CG3002 Embedded System Design Project

2018/2019 Semester 2 (January)

"Dance Dance"

Design Report

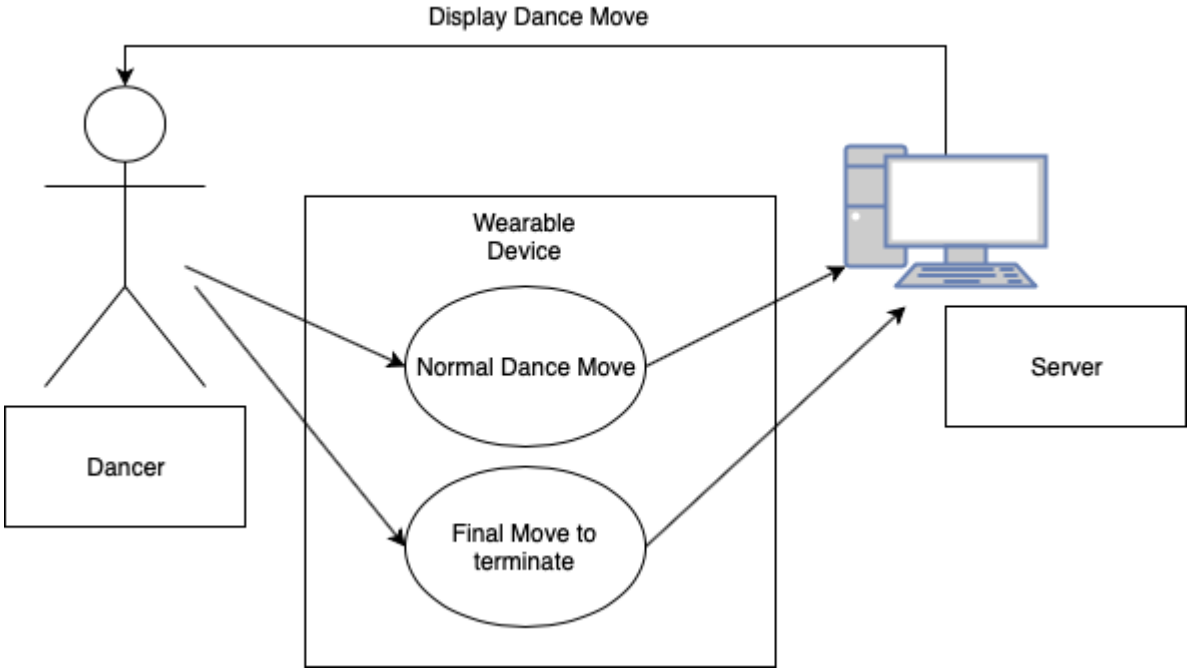
Group 3	Name	Student #	Sub-Team	Specific Contribution
Member #1	Lai Yingchen	A0146682X	Hardware	Section 2.1, 2.2, 2.3, 3.1, 3.2, 3.3, 3.4
Member #2	Goh Zu Wei	A0161897M	Hardware	Section 3.5, 3.6, 3.7
Member #3	Ng Kheng Yi	A0155159Y	Firmware	Section 4.1,4.2,4.3,6
Member #4	Nguyen Thanh Son	A0161299W	Firmware	Section 1,2.1,4.3,4.4
Member #5	Harsh Gadodia	A0135792X	Software	Section 1,5.1,5.2,5.3
Member #6	Xie Jihui	A0147969E	Software	Section 2.3, 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7

Table of Contents

Section 1 System Functionalities

This system is designed to detect discrete dance moves by a dancer wearing our self-designed suit. A dance move is shown on a screen to the user, following which it is performed by the dancer. Wearable sensors analyse the movements of the dancers, and attempt to classify the dance move as one of the 10 discrete moves (11 if you include the “final move”) shown on the screen. The data from sensors on the wearable device is sent from the Arduino Mega 2560 to the Raspberry Pi (RPi) to analyze and predict the dance moves. This information is sent to a remote server to gauge accuracy, and precision. A “final move” is used to indicate the system that the session has ended.

1.1 Use Case Diagram



1.2 Use Cases

Use Case: Valid Dance Move

1. One random dance move is displayed on the prompt screen.
2. User performs specified dance move.
3. Device attempts to identify dance move.
4. Device sends identified dance move to server.
5. Server receives the predicted dance move and continues to the next one.
6. Prompt screen displays the next specific dance move.
7. End

Use Case: Final Dance Move

1. User performs final dance move to signal the end.
2. The sensors from the user's body sent the data to the system.
3. Device attempts to identify dance move.
4. Device correctly identifies dance move as final dance moves.
5. System terminates.
6. End

Extraordinary Use Case: Idle/Fail

If system hangs or fails or unexpectedly stops working otherwise:

1. Fail gracefully
    a. Not just crash as far as possible.
2. Display warning to user that the system is not working.
3. Reboot if necessary.

1.3 Feature List

- Arduino Mega 2560:
    + Receive data and filter noises from sensors.
    + Push data to the Raspberry Pi through UART, every 500ms.
- Raspberry Pi:
    + Receive data from the Arduino through UART.
    + Clean data before sending it to the Machine Learning (ML) module.
    + The ML module returns a predicted dance move, which will be sent to the remote evaluation server.
- Other hardware:
    + A level shifter is used as a "bridge" to transfer data between the Arduino and RPi.
    + Accelerometers sensors to capture the body's movements.

1.4 Non-functional requirements

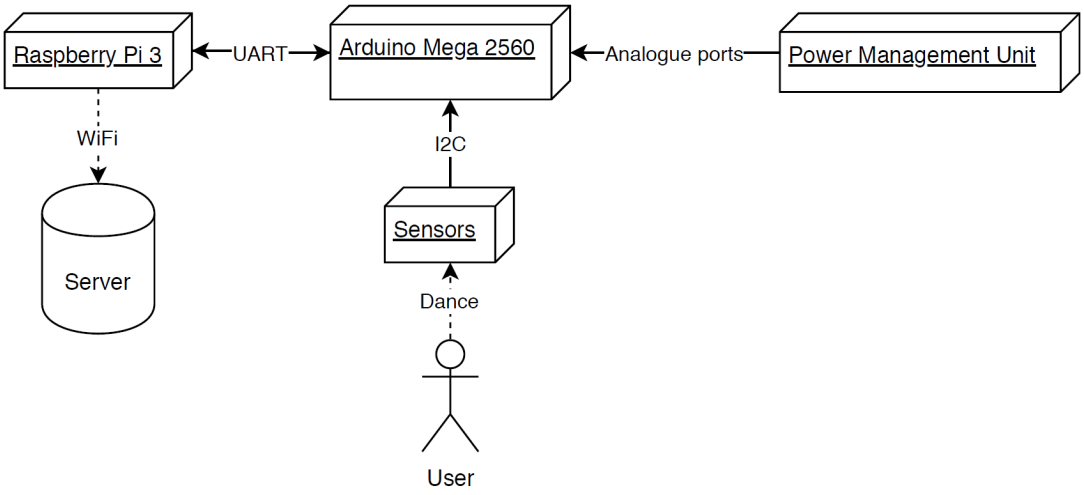
1. In Raspbian on RPi, only the necessary processes are allowed to run, to reduce resource consumption.
2. The system should be able to last at least 1 hour continuously.
3. The system's hardware should not restrict the users' body movement.
4. The system's hardware should be light and sturdy while users' dancing.
5. The system's hardware should be comfortable to wear for prolonged periods, easy to put on and remove.

1.5 User Story

Priority	As a ...	I want to ...	So that I can ...
***	user	identify my dance move accurately	continue to the next dance move and score points
***	user	validate my dance move as fast as possible	continue to the next dance move and have a smooth transition
***	user	stop the detection with a final move	quit the session and take off the devices without any incorrect output
***	user	use the device for a reasonable time period without the battery dying	complete the dance routine without having to charge in between
**	user	use a light and sturdy device	have a comfortable dancing experience
**	user	wear and remove the device easily	change to another user easily and quickly

Section 2 Overall System Architecture

2.1 UML Deployment Diagram



2.1.1 Implementation on Raspberry Pi and the Arduino Mega

The Arduino Mega 2560 will have 2 processes to collect data from sensors and push it to the Raspberry Pi, using FreeRTOS to schedule the tasks.
The Raspberry Pi will run 2 processes in parallel, which are receiving data from the Arduino and analyse it using the Machine Learning (ML) module. Received data is stored in a buffer in Raspberry Pi, to avoid loss of data received when the ML is analysing.

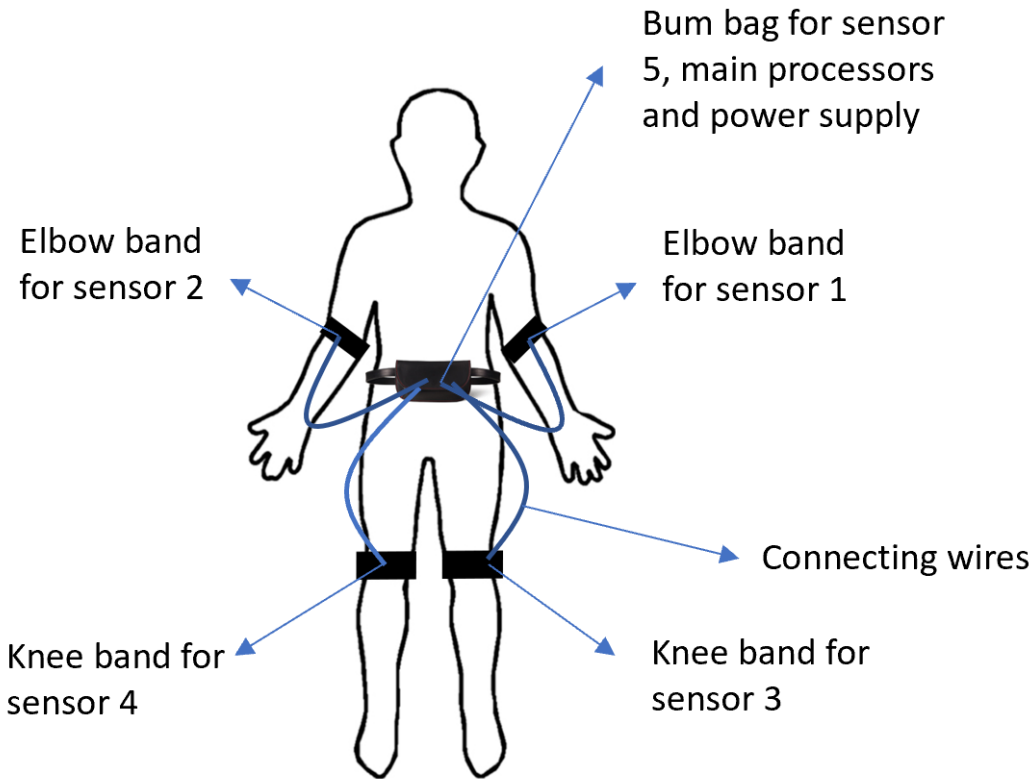
2.1.2 Communication Protocol between RPi and Mega, and between RPi and Server

After the Arduino Mega pre-processes the data from the sensors, it packs the data into a packet, included with a calculated checksum, and sends to the Raspberry Pi through UART at predefined intervals.

2.1.3 Hardware Components in Design Interfaced with RPi and Mega

Gyro-accelerometers communicate with Arduino Mega through I2C. Analogue current sensor communicates with Arduino through the analogue input pins on Arduino. Arduino then sends the relevant data through UART to RPi.

2.2 Hardware Design



2.3 Main Algorithm

Step 1: Sensor Data

The gyro-accelerometers will read the orientation and acceleration of the various body parts in x, y, z axis. Current sensor reads the current value. Arduino Mega 2560 collects gyro-accelerometer data through I2C, analogue input from current sensor, and reads the positive voltage from power supply through another analogue input pin.

Step 2: Data Segmentation

The Raspberry Pi will collect and send data every 0.5 second with overlapping frame technique. For every dance move, the model will record each result after each executions. A threshold will be used to indicate what prediction should be made after several seconds of detection. If some class of dance move is predicted more than the threshold, then that class will be the output.

Step 3: Feature Extraction

The features are splitted into Time Domain Features and Frequency Domain Features (See Section 5.2). We select 14 features in total, which means the dimension for the classifier is 14. There might be an issue on complexity. In the experiment, we will try to reduce the number of features under the premise of acceptable accuracy.

Step 4: Data Collection

For each round of the experiment, we will collect 18 sets of data for each dance move. Every member do each dance move three times in relatively different speed and body movement. Then the data will be divided as training set, testing set, and validation set for different purpose.

Step 5: Model Implementation, Training, Testing and Validation

Four supervised learning algorithms will be applied to implement different models (see Section 5.3 and 5.4). The models will be trained, tested and validated for several rounds. The performance will be compared and the final model should be the best one.

Section 3 Hardware Details

3.1 Identification of Components

Component Name	Quantity	Datasheet
Raspberry Pi 3	1	https://www.raspberrypi.org/documentation/hardware/computemodule/datasheets/rpi_DATA_CM_2p0.pdf
Arduino Mega 2560	1	Board Datasheet: https://www.robotshop.com/media/files/PDF/ArduinoMega2560Datasheet.pdf Chip Datasheet: http://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf
SanDisk Micro SD Card (16GB)	1	N. A.
MPU 6050 GY-521 3 Axis Gyro Accelerometer Sensor Module	5	Board Datasheet: http://synacorp.my/v2/en/index.php?controller=attachment&id_attachment=635 Chip Datasheet: https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf
Current Sensor INA 169	2	http://www.ti.com/lit/ds/symlink/ina139.pdf
Level Shifter TXB0108PWR	1	http://www.ti.com/lit/ds/symlink/txb0108.pdf
Dual-USB Socket	1	N. A.
USB Connector Socket A	1	N. A.
Power Adapter For Pi 3	1	N. A.
Breadboard	1	N. A.
Veroboards	1	N. A.
Power Cable for Arduino	1	N. A.
HDMI to DVI Cables	1	N. A.
Wires	N. A.	N. A.

Header Pins, Plier, Wire Stripper & Cutter	1	N. A.
USB Safety Tester	1	N. A.

3.2 Pin Tables

3.2.1 Arduino Mega 2560 Pin Table

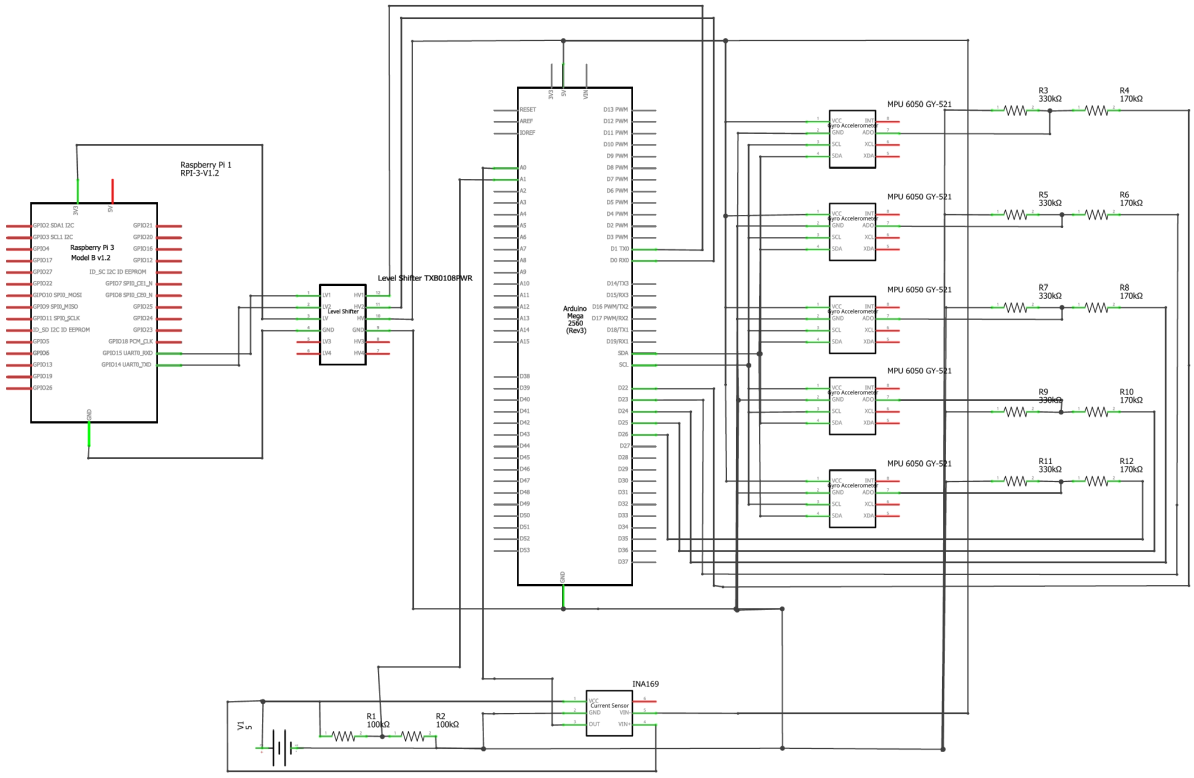
Pin Number	**Pin Name**	**Connected to**	**Explanation**
20	SDA	MPU 6050 GY-521 Gyro Accelerometer (SDA Pin)	Serial data line
21	SCL	MPU 6050 GY-521 Gyro Accelerometer (SCL Pin)	Serial clock line
	5V	MPU 6050 GY-521 Gyro Accelerometer (VCC Pin)	5V reference for gyro accelerometer
	GND	MPU 6050 GY-521 Gyro Accelerometer (GND Pin)	Ground reference for gyro accelerometer
D22		MPU 6050 GY-521 Gyro Accelerometer 1 (AD0 Pin)	Controls the I2C address of gyro-accelerometer
D23		MPU 6050 GY-521 Gyro Accelerometer 2 (AD0 Pin)	Controls the I2C address of gyro-accelerometer
D24		MPU 6050 GY-521 Gyro Accelerometer 3 (AD0 Pin)	Controls the I2C address of gyro-accelerometer
D25		MPU 6050 GY-521 Gyro Accelerometer 4 (AD0 Pin)	Controls the I2C address of gyro-accelerometer
D26		MPU 6050 GY-521 Gyro Accelerometer 5 (AD0 Pin)	Controls the I2C address of gyro-accelerometer
0	RX0	Level Shifter TXB0108PWR (HV2 Pin)	UART reception

1	TX0	Level Shifter TXB0108PWR (HV1 Pin)	UART transmission
	5V	Level Shifter TXB0108PWR (HV Pin)	5V reference for level shifter
	GND	Level Shifter TXB0108PWR (GND Pin)	Ground reference for level shifter
A0		Current Sensor INA 169 (OUT Pin)	Analogue input (for current measurement)
A1		Input voltage of power unit	Analogue input (for voltage measurement)
	5V	Current Sensor INA169 (Vin- Pin)	5V reference for Arduino
	GND	Ground of power unit	Ground reference for Arduino

3.2.2 Raspberry Pi Pin Table

Pin Name	**Connected to**	**Explanation**
RX0	Level Shifter (LV1 Pin)	UART reception
TX0	Level Shifter (LV2 Pin)	UART transmission
3V3	Level Shifter (LV Pin)	3.3 V reference for Raspberry Pi
GND	Level Shifter (GND Pin)	Ground reference for Raspberry Pi

3.3 Schematics



Voltage divider for Arduino pin A4:
Arduino analog input pin can read maximum 5V. Although we intend to have a 5V power supply, in practical situations, the actual voltage from the power supply might fluctuate. In order to protect the analog pin, a voltage divider is used to divide the voltage into half before connecting to pin A4. Arduino will multiply the input from A4 by 2 to obtain the actual voltage.

Voltage dividers for AD0 pins of gyro-accelerometers:
The AD0 pin of gyro-accelerometer controls the I2C address, and takes in a “high” of 3.3V instead of 5V. However, the “high” of Arduino output pin is 5V. Therefore we use a voltage divider to convert the maximum of Arduino pin output to 3.3V before connecting to the AD0 pin.

3.4 Algorithms / Libraries Used

Sensor	**Library Used**
MPU 6050 GY-521 Gyro Accelerometer	Wire.h Serial
Current Sensor INA169	AnalogRead

Algorithm for reading multiple gyro-accelerometers:
The gyro-accelerometers use the I2C interface. Each gyro-accelerometer’s address can be configured to 0x69 or 0x68 by supplying high (3.3V) or low (0V) to pin AD0, allowing the user to use two gyro-

accelerometers at the same time. However, to use 5 sensors, varying the address is not enough. We will connect each AD0 pin of the gyro-accelerometers to an output pin of Arduino. At each clock cycle, only one gyro-accelerometer's AD0 pin will be set to low, corresponding to address 0x68. Arduino will read solely from address 0x68, ignoring 0x69, and therefore ignoring the other 4 gyro-accelerometers. This means that the reading of all 5 gyro-accelerometers will take at least 5 clock cycles, with each cycle only reading one of them.

3.5 Power consumption estimation

To estimate power consumption, we have identified 2 subsystems.

Subsystem 1 : Arduino Mega 2560 + Current Sensor INA 169 + MPU 6050 GY-521 3 Axis Gyro Accelerometer Sensor Modules
Subsystem 2 : Raspberry Pi 3

Components	**Voltage Input**	**Current drawn**	**Power Requirement**
Arduino Mega 2560	5 V	100mA	0.5W
Raspberry Pi 3	5.1V	2.5A	12.75W
MPU 6050 GY-521 3 Axis Gyro Accelerometer Sensor Module	5V	3.9mA	0.0195W
Current Sensor INA 169	5V	10mA	0.05W

3.6 Battery

After identifying the estimated power consumption of the various components, to calculate the load capacity, we need the total power used by the components, the voltage input for the components and the amount of time we need the system to be running.

Total power of the components

Components	**Power Calculation** $P = V \times I$	**Power Requirement**

Arduino Mega 2560	5 V × 100 mA	0.5W
Raspberry Pi 3	5.1V × 2.5A	12.75W
MPU 6050 GY-521 3 Axis Gyro Accelerometer Sensor Module	5V × 3.9mA × 5	0.0975W
Current Sensor INA 169	5V × 10mA	0.05W
	Total power	13.40W

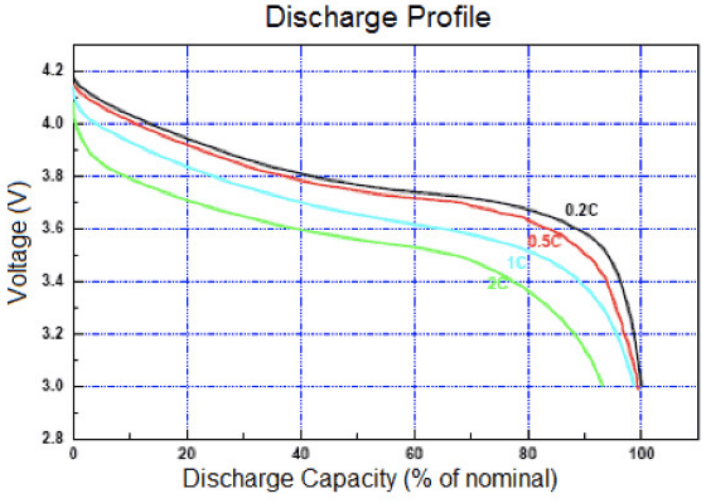
We are looking for a 1.5 hours system operation time along with a battery output voltage is 5V.

V_{BB} = 5V
Time for system to operate = 1.5h

Load Energy = 13.40W × 1.5h = 20.1Wh

Load capacity = 20.1Wh / 5V = 4020 mAh

Based on the above calculation for the battery capacity, our group are looking to opt for a Lithium ion battery bank with a capacity of 5000mAh as our power supply. Due to the fact that the Lithium ion battery pack is rechargeable, and our load capacity requires a battery capacity that is at least 4020 mAh or higher, we saw a 5000mAh Lithium ion battery pack suitable as it is relatively lighter than battery packs of higher capacities and that it comes in a single block, which makes it easy to mount on a person.



Looking at the about graph, although the voltage for Lithium ion batteries in various C rates are below 5v, in battery some banks, they have a built in conversion circuit to allow the battery bank set its output voltage to 5V. Allowing our group to use such a set up for our components.

3.7 Power optimisation

To reduce power consumption of the system by optimizing power, we can look into each subsystem, namely the Arduino Mega 2560 and the Raspberry Pi 3. To optimize power at runtime, the frequency of data collection by Arduino through the sensors can be set to the requirements of the Raspberry Pi's data sampling and data processing requirement. This is to remove unnecessary sensor readings obtained from the sensors and sent to the Raspberry Pi, which causes more power being used in the subsystems.

Also, looking at the load capacity, the Raspberry Pi is the component that draws the most power between the subsystems. Thus for the Raspberry Pi, disabling peripherals like the HDMI and Composite (TV Out), as well as only running useful active processes reduces power consumption. Also, as the Raspberry Pi will be running the machine learning algorithm, ensuring that the algorithms are optimized will reduce data processing time and reduces power consumption.

Section 4 Firmware & Communications Details

4.1 FreeRTOS

At the time this report is written, the newest version of FreeRTOS is *v10.1.1*.
FreeRTOS files will be downloaded from the following github repo:
https://github.com/feilipu/Arduino_FreeRTOS_Library
The relevant library files will then be moved into the Arduino for usage.
FreeRTOS will be used for the synchronisation of all the different tasks which are collecting data from sensors and sending of information from the Arduino Mega to the RPi.

4.2 List of Processes

4.2.1 Processes for Arduino Mega

1. Receive data from Sensors
2. Sending Data to RPi

Each cycle of the processes of the Arduino Mega will be triggered every 500ms, with the process of gathering the readings of the different sensors having priority over that of sending the data to RPi, ensuring that the the latest readings from the different sensors are being sent over. As mentioned previously, FreeRTOS will be used in order to synchronize the tasks, through the use of xEventGroupSync().

4.2.2 Processes for RPi

1. Receive data from Mega
2. Analyze and predict the dance moves from the data received
3. Send the predicted dance move to a remote evaluation server

4.3 Protocol for coordination between Arduino and RPi

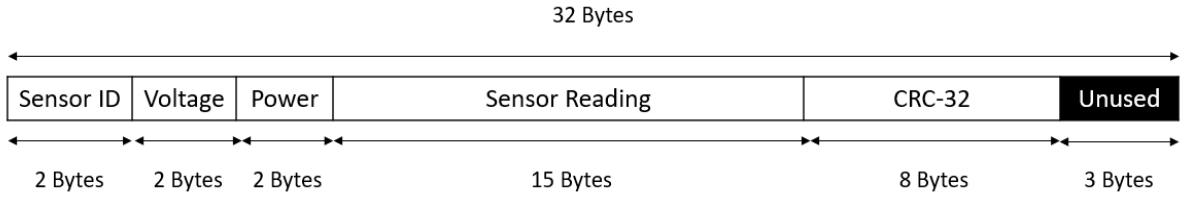
The messages between two devices are sent through UART, with a baud rate of 9600 bps.
As the Arduino Mega 2560 runs at 16MHz, using baud rate generator formula, baud rate of 9600 bps is chosen since it is fast enough and only has the error rate of 0.2%, while the error rate of baud rate of 115200 bps is 3.7%, which is too high.

On bootup, the Arduino will actively listen messages sent from the RPi. After the RPi finishes booting up, it sends a "*start*" message to initiate the handshake with the Arduino. The Arduino then sends back an "*acknowledged*" message to the RPi to confirm that it is running well, and start to push data from sensors to the RPi, every 500ms. The Arduino stops sending data to the RPi when it receives a "*stop*" message. All the messages are included with a checksum, calculated using CRC-32, to ensure no loss or faulty data received on RPi.

Packets sent will be a total of 32 bytes, containing the following information:

- Address/Id (2 bytes)
- CRC-32 (*Cyclical Redundancy Checking*) (8 bytes)
- Sensor Reading (15 bytes)
- Voltage Info (2 bytes)
- Power consumption info (2 bytes)
- Unused (3 bytes)

The packets are sequenced as shown in the following diagram**:**



4.4 Communication between RPi and the evaluation server

Communication between the RPi and the server will be symmetrically encrypted with Advanced Encryption Standard (AES) in Cipher Block Chaining (CBC) mode. In CBC, the message is being XORed with the previous message or a random initialization vector if the message is the first one.

Using Socket, the server listens on one of its ports for messages from RPi. The messages are the predicted dance moves from the Machine Learning module.

Section 5 Software Details

We need to implement a classification programme with the following objectives:

1. Extract relevant movement data from the sensors, while filtering out noise
2. Understand and analyze the movement data
3. Predict which of the 11 (dance moves + final move) groups the dance move belongs to

Since this exercise is rather open ended and it would be near impossible to do this manually due to copious amounts of data generated from the sensors, machine learning would be necessary.

5.1 Data Segmentation

The data of the project are collected from multiple sensors as data streams. There is a new set of data coming in every several microseconds. Therefore, it is essential to segment the data correctly so that we can distinguish the beginning and end of a single dance move. To do this, we will use a new class, neutral movement, in which the user just stands still and does no movements. Therefore, there are 12 classes in total for the project.

Another point we found important is how the server choose the correct predictions. Assume the model runs every 0.5 seconds and the interval of two neutral movements is 10 seconds, which means a dancer dances for 10 seconds, then the model will produce 20 outputs. We set a threshold for each predictions. When the number of times that some class is detected, the model will output the prediction of that class.

5.2 Potential Features

Based on analysis the given dance move videos, we decided to use the following features for this project. Some of the features might be dropped during implementation, as it is possible to minimize the dimensionality to reduce the complexity.

5.2.1 Time Domain Features

- Mean of the linear velocity of the arm
    - x axis
    - y axis
    - z axis
- Mean of the linear velocity of the leg
    - x axis
    - y axis
    - z axis
- Mean of the angular velocity of the body
    - x axis

    - y axis
    - z axis
- Maximum of the linear velocity of the arm
    - z axis
- Maximum of the linear velocity of the leg
    - z axis

5.2.2 Frequency Domain Features

It is likely that the time domain features are enough for the project to detect the simple dance move, so these features are additional considerations when implementing the software.

- Peak frequency of the arm
- Peak frequency of the leg
- Peak frequency of the body

5.3 Learning Paradigm

There are, at a high level, three main machine learning paradigms

- Supervised Learning
  Supervised learning is a machine learning approach to infer a function with a set of labelled training data (McNulty, 2015).
- Unsupervised Learning
  Unsupervised learning is a machine learning approach to infer a function with training data which are not labelled or categorized (McNulty, 2015).
- Reinforcement Learning
  Reinforcement learning is about that an agent takes actions which maximizes the cumulative rewards (Bajaj, n.d.).

We decided to use supervised learning. Supervised learning is easier to implement and able to produce accurate predictions for this project. The reason is that, according to the definition above, the data need to be labelled. The data in this project are extracted from sensors and can be labelled as different classes. To be specific, there are 11 dance moves. We labelled each of them as one class and when we do the particular dance move, we collect data and then we label the data with its class.

5.4 Learner Model

With supervised learning, the goal for this project is to find a relationship between input data (body movement) and output data (categories of dance moves) so we can make accurate predictions. There are different models in supervised learning. Given the same data, the accuracy can be different because of the model used. Therefore, choosing a suitable model is essential.

The following section introduces some commonly used models in supervised learning. We make comparisons and illustrate the reason why we choose them.

5.4.1 k-Nearest Neighbour

K-Nearest Neighbour (k-NN) is a non-parametric algorithm, which is commonly used for classification and regression in the industry. The parameter k is set by the programmer. K-NN simply separates training data into different class during training, and find k nearest neighbours of the input data point when testing and predicting. The distance can be computed in many methods, in our project, we decided to use Euclidean distance (Karpathy, n.d.).

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

K-NN is actually easy to implement and can provide relatively high accuracy with enough data. However, it is expensive for computation. The time complexity is $O(nd^2)$, where n is the number of samples and d is the number of features (Patel, 2017). Moreover, it requires huge memory because it stores all data in groups. As we need to do a real-time prediction and the memory for the Raspberry Pi is limited, k-NN might not be the best choice.

5.4.2 Support Vector Machine

A support vector machine (SVM) is a linear discriminative classifier model, which finds a optimal hyperplane which classify training data and categorizes new data points. Specifically speaking, each data point is plotted in an n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate (Ray, 2017).

By the defination, SVM only classifies two classes linearly. Thus, to use SVM in this project, where there are 11 classes (11 dance moves), we will take advantage of one-against-the-rest approach (Watkins & Weston, 1998), which means distinguishing the first class against the other 10 classes, then take the result and applied to the second classes again. To improve the time complexity, binary search can be used as well.

5.4.3 Perceptron Learning Algorithm

A perceptron model takes an input and computes the weighted sum. It returns 1 if the sum is larger than the threshold, and returns 0 otherwise. Such a perceptron model is only applicable on linear classification. Thus, to apply Perceptron Learning Algorithm (PLA) in this project, we need to apply the same one-against-the-rest approach described in the section of SVM.

5.4.4 Artificial Neural Network

Artificial neural networks (ANN) are computing systems vaguely inspired by the biological neural networks that constitute animal brains. The neural network itself is not an algorithm, but rather a framework for many different machine learning algorithms to work together and process complex data inputs (Artificial Neural Networks as Models of Neural Information Processing, n.d.).

ANN with combined algorithms and multiple layers can be very powerful and provide high accuracy of prediction. However, an ANN model needs a large data set for training to achieve this. Therefore, in this project, ANN may not be the best choice in our opinion because of the lack of data.

We will explore all mentioned models in this project and make comparisons of the performance. According to the performance, we can choose the best algorithm for this project.

5.5 Training of Models

To train the models, we decided to collecting 18 sets of data (there are 6 members and each of us dances for 3 times) for each dance move for the each round of training. Each member will dance in different speed and slightly different body movement so that our data sets are more comprehensive. When the accuracy of prediction is high enough, for example, 90% accurate, we will stop training all models. Based on the performance of each model, we will choose the best one as the final model and continue training it.

We will apply some open source libraries for implementing and training the model.

- Python 3.7
  With powerful pre-built libraries and high flexibility, Python is the most popular programming language for data science. We will use Python and some libraries for the model implementation, including but not limited to PyTorch and Scikit-Learn.
- Scikit-Learn
  Scikit-Learn is a machine learning library for Python, which provides both supervised and unsupervised learning algorithm by a consistent interface.
- PyTorch
  PyTorch is a relatively new machine learning library developed by FAIR(Facebook AI Research Team). The features like simplified preprocessors, custom data loaders, and multiGPU support are beneficial for our project.

5.6 Testing of Models

About 20% of the data collected will be separated as testing data for testing purpose. All the models will be tested using the same testing data set.

Two values will be considered in testing phase.

- Training Error
  Training error is the error produced when the model was trained on. The average error is computed using this formula:

$$\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$$

L is the loss function

- Test Error
  Test error is the error gotten when we run the model on the test data set. It is computed by the classification error.

5.7 Validation of Test Data

To ensure that the module is not biased and can detect and distinguish the data patterns correctly, we need to validate the model. We will apply the following two methods for validation.

- Leave-One-Out Cross-Validation (LOOCV)
  This method is processed in the following steps:
  1) Divide the data set into two groups, one is training set, the other is validation set
  2) Use the training set to train the model
  3) Use the validation set to do evaluation and compute corresponding mean square error (MSE)
  4) Repeat the process N times (where N is the size of the thetire data set) and compute the average MSE to estimate the test MSE
- K-Fold Cross-Validation
  This method is processed in the following steps:
  1) Divide the data set into K subsets
  2) Choose one subset to be the validation set and the other subsets are training set
  3) Compute the square error for this case
  4) Repeat step 2-3 K times, make sure each subset is chosen to be the validation set once
  5) Compute the MSE

  In our project, we will try K=5 and K=10 and make comparison.

It is found that LOOCV performs better when the data set is small, thus, we may use LOOCV in the first few rounds of training and apply K-Fold when the data set becomes large enough.

Section 6 Project Management Plan

Week	Milestones
4	Completion and Submission of Initial Design Report (All)
5	Confirmation of initial design of product (All) Start working on MEGA/RPi basic framework (Comms) Research and Implementation of relevant algorithms for Machine Learning (Software)
6	Purchasing of relevant parts for system) Machine Learning Model Implementation (Software)
Recess	Preparation for individual subcomponent evaluation
7	Integration of relevant systems and subcomponent demonstration
8	Completion of first prototype
9	Testing + Refining for first 5 moves
10	Buffer week
11	Inclusion of all 10 moves
12	Testing with all moves
13	Finalisation of wearable Preparation of report + Final evaluation

References

Artificial Neural Networks as Models of Neural Information Processing. (2019). Retrieved from https://www.frontiersin.org/research-topics/4817/artificial-neural-networks-as-models-of-neural-information-processing

Bajaj, P. Reinforcement learning - GeeksforGeeks. Retrieved from https://www.geeksforgeeks.org/what-is-reinforcement-learning/

Karpathy, A. CS231n Convolutional Neural Networks for Visual Recognition. Retrieved from http://cs231n.github.io/classification/#nn/

McNulty, E. (2015). What's The Difference Between Supervised and Unsupervised Learning? - Dataconomy. Retrieved from https://dataconomy.com/2015/01/whats-the-difference-between-supervised-and-unsupervised-learning/

Patel, S. (2017). Chapter 4: K Nearest Neighbors Classifier – Machine Learning 101 – Medium. Retrieved from https://medium.com/machine-learning-101/k-nearest-neighbors-classifier-1c1ff404d265

RAY, S. (2017). Understanding Support Vector Machine algorithm from examples (along with code). Retrieved from https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

Weston, J., & Watkins, C. (1998). Multi-class support vector machines. Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, May.