would see an increase in its amplitude and decrease in its frequency.

2.3 Main algorithm for activity detection

Below is the brief summary of the major steps algorithm. The details of each step will be illustrated in the sections below. The collaboration diagram is shown above in Figure 2.

1. Motion and muscle data from gyroscope and accelerometer is transmitted to the Beetle.
2. The Beetle takes in the data through reading the analog input and Arduino's built-in "Wire.h" library.
3. The Beetle preprocesses the data and transmits through bluetooth to the laptop.
4. The laptop will encrypt then encode the motion data before transferring to Ultra96 through TCP/IP and SSH tunnelling.
5. Ultra96 evaluates the motion data and uses its previously trained machine learning model to predict the dance move based on the data. The detailed explanation of how the training data is gathered and used is illustrated in section 5.1.4 model training, validation and testing.
6. The muscle data and the model's prediction of the dance move will be encrypted, encoded and sent to the evaluation server through TCP/IP.

Section 3 Hardware Details

Section 3.1: Hardware sensors [Yaoyue]

3.1.1 Components / Devices

The following components / devices will be used:

- Amazon Basics AA Performance-Capacity 2,000 mAh Rechargeable Batteries
- 18650 Charging Module[1]
- DFR0339 Bluno Beetle[2]
- MPU6050 GY-521 Triple Axis Accelerometer[3]
- AT-04-001 MyoWare™ Muscle Sensor[4]

The 18650 charging module will take in voltage input from batteries totalling a voltage of 3V (2 x 1.5V Amazon Basics AA Batteries) and then supply a steady 5V to DFR0339. This particular charging module is chosen as it not only serves as a voltage regulator, but it also offers a circuit board protection feature and LED digital display power supply, which will be helpful for our project. Each DFR0339 will then be connected to one MPU6050. One of the DFR0339 will also include AT-04-001 to collect the muscle data. DFR0339 will obtain sensor data from MPU6050, and transmit the data through bluetooth to the laptop.

[1] *2.21SG$ 20% OFF|LED Dual USB 5V 2.4A Micro/Type C USB Mobile Power Bank 18650 Charging Module Lithium Battery Charger Board Circuit Protection|Integrated Circuits|—AliExpress*. (n.d.). Aliexpress.Com. Retrieved September 4, 2021, from //www.aliexpress.com/item/1005002133023221.html?src=ibdm_d03p0558e02r02&sk=&aff_platform=&aff_trace_key=&af=&cv=&cn=&dp=

[2] *DFR0339 DFRobot Datasheet.* DFRobot. https://www.application-datasheet.com/pdf/dfrobot/dfr0339.pdf

[3] *MPU6050.* http://www.haoyuelectronics.com/Attachment/GY-521/mpu6050.pdf

[4] *3-lead Muscle / Electromyography Sensor for Microcontroller Applications.* Myoware. https://cdn.sparkfun.com/datasheets/Sensors/Biometric/MyowareUserManualAT-04-001.pdf

3.1.2 Schematic

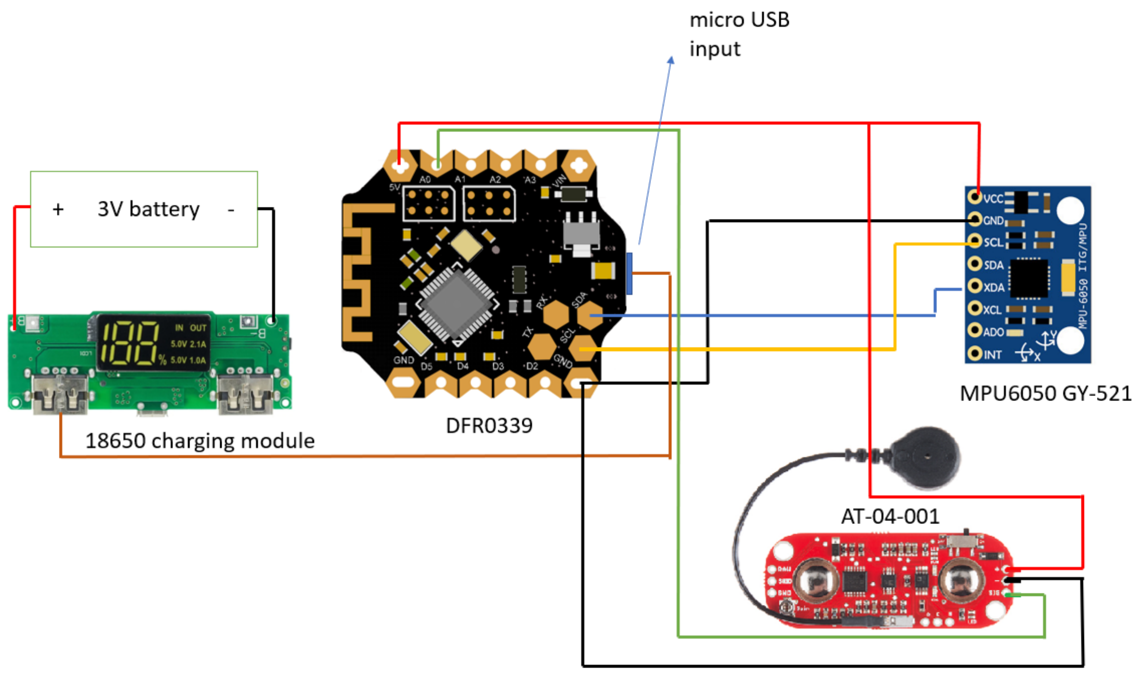


Figure 3. Schematic

3.1.3 Pin table (DFR0339 and MPU6050)

DFR0339	MPU6050	AT-04-001
5V	VCC	+
SDA	SDA	
SCL	SCL	
GND	GND	-
A0		SIG

3.1.4 Operating voltage

The operating voltage of DFR0339 is 5V, while the operating current is unspecified. Taking the maximum current output of a laptop USB port as a reference, we assume that the maximum operating current of DFR0339 is 1A.

The operating voltage of MPU6050 is between 2.375V to 3.46V. However, as the breakout board has a voltage regulator, we can connect the board directly to the 5V source from DFR0339. We do not need to consider the input current in this case.

The operating voltage of AT-04-001 is between 2.9V and 5.7V, while the operating current is unspecified. We can connect the board directly to the 5V source from DFR0339.

3.1.5 Algorithms / Libraries to use

We make use of the Arduino platform's in-built library (Wire) to establish an I2C connection between the Beetle and MPU6050. Using this library, we can obtain the gyroscope and accelerometer sensor values[5]. The muscle sensor values can be read directly from the analog input. We can detect the start of a move through the accelerometer values. At rest, the sensor value of the accelerometer would be approximately 1g at the z-axis while that at the x and y axes would be approximately 0. This would change when there is a move due to acceleration brought by the movement. Processing of the raw data could be explored during the project.

Section 3.2: Hardware FPGA [Marcus]

3.2.1 Ultra96 synthesis and simulation

The synthesis will be set up for the target device, xczu3eg-sbva484-1-i, which is the part on the board Ultra96 v2. Synthesis will be done once in Vitis High Level Synthesis (HLS). HLS will synthesize the C code which contains the neural network algorithm into a Register Transfer Level (RTL) implementation and package it into an Intellectual Property (IP) for use in Vivado Design Suite. It is also possible to run simulation in HLS with C/RTL Cosimulation to verify that the RTL is functionally identical to the C source code. Subsequently, synthesis and simulation will be done in Vivado Design Suite. Synthesis transforms the RTL design into a gate-level representation. Simulation in Vivado Design Suite allows one to perform behavioral, functional, and timing simulations, before generating the bitstream that programs the FPGA on the Ultra96.

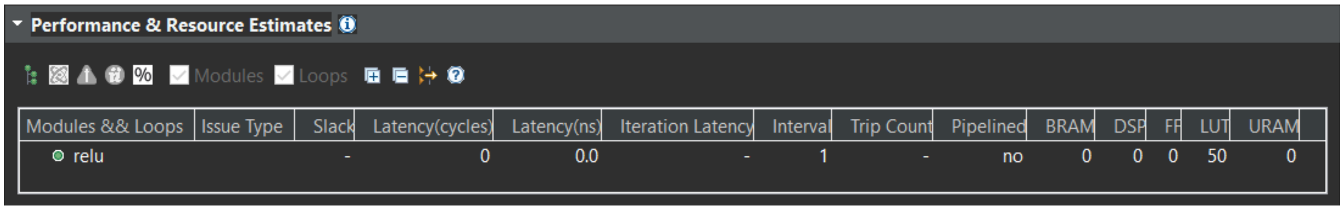
[5] Tutorial: How to use the GY-521 module (MPU-6050 breakout board) with the Arduino Uno. (2017, October 5). *Michael Schoeffler*. https://mschoeffler.com/2017/10/05/tutorial-how-to-use-the-gy-521-module-mpu-6050-breakout-board-with-the-arduino-uno/

3.2.2 How the neural network model will be realized on the FPGA board

All values will be in floating point representation, to avoid analyzing the minimum number of bits needed to maintain enough precision for the input data, weight, bias and output data. After the neural network has been trained, the final weight and bias values can be extracted and stored. The layers of the neural network (input, hidden and output) will be coded in C or C++, referencing the weight and bias values extracted. The activation functions can be resource intensive to calculate or generate, thus using a Lookup Table to reference the pre calculated values is much more efficient. The C/C++ code can be synthesised in Vitis HLS into an IP that can be connected to the FPGA IP in the IP integrator in Vivado Design Suite. Simulations can then be run to test the functionality and whether the neural network on the FPGA has output (accuracy) which differs from the trained neural network on the computer. After the simulations are accurate, the bitstream can be generated to program the FPGA board with the neural network using the Overlay class in PYNQ (Python productivity for Zynq).

3.2.3 Design's timing, power and area

The design's timing performance can be seen in the report timing summary in Vivado Design Suite. This allows the maximum and minimum delay for the neural network's output to be measured. Design's power can be measured by monitoring the powerlines, this can be achieved by running the PYNQ script in the Jupyter Notebook on the Ultra96. The design's area can be measured in terms of the number of FFs and LUTs used on the FPGA board. This can be seen in the utilization estimates from synthesis which generates the FFs and LUTs used, as shown below in Figure 4.



Modules && Loops	Issue Type	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
relu		-	0	0.0	-	1	-	no	0	0	0	50	0

Figure 4. Utilization estimates

3.2.4 Potential optimization of the neural network accelerator

Floating point representation for the values can be resource intensive, hence using fixed point conversion processes where appropriate can reduce the resource usage. With this it is possible to achieve better efficiency without sacrificing the accuracy of the neural network.

3.2.5 Power management of Ultra96 CPU + FPGA

There are certain peripherals that can be configured to be switched off to avoid power consumption as these peripherals will never be used. Some of the peripherals include display port, UART, LEDs and the USB ports. The change in power consumption can be monitored through PYNQ.

Section 4 Firmware & Communications Details

Section 4.1 Internal Communications [Tammy]

4.1.1 Time-Oriented Task Manager

Tasks on Beetle would be managed sequentially as per the code. Steps that carry essential information would be able to issue an interrupt which would be processed once the current step has been executed. The interrupt would then call the next function and the Beetle will continue to process the steps sequentially.

4.1.2 Setup And Configuration of BLE Interface

The Beetle would be configured through AT commands on arduino. These AT commands would enable the user to set up the Beetle for bluetooth communications. The Beetle would be set to be the peripheral device while the laptop would be the central device. The BLE host and controller would be set on Ubuntu Linux possibly using the lines of code in the diagram below.

```
> hciconfig
> hciattach /dev/ttyPS1 -t 10 any 115200 noflow nosleep
```

4.1.3 Protocol Over BLE

This process would begin from the laptop. It would first discover the Beetles and once it has done so, it will then establish a connection by initiating the handshaking. The initial handshake will be a three-way handshake in order to ensure that connection is secure.

Currently, the table below displays some of the basic packet types that would be implemented.

No.	Packet Type	Detail
1	ACK	Acknowledges that it has received a packet
2	Read	Reading data from destination
3	Write	Writing data from destination
4	Wave	To initiate a connection

Each packet type would have an index which would be included in the packet format to indicate the type of packet being sent. The BLE Packet Format would begin with a preamble. Next, it will be followed by an access address, then the protocol data unit, and finally the cyclic redundancy check. Finally, the baud rate that is going to be used would be 115200.

4.1.4 Managing Reliability

During communications, reliability of data would be handled using checksum and acknowledgements. If no data is being received after a period of time, the laptop would send polls to ensure that connection is still secure.

In the event the Beetle moves out of range, there will be a loss of connection between the Beetle and the laptop. If this happens, the laptop would periodically look for the Beetle to re-establish a connection. Once discovering the Beetle, it would initiate a 3-way handshake with the Beetle again. This also would work if the Beetle has been switched off and turned on again.

Section 4.2 External Communications [Manuel]

4.2.1 Communication between Ultra96 and evaluation server

Python's Socket is a low-level programming module specially used for networking purposes like creating TCP/IP servers or clients. The server will be the provided evaluation server and the client will be the Ultra96. Since both the Evaluation server and the Ultra96 are inside the NUS Firewall, there is no need to do SSH tunneling.

The message format will follow the evaluation server's chosen format of:

```
position, 1 action, sync
E.g.: “1 2 3|mermaid|3.0250000000000000”
```

The positions part will be one of the following strings according to the out of the Neural Network process:

```
POSITIONS = ['1 2 3', '3 2 1', '2 3 1', '3 1 2', '1 3 2', '2 1
3']
```

The action part will be one of the following strings according to the out of the Neural Network process:

```
ACTIONS = ['mermaid', 'jamesbond', 'dab', 'window360',
'cowboy', 'scarecrow', 'pushback', 'snake']
```

The sync part will be the synchronization time between the 3 dancers according to the calculations based on Network Time Protocol (NTP). NTP can be used to synchronize all participating computers to within a few milliseconds of Coordinated Universal Time (UTC).

TCP provides reliable, ordered, and error-checked delivery of a stream of octets (bytes) between applications running on hosts communicating via an IP network. In order to ensure proper communication between the server and the client, the encryption protocol used needs to be the same on both ends. The evaluation server uses Advanced Encryption Standard (AES) that is provided by Python's Crypto.Cipher module. In this process a key needs to be provided (secret_key in eval_server.py) and an Initial Vector (iv in the code) needs to be generated in the Client and then sent to the Server with the cipher text. The evaluation server uses Cipher block chaining (CBC) and so will the client.

```
AES.new(key, AES.MODE_CBC, iv)
```

Once encrypted, the iv will be added in front of the cipher_text and then encoded using base64 encoding provided by Python's base64 module. Finally, this encoded message will be sent through the internet via TCP/IP. The server will then decode the encoded message, get and use the iv and the secret_key to decrypt the message and parse the information accordingly. Overall, the Python modules used are: os, sys, random, time, socket, base64, and Crypto.Cipher.

4.2.2 Communication between laptops and Ultra96

Python's Socket is a low-level programming module specially used for networking purposes like creating TCP/IP servers or clients. The server will be the Ultra96 and the client will be the Laptop with the Bluno Link. Since the Ultra96 is inside the NUS Firewall, SSH tunneling is required to be performed by the server and the client.

The message format will follow the evaluation server's chosen format of:

```
dancer number, motion data, sync
```

The dancer number will be the identification of the source of the motion data. The motion data will be from the accelerometers and other sensors sent to the laptop via bluetooth.

The sync part will be the relative time difference between the 3 dancers according to the calculations based on Network Time Protocol (NTP). NTP can be used to synchronize all participating computers to within a few milliseconds of Coordinated Universal Time (UTC).

TCP provides reliable, ordered, and error-checked delivery of a stream of octets (bytes) between applications running on hosts communicating via an IP network. In order to ensure proper communication between the server and the client, the encryption protocol used needs to be the same on both ends. The server uses Advanced Encryption Standard (AES) that is provided by Python's Crypto.Cipher module. In this process a key needs to be provided (secret_key in eval_server.py) and an Initial Vector (iv in the code) needs to be generated in the Client and then sent to the Server with the cipher text. The server uses Cipher block chaining (CBC) and so will the client.

```
AES.new(key, AES.MODE_CBC, iv)
```

Once encrypted, the iv will be added in front of the cipher_text and then encoded using base64 encoding provided by Python's base64 module. Finally, this encoded message will be sent through the internet via TCP/IP. The server will then decode the encoded message, get and use the iv and the secret_key to decrypt the message and parse the information accordingly. Overall, the Python modules used are: os, sys, random, time, socket, base64, and Crypto.Cipher.

4.2.3 Network Time Protocol for Dance Synchronization

Network Time Protocol (NTP) can be used to synchronize all participating computers to within a few milliseconds of Coordinated Universal Time (UTC). Using a hierarchical, semi-layered system, the laptops will be synchronized over a network to stratum 1 servers (NTP servers). The beetles will be synchronized to the stratum 2 servers (Laptops). Due to the slower bluetooth connection between the beetles and the laptops, calibrations (clock offset) will be done to compensate by setting benchmarks beforehand. In case of Stay at home Notice, the laptops connection speeds to the Ultra96 will also be benchmarked and necessary calibrations (clock offset) to the clock synchronization algorithm will be made. Sensors will send a start packet to mark the beginning of the dance move. Local timestamps will accompany the motion data to the Ultra96 in order to calculate relative time difference between packets from different dancers. With the relative time difference, clock offset and the connection delay, actual time difference can be calculated for synchronization delay.

Section 5 Software Details

Section 5.1 Software Machine Learning [Zihao]

In this section, we have done an analysis based on an online dataset to assist our design. The dataset used is Smartphone and Smartwatch-Based Biometrics Using Activities of Daily Living (Weiss, Yoneda, & Hayajneh, 2019). Weiss etc. conducted the experiment by allocating accelerometers and gyroscopes on wrist and waist separately. In total, 4 sensors (i.e., 2 accelerometers and 2 gyroscopes) are used, which is similar to our design. We have done a three-label classifications (i.e., three different activities namely walking, jogging and typing) on this dataset using the two accelerometer readings of one participant. We have tested the classification performance using k-nearest neighbors (kNN), support vector machines with linear kernel and multilayer perceptrons on this dataset. The results of data processing and model performance on this dataset will assist us in making decisions on our design.

5.1.1 Data segmentation

Since the data will be sent to the software continuously, it is crucial to choose the appropriate sliding window size to analyze the incoming data. If the window size is too small, insufficient information will be collected for data analysis, leading to inaccurate classification results. If the window size is too large, too much information will be passed in for classification and there may be multiple movements or events occurred during the chosen window, leading to high runtime and incorrect classification results.

Currently, we have decided to set the sensor sampling rate at 20Hz. Each sensor will output the data in three axes (i.e., X, Y and Z) and there are four sensors (i.e., 2 accelerometer and 2 gyroscopes) used per participant, which means that there will be 240 data points collected per second per participant and 12 features in total. Moreover, we observe that all dance movements can be finished within or around 5 seconds. Thus, we have decided to use the window size of 5 seconds and make the prediction based on the 5-second data, which is 1200 data points. However, it may not be the optimal window size. The sliding window size will be adjusted during the experimental process.

The start and the end of the event can be observed from the sudden change of the sensor values. Different events have different intensity, leading to different extent of variation on the reading of the vertical axis (see Appendix Figure 1-6). Thus, we can define a threshold of variation to differentiate different activities.

5.1.2 Data processing and feature extraction

There are two options for us to extract useful information from the raw data, namely data processing and feature extraction. We will test these two options on our design and derive the optimal algorithm once the actual data is collected.

We have done data processing on the online dataset and the data processing technique that we use works on that particular dataset. The actual effectiveness of this data processing technique will be tested on our design once the hardware is built and data is collected. We will apply data transformation on the raw data. Firstly, we pass the raw data into a fifth order butterworth bandpass filter in order to extract useful information from a particular frequency band. Secondly, the results will be squared in order to increase the difference in the intensity of different activities for easier differentiation (i.e., smaller values get smaller and larger values get larger, leading to a more obvious gap among different activities). Thirdly, we will apply a Savitzky-Golay filter to smooth the data. Fourthly, the results will be standardised before passing into the machine learning model. The results will be scaled into the range of 0 to 1, which will ensure data in all dimensions have the same scale. This will ensure all features have the same influence on the final classification outcome. The overall flow is summarised as 5th order butterworth bandpass filter → squaring the results → Savitzky-Golay filter → normalisation.

If the above data processing algorithm does not work on the actual data, we will perform feature extraction on the raw data to obtain effective and efficient information. The features can be classified into three different domains, namely time domain features, frequency domain features (i.e., after fast Fourier Transform) and time frequency domain features (i.e., after Discrete Wavelet Transform). Simão etc. have proven four features that will perform well for classification, namely mean absolute value, waveform length, Willison amplitude and auto-regressive coefficients (Simão, Mendes, Gibaru, & Neto, 2019). We will apply these four features on our actual data to test out the performance and verify their effectiveness. Moreover, if the runtime is too long, we will apply dimensionality reduction techniques such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) to reduce the dimension of the matrix to reduce the runtime. PCA focuses on the data with the most variation while LDA focuses on maximising the separability among known categories. The final decision will be made after testing on the actual data.

5.1.3 Potential machine learning models

We have planned to implement three potential machine learning models to classify the dance movements, namely k-nearest neighbours (kNN), support vector machine with linear kernel and multilayer perceptrons. There are two criteria for us to decide the appropriate machine learning model, namely classification accuracy and runtime. We have performed the data processing algorithm described in 5.1.2 (see Appendix Figure 7) and tested the performance of these models on the online dataset and the accuracy is very high (see Appendix Figure 8, 9, 10 and 11). The kNN that we used makes use of Euclidean distance and the k value is set to 3. The multilayer perceptron that we used has a hidden layer size of (1, 1, 1) and a maximum iteration of 1000. The very high accuracy of these three models may be due to the straightforward classification task, which is only a three-label classification on three easily differentiable activities (i.e., walking, jogging and typing) with a small data size.

In addition, kNN will not be a suitable model as the runtime is relatively long. It has to loop through all examples in the dataset to locate the k nearest neighbours. We have implemented the kNN algorithm from scratch using only the numpy library. Even though the accuracy is high, the runtime is extremely long (i.e., for the three-label classification, it takes around 145.4 seconds). Thus, it is not feasible for kNN to give real-time responses to the user regarding the classification results as it is not user-friendly. Hence, the final decision of the model choice will be made between multilayer perceptron and support vector machine with linear kernel after testing on the actual data.

5.1.4 Model training, validation and testing

Our group has 5 members and we all will be involved in the data collection process. Each of us will perform a 1-minute dance for all 8 dance moves. Since the sampling rate of the sensors is 20Hz and 4 sensors are used per participant (i.e., each sensor will output data of three axes), in total around 2304000 data points will be collected for model training, validation and testing.

At first, data of 8 movements from 4 people will be used to train and validate different machine learning models. We will split the data into a training set (i.e., 75%) and a validation set (i.e., 25%). Then, we will fit data from the training set into the machine learning models and generate a confusion matrix, classification report and runtime analysis using data from the validation set to compare the performance of different machine learning models. After that, we will use data of 8 dance moves from the remaining one person as the unseen data to test different machine learning models (i.e., as the test set and a trial run). Similarly, we will generate a confusion matrix, classification report and runtime analysis using data from the test set to compare the performance of different machine learning models. Upon selecting the most effective and efficient model, we will combine all data together (i.e., 5 people of 8 dance moves) to form a full dataset and retrain the selected machine learning model. Finally, we will save the latest model and update the weights and bias.

In general, we will use machine learning models from scikit-learn library first. If the results are promising, we will implement the model from scratch using only the numpy library.Then we will train and save the model, weights and bias and pass them to the member who is in charge of FPGA programming.

Section 6 Project Management Plan [All members]

Week	Task to do	Members involved
4	Design Report	All
	Solder Work	Yaoyue
	Try and error for several data processing techniques and machine learning models on an online dataset	Zihao
	Setup Vivado Design Suite and Vitis HLS for the Ultra96 v2	Marcus

	Establish bluetooth connection between Beetle and laptop	Tammy
	Establish Client - Eval Server connection using TCP/IP	Manuel
5	Try and error for several data processing techniques and machine learning models on an online dataset	Zihao
	Send packets through bluetooth	Tammy
	Set up a cryptography standard for all Clients and Servers	Manuel
	Connect the IP developed in HLS to FPGA in Vivado Design Suite	Marcus
	Get all hardware sensors to connect with Beetles	Yaoyue
6	Individual progress checkpoint	All
	Implement an appropriate neural network model from scratch using python	Zihao
	Design wearable	Yaoyue
	Research and Implement SSH Tunneling for Laptop - Ultra96 connection	Manuel
	Improve reliability of connections	Tammy
	Program FPGA with PYNQ and attempt a roundtrip data transfer between FPGA and CPU	Marcus
Recess week	Implement another machine learning model from scratch using python	Zihao
	Research and Implement NTP synchronization	Manuel
	Research methods on data-preprocessing and implement it if necessary	Yaoyue
	Ensure concurrency and security	Tammy
	Implement the neural network model in C/C++ for the HLS	Marcus

7	Individual subcomponent test and video recording of individual subcomponent	All
	Data collection (5 people with 8 dance movements)	All
	Integration of individual subcomponents to the final product	All
8	Integration of individual subcomponents to the final product	All
	Retrain the machine learning model with the actual dataset for the first three dance moves	Zihao
9 and 10	First system checkpoint (1 dancer, first 3 moves, no positions)	All
	Complete the positioning algorithm	All
11	Second system checkpoint (3 dancers, first 3 moves, positions)	All
	Peer review	All
	Retrain the machine learning model with the actual dataset for all 8 dance moves	Zihao
12	Ensure the whole system is working	All
13	Final evaluation (3 dancers, 8 moves, positions)	All
	Final design report, documentation and code submission	All

References

Base64 — Base16, Base32, Base64, Base85 Data Encodings — Python 3.9.7 documentation. (n.d.). Python.Org. Retrieved September 4, 2021, from https://docs.python.org/3/library/base64.html

Crypto.Cipher package — PyCryptodome 3.9.9 documentation. (n.d.). Python.Org. Retrieved September 4, 2021, from https://pycryptodome.readthedocs.io/en/latest/src/cipher/cipher.html

del Toro, S. F., Santos-Cuadros, S., Olmeda, E., Álvarez-Caldas, C., Díaz, V., & San Román, J. L. (2019). Is the Use of a Low-Cost sEMG Sensor Valid to Measure Muscle Fatigue? *Sensors (Basel, Switzerland)*, *19*(14), 3204. https://doi.org/10.3390/s19143204

GeeksforGeeks. (2021, January 19). Network Time Protocol (NTP). https://www.geeksforgeeks.org/network-time-protocol-ntp/

ntplib. (2021, May 28). PyPI. https://pypi.org/project/ntplib/

Simão, M., Mendes, N., Gibaru, O., & Neto, P. (2019). A Review on Electromyography Decoding and Pattern Recognition for Human-Machine Interaction. *IEEE Access*, *7*, 39564–39582. https://doi.org/10.1109/ACCESS.2019.2906584

Socket — Low-level networking interface — Python v3.0.1 documentation. (n.d.). Python.Org. Retrieved September 4, 2021, from https://docs.python.org/3.0/library/socket.html

Sridhar, J. (2018, February 8). Using AES for Encryption and Decryption in Python Pycrypto [Blog]. Novixys Software Dev Blog. https://www.novixys.com/blog/using-aes-encryption-decryption-python-pycrypto/#3_Initialization_Vector

Weiss, G. M., Yoneda, K., & Hayajneh, T. (2019). Smartphone and Smartwatch-Based Biometrics Using Activities of Daily Living. *IEEE Access*, *7*, 133190–133202. https://doi.org/10.1109/ACCESS.2019.2940729

Wikipedia contributors. (2021, July 22). Block cipher mode of operation. Wikipedia. https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Cipher_block_chaining_(CBC)

Appendix [Zihao]

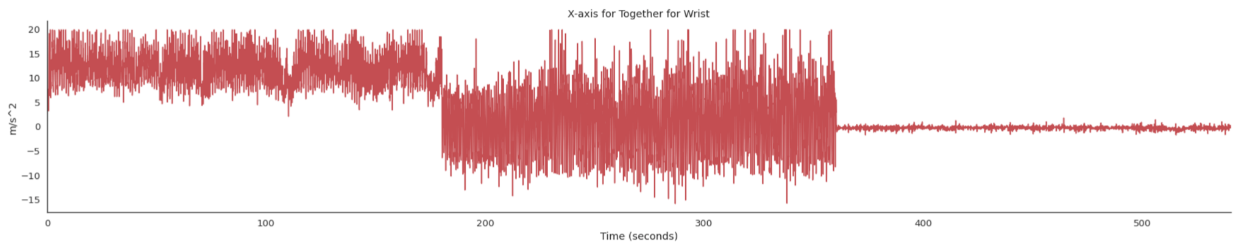


Figure 1. X-axis for accelerometer on the wrist

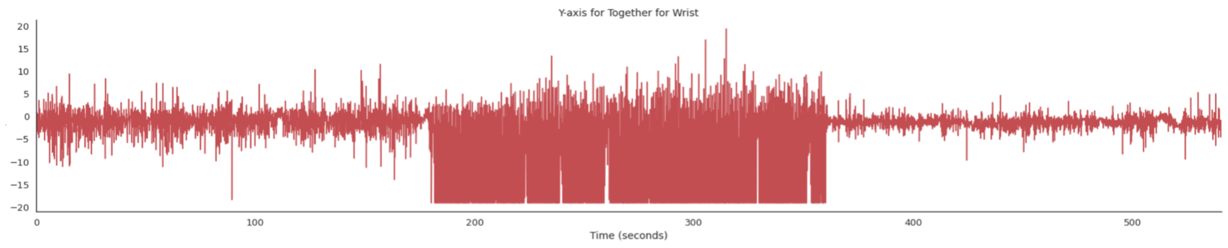


Figure 2. Y-axis for accelerometer on the wrist

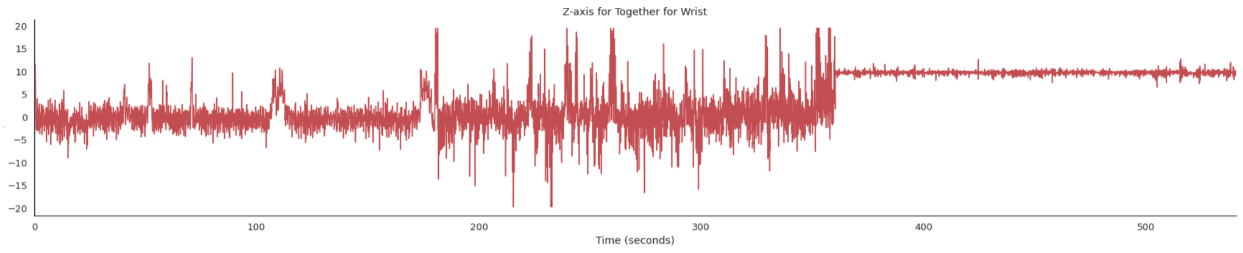


Figure 3. Z-axis for accelerometer on the wrist

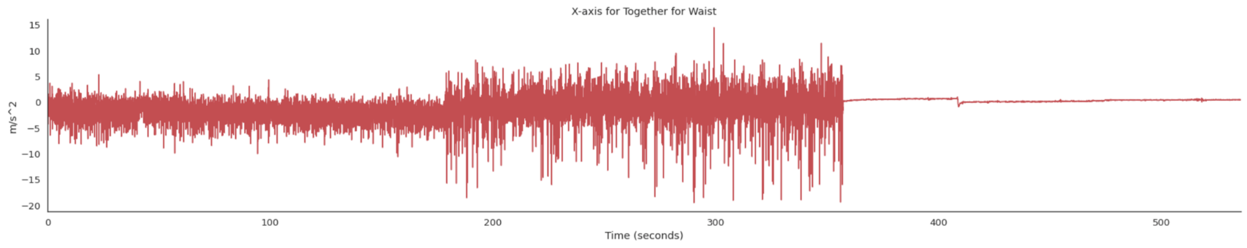


Figure 4. X-axis for accelerometer on the waist

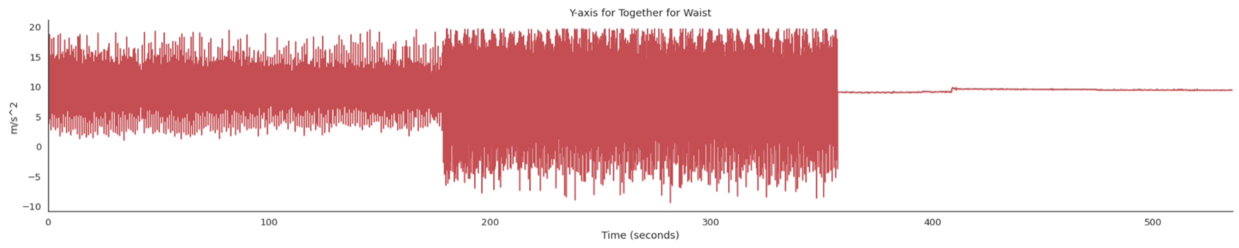


Figure 5. Y-axis for accelerometer on the waist

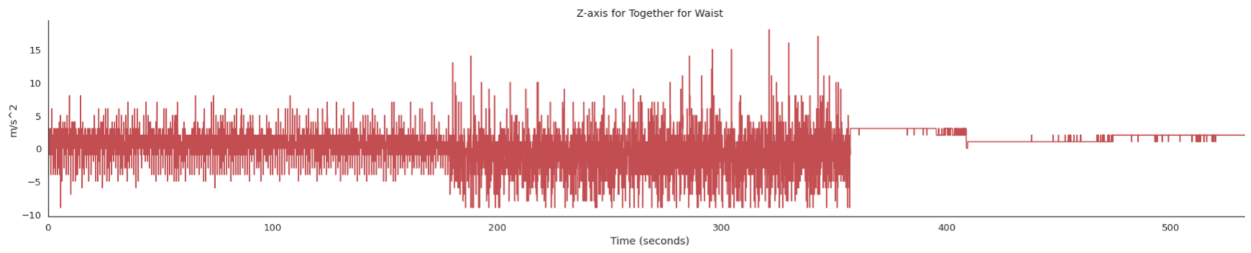
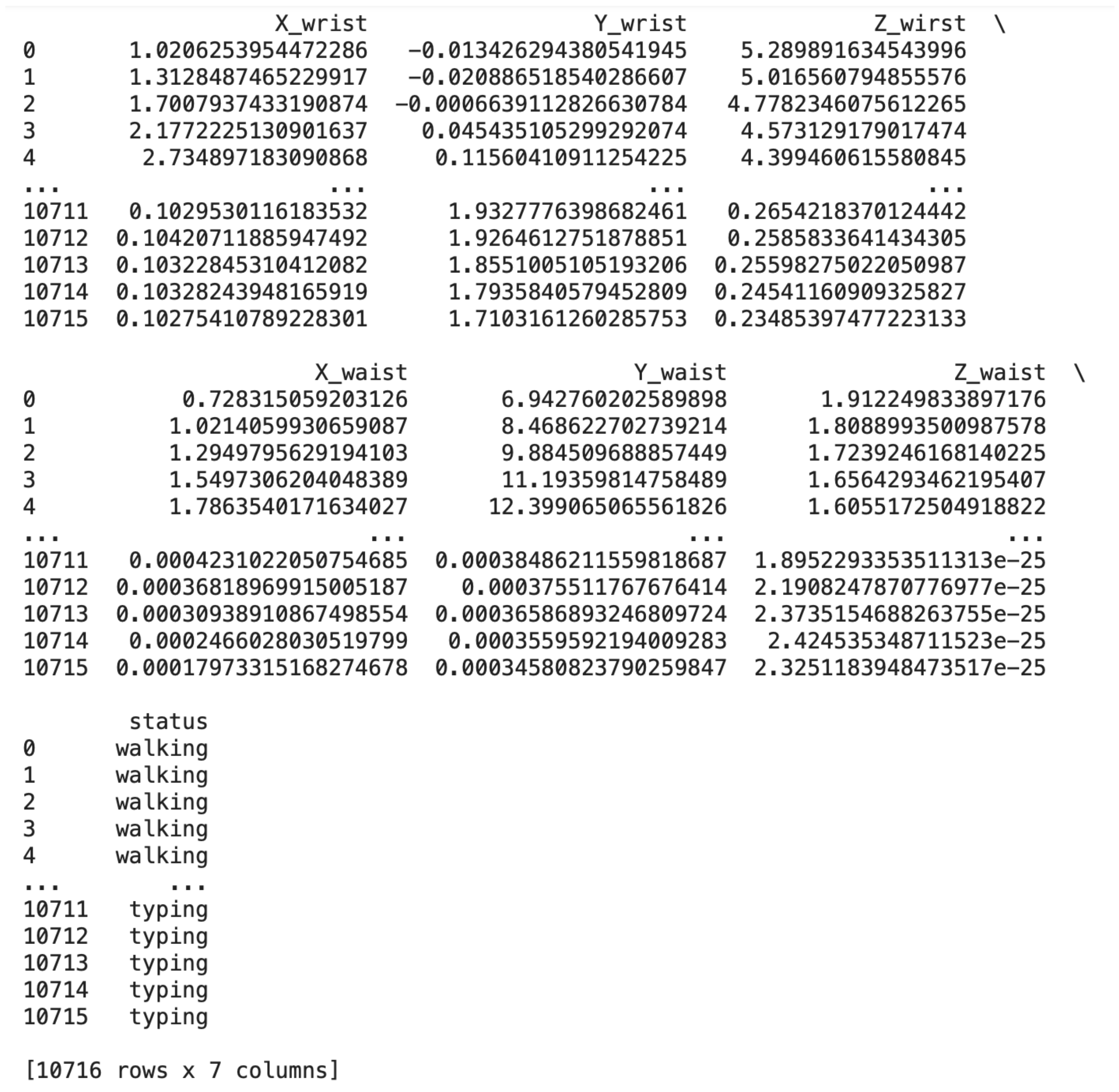


Figure 6. Z-axis for accelerometer on the waist



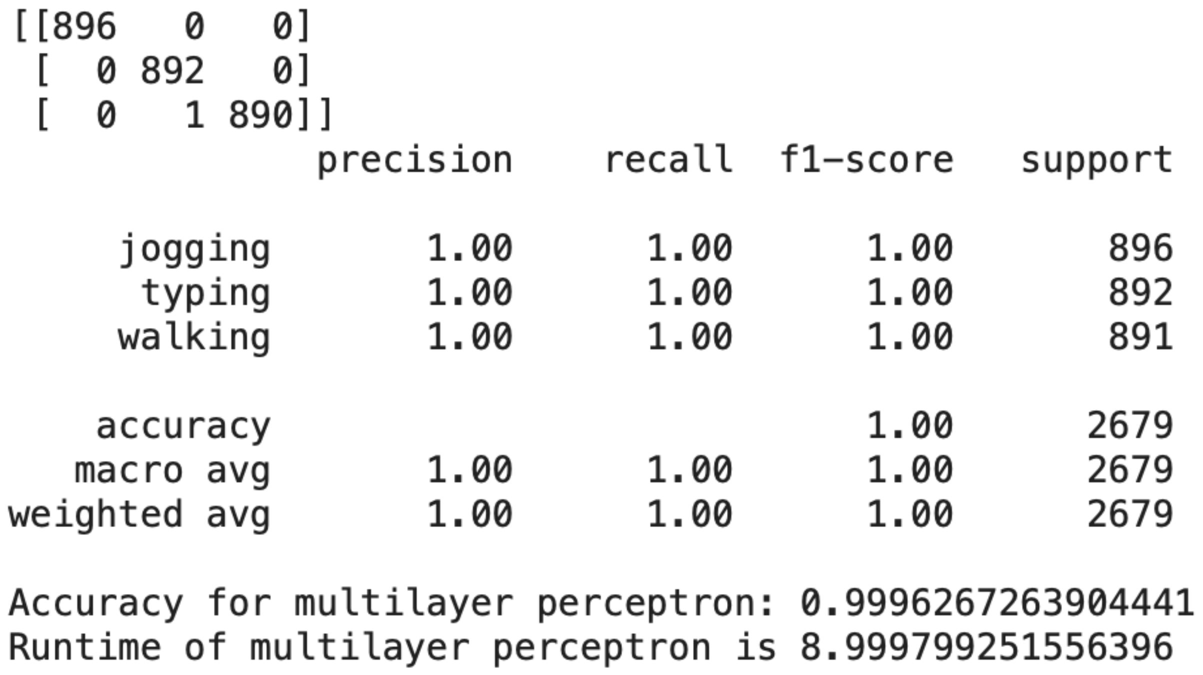
```
                X_wrist                Y_wrist              Z_wirst  \
0       1.0206253954472286   -0.013426294380541945   5.289891634543996
1       1.3128487465229917   -0.020886518540286607   5.016560794855576
2       1.7007937433190874  -0.000663911282663078 4   4.7782346075612265
3       2.177222513090163 7   0.045435105299292074   4.573129179017474
4        2.734897183090868     0.1156041091125422 5   4.399460615580845
...                    ...                     ...                 ...
10711   0.1029530116183532      1.9327776398682461   0.2654218370124442
10712  0.10420711885947492      1.9264612751878851   0.2585833641434305
10713  0.10322845310412082      1.8551005105193206  0.25598275022050987
10714  0.10328243948165919      1.7935840579452809  0.24541160909325827
10715  0.10275410789228301      1.7103161260285753  0.23485397477223133

                   X_waist                Y_waist                Z_waist  \
0          0.728315059203126      6.942760202589898      1.912249833897176
1         1.0214059930659087      8.468622702739214     1.8088993500987578
2         1.2949795629194103      9.884509688857449     1.7239246168140225
3         1.5497306204048389      11.19359814758489     1.6564293462195407
4         1.7863540171634027     12.399065065561826     1.6055172504918822
...                      ...                    ...                    ...
10711   0.0004231022050754685  0.00038486211559818687  1.8952293353511313e-25
10712  0.00036818969915005187    0.000375511767676414  2.1908247870776977e-25
10713  0.00030938910867498554  0.00036586893246809724  2.3735154688263755e-25
10714   0.0002466028030519799   0.0003559592194009283   2.424535348711523e-25
10715  0.00017973315168274678  0.00034580823790259847  2.3251183948473517e-25

        status
0      walking
1      walking
2      walking
3      walking
4      walking
...        ...
10711   typing
10712   typing
10713   typing
10714   typing
10715   typing

[10716 rows x 7 columns]
```

Figure 7. Results after data processing (before normalisation)



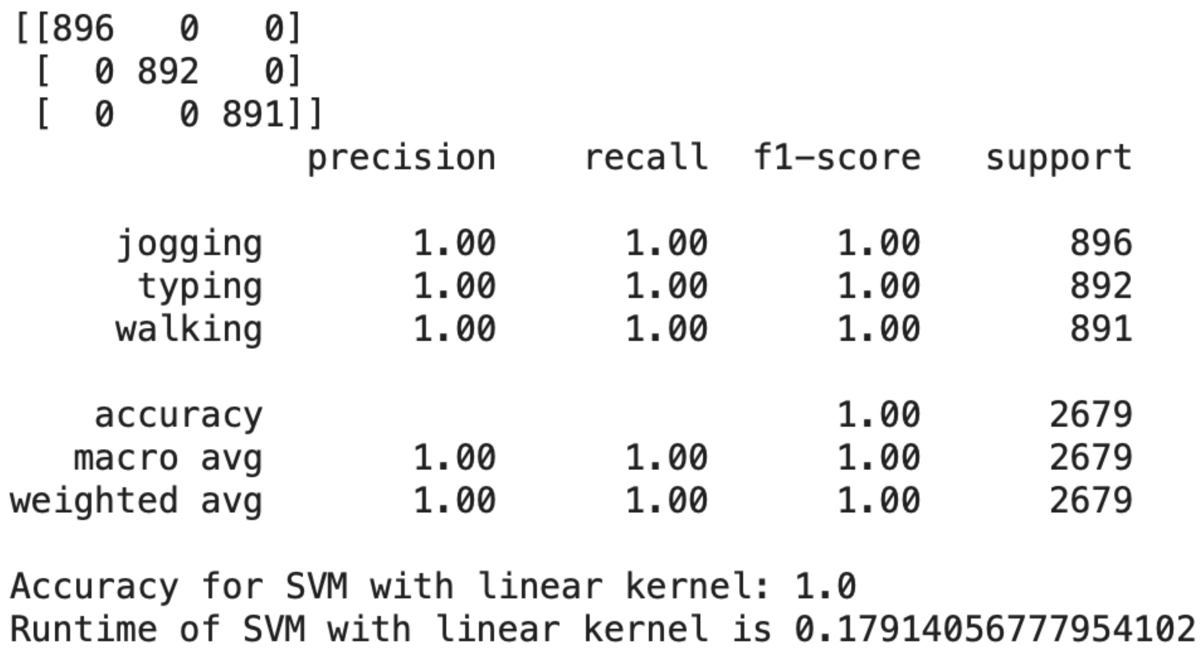
```
[[896   0   0]
 [  0 892   0]
 [  0   1 890]]
              precision    recall  f1-score   support

     jogging       1.00      1.00      1.00       896
      typing       1.00      1.00      1.00       892
     walking       1.00      1.00      1.00       891

    accuracy                           1.00      2679
   macro avg       1.00      1.00      1.00      2679
weighted avg       1.00      1.00      1.00      2679

Accuracy for multilayer perceptron: 0.9996267263904441
Runtime of multilayer perceptron is 8.999799251556396
```

Figure 8. Performance of multilayer perceptron using scikit-learn library



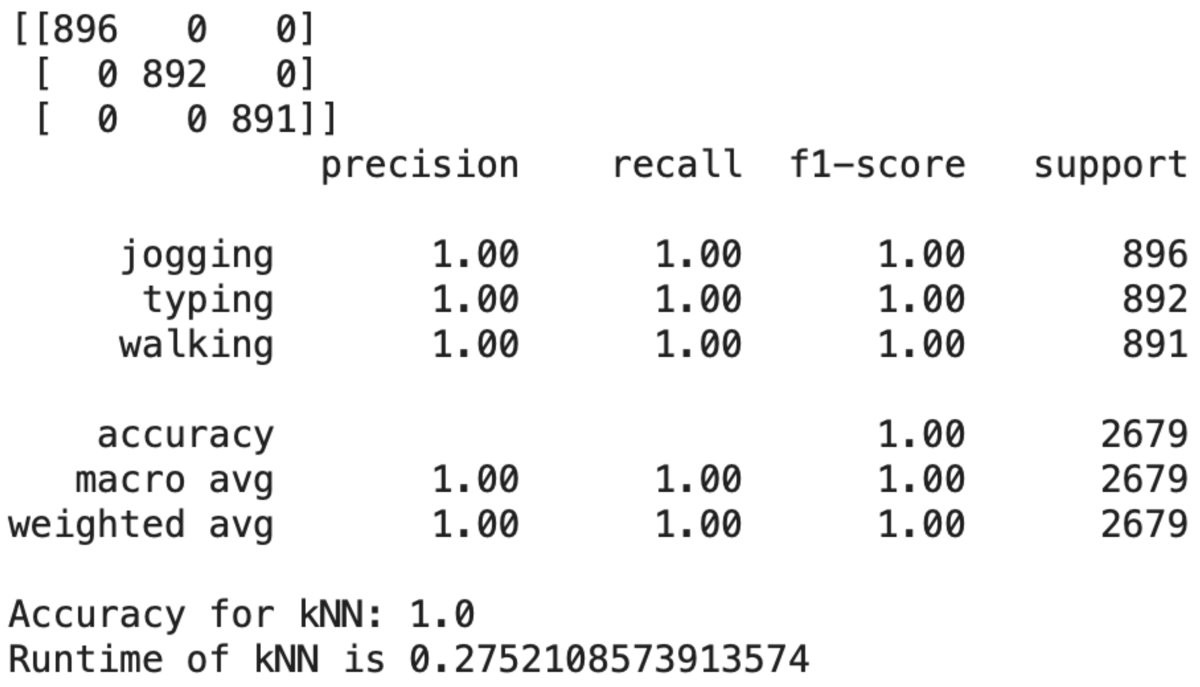
```
[[896   0   0]
 [  0 892   0]
 [  0   0 891]]
              precision    recall  f1-score   support

     jogging       1.00      1.00      1.00       896
      typing       1.00      1.00      1.00       892
     walking       1.00      1.00      1.00       891

    accuracy                           1.00      2679
   macro avg       1.00      1.00      1.00      2679
weighted avg       1.00      1.00      1.00      2679

Accuracy for SVM with linear kernel: 1.0
Runtime of SVM with linear kernel is 0.17914056777954102
```

Figure 9. Performance of SVM with linear kernel using scikit-learn library



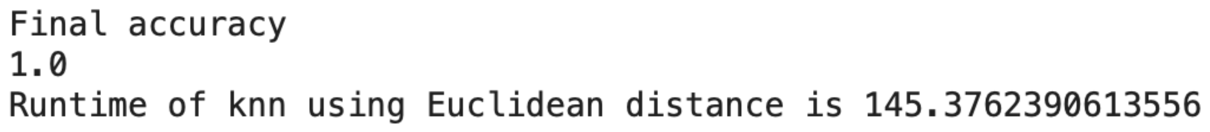
```
[[896   0   0]
 [  0 892   0]
 [  0   0 891]]
              precision    recall  f1-score   support

     jogging       1.00      1.00      1.00       896
      typing       1.00      1.00      1.00       892
     walking       1.00      1.00      1.00       891

    accuracy                           1.00      2679
   macro avg       1.00      1.00      1.00      2679
weighted avg       1.00      1.00      1.00      2679

Accuracy for kNN: 1.0
Runtime of kNN is 0.2752108573913574
```

Figure 10. Performance of kNN using scikit-learn library



```
Final accuracy
1.0
Runtime of knn using Euclidean distance is 145.3762390613556
```

Figure 11. Performance of kNN that is built from scratch