



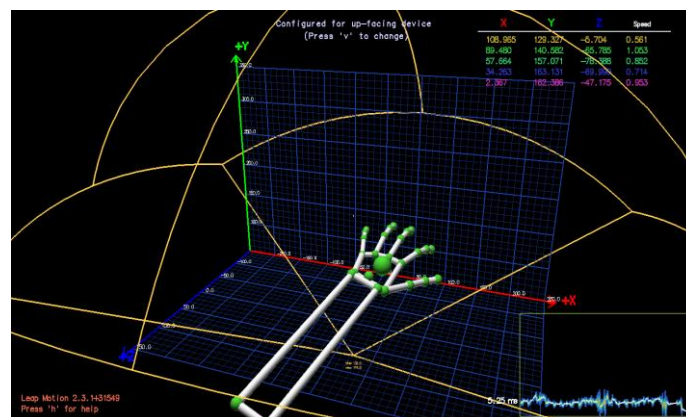
School of Engineering and Information Technology

Electrical Power and Industrial Computer System Engineering

Interactive Gesture Controller for a Motorised Wheelchair

This report is submitted to the school of Engineering and Information Technology at Murdoch University, in partial fulfilment for an honours degree in Engineering.

2015



Author: Jade Sciberras

Supervisor: Dr David Parlevliet

Abstract

This paper explores in great detail the design and testing of a gesture controller for a motorised wheelchair. For some, motorised wheelchairs are part of their everyday life. Those individuals who depend on their motorised wheelchair do so for a vast range of reasons; therefore, it is reasonable to assume that modifying and improving upon the standard joystick controller for a motorised wheelchair can benefit a person's way of life significantly.

The design of the gesture controller is heavily based around the user's needs so as to benefit them and compliment their strengths to give them more control. For individuals with limited movement and dexterity, the user interface, system responsiveness, ergonomics and safety were considered when engineering a system that is intended for people to use.

A device capable of recognising a hand gesture was carefully chosen. The technology that is readily available for this application is relatively new and not extensively documented. The LEAP motion sensor was chosen as the hand gesture recognition device to be the controller for a wheelchair. This device has hand recognition software but the device's software lacks the predictability and accuracy required for a motorised wheelchair controller. Through testing, the controller accuracy improved. Although this controller is adequate for a laboratory environment, further testing and development will be required for this alternative wheelchair controller to evolve into a commercial product.

The gesture triggered controller was designed around the capabilities of the developer's hand; but the method outlined in this paper is transferable to any individual hand size and more importantly the limitations of their hand gestures. The outcome of this thesis is a customised, non-invasive hand gesture controller for a motorised wheelchair that is able to be fully tailored to a person's capability without losing its responsiveness or accuracy.

Abbreviations

- EPW - electric-powered wheelchairs
- WEKA - Waikato Environment for Knowledge Analysis
- PWM - Pulse Width Modulation
- FPS - Frames per second
- ROC - Receiver Operating Characteristic
- SDK – Software Development Kit
- GUI – Graphical User Interface
- API – Application Programming Interface
- UDP – User Datagram Protocol
- PCB - Printed Circuit Board

Table of Contents

Interactive Gesture Controller for a Motorised Wheelchair	i
Abstract.....	i
Abbreviations.....	ii
Table of Contents.....	iii
Student Declaration	1
Acknowledgments.....	1
1. Introduction	2
2. Wheelchairs	3
2.2 Controlling the Movement of a Wheelchair	5
2.3 Research Analysis for alternative control methods for a Motorised Wheelchair	7
3. Selecting a Device for Detecting Hand Gestures	8
3.1 HOVER	11
3.2 GestIC® Technology	12
3.3 LEAP Motion Controller Justification	14
4. Leap Motion.....	15
4.1 Hardware	16
4.2 Software.....	16
4.3 Application Programming Interface [API] Structure Overview.....	17
4.3.1 Vector Class.....	18
4.3.2 Finger Class	18
4.3.3 Hand Class.....	21
4.3.4 Frame Class	22
4.3.5 Listener Class and Controller Class	22
4.4 Online Reviews.....	23
5. Machine Learning.....	24
5.1 Data mining.....	24
5.2 Waikato Environment for Knowledge Analysis [WEKA]	25
5.3 Decision Trees	25
6. Ergonomics and User Experience.....	26
6.1 Easy Gestures	26
6.2 Exaggerated Responses and Dynamic Feedback	27
6.3 Intuition.....	27

6.4 Resources	27
7. Project Design	28
7.1 Hand gesture Controller Design.....	29
7.1.1 Leap Data Sampling Application: LEAP Listener.....	32
7.1.2 WEKA Results	33
7.1.3 Conclusion of Hand Gesture Design Process	38
7.2 Server Application Description	39
7.3 Controller Function Program Design.....	39
7.4 Robotic Testing Platform	45
8. Test Result.....	49
9. Problems Encountered.....	51
9.1 LEAP motion Connectivity.....	51
9.2 Robot Test platform Troubleshooting	52
9.3 Latency Issues	52
10. Future Improvements	53
11. Conclusion.....	55
12. References	56
Appendices.....	63
Appendix A.....	63
LEAP Data Sampling Application	63
Appendix B	65
WEKA Data File.....	65
Learned Predictive Model.....	69
Appendix C	73
Server Application Program	73
Appendix D.....	78
Appendix E	81
Full Bridge Motor Driver L298N	81
Raspberry Pi 2 Specs	81
Bill of Materials [BOM].....	82

List of Figures

Figure 1: Typical Motorised Wheelchair Controller- Creative Commons (Wikipedia 2015)	5
Figure 2: Microsoft Kinect (Dev.windows.com 2016)	9
Figure 3: Intel Perceptual Computing: Intel RealSense Camera (Software.intel.com 2016)	9
Figure 4: DUO3D-DUO mini MLX (Duo3d.com 2015)	9
Figure 5: LEAP Motion (Motion 2016)	9
Figure 6: HOVER board (Hover Labs Co 2015)	11
Figure 7: Leap Motion Controller.....	15
Figure 8: Diagnostic Visualiser Images of Different Sphere Radius'	21
Figure 9: Overall System Flow Diagram.	39
Figure 10: Gesture Controller Procedure Diagram	40
Figure 11: Drive Engine Procedure Logic Diagram.....	41
Figure 12: Index and Thumb and Index and Middle Flowchart procedure Representation.....	42
Figure 13: Thumb and Pinky Gesture Flowchart Procedure	43
Figure 14: Engine Equalisation Procedure for Motor Speed Stabilisation.....	44
Figure 15: Diagram Representation of Test Robot Motor Rotation Direction	46
Figure 16: Robot Test Platform	46
Figure 17: Wiring Diagram of Robot Test platform	46
Figure 18: LEAP motion Troubleshooting Tab in Settings- Recalibration	51
Figure 19: L298N Stepper Motor Driver Controller Board for Arduino	81
Figure 20: Image of a Raspberry Pi Model 2.0	81

List of Tables

Table 1: Alternative Wheelchair Controllers.....	6
Table 2: Comparison of Depth-Sensing Cameras and Tracking devices.	10
Table 3: Finger Name Code Representation	19

Table 4: Bone Name Code Representation.....	19
Table 5: Joint Name Code representation	20
Table 6: Hand Gesture Representation.....	30
Table 7: Confusion Matrix for all Hand Gestures generated from WEKA.....	34
Table 8: Detailed Accuracy by Gesture Type	35
Table 9: Summary of Total data accuracy results formulated from WEKA J48 Algorithm	35
Table 10: New Detailed Accuracy of Gesture: Generated from more collected Data.....	36
Table 11: Confusion Matrix for Selected Hand Gestures generated from WEKA.....	37
Table 12: Hand Gesture Wheelchair Function Task Allocation.....	38
Table 13: Results If the tasks were able to be completed	50
Table 14: Robot Platform GPIO to L298N Arduino Motor Driver Board Pin Connections.....	82
Table 15: BOM for Robot Platform Base.....	82
Table 16: Expendable materials list for Robot Platform	84

Student Declaration

Unless referenced, I declare that the following report is my own work.

Acknowledgments

I acknowledge my supervisor, David Parlevliet, for seeing me through this project. His expertise and knowledge has helped keep this project on track.

I would like to thank my peers and long term friends, Ben Pattimore, Adam Gioffri, Michael Colson, Jordan Goodchild, Frederick Gao, Richard Lee and Alex Pechkov. I would also like to acknowledge my partner, Jake Alamdar, his love, support and understanding has made this project a pleasant experience. A warm thankyou goes towards my parents, Anthony Sciberras and Maria Concetta Sciberras.

Lastly, I would like to acknowledge the companies and online communities that keep their products and software resources open source. Without the aid of free software developing environments, there would be little motivation for future innovations.

1. Introduction

Sensing and tracking technology is a fairly new area of the consumer market that has taken off in recent years. Depth and tracking sensors have been widely used for gaming and other computer 'app based' application. An example is the popular Xbox Kinect (Support.xbox.com). This device is able to map out the human body and track its movements. The moves are displayed on a screen or television as an interactive way to play games. Similar devices provide recognition and tracking for facial features and some devices are accurate enough to track hand gestures. These cameras and sensors are starting to be incorporated into laptops, desktops and keyboards. They are rumoured to take over and replace computer mouses, track pads and touch screens as these devices Pick up gesture recognition that will zoom, minimise and swipe without physically touching a device.

Although many reviews about these products point out that they are not suitable for commercial use (Engadget 2016), projects have emerged where they are being used for precision (Visnjic 2016). Robotic control and drone control are examples of laboratory based experiments that have the potential of becoming commercialised.

The aim of this thesis project is to design an interactive hand gesture controller to manoeuvre a wheelchair. An elegant solution is required to cater for the requirements of the users. An emphasis on user safety has been incorporated into the final product. This project ultimately explores the capabilities of a hand gesture controller device and explores its accuracy. The project aims to provide awareness of different technologies currently employed by wheelchairs and provides a comparison of the different tracking and body recognition devices.

2. Wheelchairs

The purpose of wheelchairs are to help the physically disabled, elderly or people who suffer from extreme fatigue and short to long term injury where walking is difficult or impossible. According to past studies, approximately 5 million people in the European Union, 2.8 million people in the United States, and 2.2 million people in Australia are wheelchair users (Newdisability.com 2015). Approximately there are ten times more wheelchair users in Australia per capita compared to the United States and European Countries, therefore providing reason for this projects objectives. With the number of wheelchair users predicted to increase, it makes sense that the design of a wheelchair will improve with technology.

Two different types of wheelchair classes are the manually propelled and the electrically power. More extreme versions have been made for competitive sport and all terrain purposes. There have been many advances to the standard wheelchair. Improvements have been made to the design and quality of materials; gyroscopic technology and various sensors have been incorporated to increase the functionality and sustainability of the common wheelchair (Hiremath, Ding and Cooper 2013).

Motorised Wheelchair or electric-powered wheelchairs (EPW), are propelled by electric motors rather than manually moving the wheels in order to commute and manoeuvre. EPW'S are used be people who are physically unable to manually propel a manual wheelchair over any distance due to a physical impairment or disability. Individuals who suffer from fatigue and cardiovascular stress benefit from EPW's to provide them with a better quality of life.

Smart Wheelchairs (Simpson 2008) are improved variations on the standard EPW that assist individuals that are physically disabled. Smart wheelchairs differ from general motorized wheelchairs by their in-built control system. The intention is to supplement the user's task of driving and manoeuvring the motorized wheelchair and to optimise safety. Unlike the regular

motorized wheelchair that are controlled by a joystick (switch, potentiometer or touch sensitive device); a smart wheelchair will have a computer and a range of sensors, like proximity sensors, that work together to analyse feedback information to drive the motorized wheelchair.

Adding collision-avoidance to a smart wheelchair offers a marketable feature. One advantage of Smart Wheelchairs is that they are more customisable, and therefore can be tailored and or designed to suit a variety of user types. Some Smart Wheelchair platforms are designed for users with cognitive impairment. Collision avoidance technologies (Lopresti EF 2016) ensure the safety of the user in the event the wrong command is given that may result in a collision. Cerebral Palsy and Quadriplegia require different platforms that will cater for the users which have severe motor disabilities. The Smart wheelchair may have smart technologies to distinguish between accidental command executions (Kouroupetroglou) in the event of random muscle activation. Path-planning is an artificial intelligent technique incorporated into smart wheelchairs to aid users with severe motor skills (Lopresti EF 2016).

There are a wide range of technological improvements to wheelchairs; these include an EPW's motor controller.

2.2 Controlling the Movement of a Wheelchair



Figure 1: Typical Motorised Wheelchair Controller-
Creative Commons (Wikipedia 2015)

The most common mounted controller for motorised wheelchairs is in the form of a joystick as shown in Figure 1. The joystick functionality is similar to the Atari joystick controller that was popular in the 1980's. Its use is simplistic and effective as pushing the joystick forward will make the wheelchair move in a forward direction.

Wheelchair controllers can be programmed to be proportional or non-proportional (Doherty 2012). Proportional controllers will make adjustments to speed depending on how much the user deflects the joystick controller in a particular direction. Non-proportional controllers does not allow the user to have much control over speed, making these controllers more suitable for users who have less control over their body movements. Some controllers are supplemented with additional buttons that allow the user to switch between multiple control modes.

The normal EPW needs to be operated through a joystick by hand. However, the hand functionality of an individual may be limited or even not available in some patients with severe disabilities. The idea of the alternative wheelchair control is to use other parts of the body rather than hand to operate some sort of a proportional control joystick (Atwiki.assistivetech.net 2015). The alternative wheelchair control includes sip-n-puff control, chin control, head control, speech control and tongue-operated solution. These controllers are available on the market as a viable

replacement option for the joystick controller. A brief description of alternative wheelchair controllers is provided in

Table 1.

Table 1: Alternative Wheelchair Controllers

Alternative Motorised Wheelchair control	Description	Advantages/Disadvantages/Facts
Sip-n-Puff	Provides a wheelchair user full control of the motors by inhaling and then exhaling into a pneumatic tube. The pressure applied by the sips and huffs control the speed direction of the motors. Lower level sips and puffs are used for steering. (Atwiki.assistivetech.net 2015)	Uncomfortable and invasive Expensive Fairly intuitive to use but takes practice
Head Control	Various switches are mounted into the headrests. The switches are activated through head movements. A common design is that left and right wheelchair movement is activated through the left and right side headrests respectively. (Atwiki.assistivetech.net 2015)	Most common amongst wheelchair users that have good head mobility Non-invasive. Advances: research institutions have begun to incorporate ultrasonic transducers and accelerometers instead of switches.
Chin Control	A joystick is mounted in close proximity to the chin for activated control of a wheelchair. Neck movements, such as flexing, extension and rotation, activated the cup shaped chin joystick. (Atwiki.assistivetech.net 2015)	Designed for individuals with good neck and head movement
Speech control	These types of controllers are high level speech recognition systems. Common vocabulary words will prompt the motors to carry out an action that intern controls the direction of the motors. (Atwiki.assistivetech.net 2015)	Most popular for spinal injury patients. User Friendliness
Tongue Operated Controller	This is a dental mouth Piece that is situated on the roof of an individual's mouth. The device consists of nine switches. The switches are activated by the tongue. Generally the rear switches are the go slower and the front switches are to go fasted hence speed control.	Only one commercially available product: Tongue Touch Pad Invasive Uncomfortable

2.3 Research Analysis for alternative control methods for a Motorised Wheelchair

There are many different types of alternative wheelchair controllers that have been designed and tested under laboratory conditions, yet most have not been refined into marketable products. When conducting research, it was noticeable that there were more academic papers on 'brain controlled' wheelchairs than any other type of controller. The use of accelerometers to measure the Pitch, roll and yaw of head movement is a fast developing area for motor control. *Pajkanovic et al.* explores the use of a microcontroller and accelerometer system for motion control of a wheelchair for quadriplegics (Pajkanovic and Dokic 2013). The experiments showed good overall results with a 94.16% success rate for correct commands and a 13.66% error rate where the system recognises a command that was not intended by the user (Pajkanovic and Dokic 2013).

Relating to the specific topic of hand gesture recognition controllers for motorised wheelchairs, there are several papers that have explored the possibilities and limitations in this area. In the article by Nguyen Kim-Tien et al. the authors presents an approach for controlling wheelchair movement using hand gesture recognition (Kim-Tien, Truong-Thinh and Cuong 2013). Their method is able to recognize 5 different hand gestures for five status movements of wheelchair: forward, reverse, left, right and stop. In the article by Pande et al. acceleration technology is used to detect hand gestures (Pande et al. 2014). The software interprets the motion intended by user and moves accordingly (Pande et al. 2014). Depending on the direction of the Acceleration, the microcontroller controls the wheelchair directions: forward, reverse, left, right. These research articles do not give abundant information regarding the accuracy and reliability of their system.

3. Selecting a Device for Detecting Hand Gestures

There are many different approaches when it comes to designing the best solution for this project. From the background literature review, two completely different approaches were discussed: one solution experimented with an accelerometer system and the other relied upon an algorithm that detects hand gesture through changes in the curvature of the hand.

A new approach is to select a device that already has hand gesture and motion tracking capability. There are a number of products which have these features that are readily available on the market. For economic reasons and time restrictions, research into these products was conducted and the most appropriate selected.

The strengths and weaknesses of four devices are compared to establish the best device for this project. The devices are the Microsoft Kinect (Dev.windows.com 2015), Intel RealSense Camera (Intel 2015), DUO3d (Duo3d.com 2015) and LEAP Motion (Motion 2015). These devices have similar features but all vary on different levels. The advantages and disadvantages of each controller are outlined and compared in Table 2. Throughout the project, two other alternative devices were researched: they are the MG3030/3130 Microchip GestIC technology ("Introducing the World's First E-Field Based 3D Gesture Controller" 2016) and the HOVER board (Hover Labs Co 2015). Although they also have gesture recognition and development capabilities, their usefulness in achieving the aims of this thesis project varies greatly.



Figure 2: Microsoft Kinect (Dev.windows.com 2016)



Figure 3: Intel Perceptual Computing: Intel RealSense Camera (Software.intel.com 2016)



Figure 4: DUO3D-DUO mini MLX (Duo3d.com 2015)

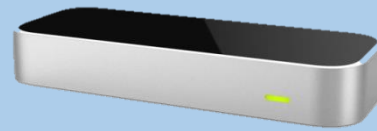


Figure 5: LEAP Motion (Motion 2016)

Microsoft Kinect (Dev.windows.com 2016) is used for gaming and applications that require skeleton tracking. It has problems with distinguishing between hands and fingers, and it requires a large power adapter. It has good depth sensing with a clean depth image, and it can track up to 6 skeletons and 25 joints per skeleton. DUO3D-DUO (Duo3d.com 2015) is a Kickstarter project that is under active development. It will be best utilised in projects that require high range and mid-range depth. Intel Perceptual Computing device (Software.intel.com 2016) is not recommended for applications that require high precision and accuracy. It is built for close range tracking, has built in gesture recognition, and it has face tracking and face recognition capabilities. The LEAP Motion sensor (Motion 2016) has limited sensing range and cannot track faces. It has gesture recognition, and it is very fast and claims to have accurate finger tracking capabilities.

Table 2: Comparison of Depth-Sensing Cameras and Tracking devices.

Type	Microsoft Kinect	DUO3D-DUO mini MLX	Intel Perpetual Computing/ Intel RealSense Camera	LEAP Motion
Information				
<i>What is it used for?</i>	<ul style="list-style-type: none"> • Xbox gaming • Applications requiring skeleton tracking at mid-range. 	<ul style="list-style-type: none"> • High-speed, mid-range depth sensing • Portable and outdoor projects 	<ul style="list-style-type: none"> • Desktop/laptop applications • Close range applications 	<ul style="list-style-type: none"> • Desktop applications • Close range applications • High speed applications
<i>Advantages</i>	<ul style="list-style-type: none"> • Good depth sensing with a clean depth image • High resolution: Depth stream is 512x424 16bits/Pixel • Skeleton tracking: up to 6 skeletons and 25 joints/skeleton • Kinect SDK raw depth data from sensors and IR images 	<ul style="list-style-type: none"> • Embedded stereo imaging for high performance 3D sensing (Duo3d.com 2015) • Ideal for small spaces and mobile projects. • High image rate at 360FPS • Includes gyroscope, temperature and accelerometer 	<ul style="list-style-type: none"> • Powered by USB and smaller than Kinect • Built for close range tracking and has built in gesture recognition like “grab and release” • Face tracking and face recognition capabilities • Developers have access to raw data. 	<ul style="list-style-type: none"> • Gesture recognition • Very fast and accurate finger tracking • Calibrate and map fingertip and finger joint positions with high accuracy. • Developer friendly with room to make apps • Compatible with MAC OS and Windows
<i>Disadvantages</i>	<ul style="list-style-type: none"> • Large power adapter needed and dedicated power cord. • Cannot easily distinguish between hands and fingers • Can only recognise large gestures. 	<ul style="list-style-type: none"> • Kick-starter project that is continuously under development • Young product with not much support • Product API doesn’t provide depth data. 	<ul style="list-style-type: none"> • Not recommended for applications that need high precision and accuracy. 	<ul style="list-style-type: none"> • Limited sensing range • Can only track fingers (no skeleton or face tracking/ recognition) • No access to depth data.
<i>Developer Capabilities</i>	Yes- free	Yes- \$95/year (Duo3d.com 2015)	Yes- free	Yes- free
<i>Elegance</i>	Not discrete. Large Rectangle structure	Not in a sturdy casing but small and discrete.	Small	Robust/ Sturdy/ Small
<i>Cost?</i>	\$149.99	\$695.00 (Duo3d.com 2015)	\$99.00	\$79.99

3.1 HOVER

The HOVER board shown in Figure 6, is a hand gesture sensor development kit that detects hand gesture movements and has touch sensitive features. HOVER technology is similar to the standard

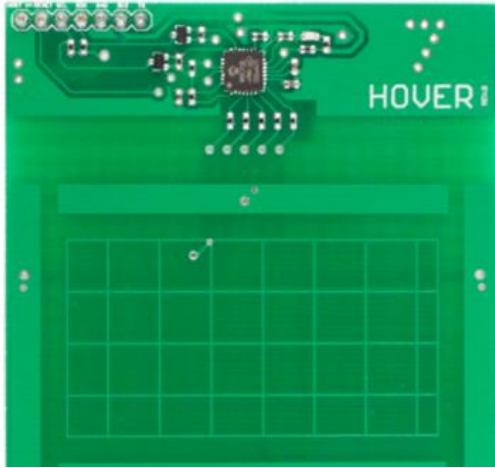


Figure 6: HOVER board (Hover Labs Co 2015)

capacitive screens used for tablets and smartphones. The screen generated an electric field; when a conductive element touches the screen, a distortion in the electric field occurs and triggers an event (Brown 2015). The actual 'hover' feature of the board is produced by a grid of capacitive electrodes that create an electric field that expands in an upward direction from the boards face;

similarly, when you move an electrically conductive body through the field, the electrodes will pick up the distortion.

This device is compatible with low voltage applications and works with any Arduino or Raspberry Pi. Some noteworthy features are listed:

1. Up to 13cm hand gesture detection range.
2. Currently recognisable hand gestures: Swipe Up, Down, Left and Right.
3. 5 electrodes for touch sensitive inputs (touchpad capabilities); double taps and simultaneous taps on multiple areas can also be detected.
4. I2C slave (multi master bus, therefore more than one chip can be connected to the same bus)
5. Powered by 3.3V

This device retails for approximately \$50. Although the HOVER is a fun and interesting hardware device, it does not accommodate for a wide range of hand gestures and therefore limits the design of the wheelchair controller significantly for this project.

3.2 GestIC® Technology

GestIC Technology is a world-first technology integrated circuit from Motorola that may be utilised as a 3D Gesture Controller (Microchip.com 2015). The MGC3030 and MGC3130 Motorola integrated circuits sense and track hand gestures through an electrical field (E-Field measurement principles) in real time (Cardinal 2012). The microchips patented GestIC technology allows developers to enable user command inputs with natural hand and finger movements. These microchips are for embedded usage applications and have a sophisticated on-chip gesture library. Hand gestures such as 'wave', 'hold', 'flick', 'air wheel', are readily available gesture features of the MGC3030 and MGC3130 microchip. Other features such as detecting hand presence, sensor touch ability and hand position tracking, can be enabled through GestIC technology. Some Typical applications include home automation, PC peripheral devices, audio products, industrial and medical switches and game controllers.

GestIC technology MGC chips do not need a host processor, therefore making interaction between intuitive hand gestures and devices simple (Microchip.com 2015). Other features include:

1. Low power design
2. GPIO interface
1. No ambient influences
2. Noise handling
3. Proximity Detection
4. 0 - 15 cm detection range
5. Accurate and reliable gesture detection every use
6. No housing requirements

7. MGS setup uses thin, low-cost sensing electrodes like a standard PCB.
8. Communication be configurable I/O ports of I2C bus

The MGC microchips work on a wide range of interfaces. The free development software (SDK) also provides a graphical user interface (GUI). This real-time gesture processing microchip technology is a robust and advanced solution gesture controlled systems.

A development board with MGC3130 GestIC 3D Gesture Recognition and tracking, proximity and touch sensing capabilities cost an average of \$179.00 Australian dollars from various vendors (Microchipdirect.com 2015). The SDK, GUI and GestIC library allow for great versatility for projects.

This technology would have been a useful tool for this project but the LEAP motion controller was purchased weeks in advance. If the time frame for this project was of no issue, the change from the LEAP Motion controller to the MGC3130 development Board would be recommended. The main notable advantage that the GestIC technology chip has over the LEAP motion controller is that it does not require host processing therefore eliminating the need for the chip to be constantly bridged through a computer. This means that in future, integration into a motorised wheelchair is seemingly more plausible.

3.3 LEAP Motion Controller Justification

The leap controller is the best motion sensor (Learn.sparkfun.com 2016) available because it satisfies the objective for a non-invasive and cheap alternative controller for a motorised wheelchair. Research into the technical specifications of this product proves that it is a cheap and reliable off- the- shelf option. It is an open source device that is compatible with many programming languages including Java, C+, C#, Unity and Python. Each language has its own LEAP SDK library (Developer.leapmotion.com 2016). The development software is a free download with easy installation. The relevant data collected from the LEAP sensor is certainly accessible and can be fed into different engineering programs such as LabVIEW (Ni.com 2016). The LEAP sensor is also compatible with Raspberry Pi (Raspberry Pi 2016) and Arduino which is convenient for project expansion purposes.

The LEAP sensor has USB connectivity and can operate on Windows 7/8 or MAC OSX 10.7 with an AMD Phenom II or an Intel i3, i5, i7 processor. As the LEAP sensor does need a processor and a graphics card, it does limit the direct application use and cannot be directly integrated with a standard motorized wheelchair. However this sensor is designed to map out the human hand in detail that exceeds the capabilities of other previous discussed devices in section 3, 3.1 and 3.2. The LEAP sensor is capable of recognising any hand gesture. This means the user is not limited by their hand size, shape or finger movement capabilities. The sensor has background image cancelation features that filter out any face or body images in the LEAP sensors range of view. This inbuilt feature makes this sensor unique from the other devices discussed and minimises any errors that may occur from accidental body movement seen by the LEAP motion sensor. For these reasons, this sensor is the best suited option to achieve the objectives for this project.

4. Leap Motion

The LEAP motion control sensor, seen in Figure 7, is able to recognise and track the position of a hand. It is a tool that is able to display the track movements of fingers and finger shaped objects. The sensors and software motion tracking capabilities include hand recognition, hand direction (palm facing up, down, left,



Figure 7: Leap Motion Controller

right), forearm, fingers and hand gestures. There have been many projects that use the functionality of the LEAP motion sensor since its release in 2013. Robotic arm control (Instructables.com 2016), 3D printer control (Abarca and Abarca 2014), gaming controllers (Motion 2016) and music synthesisers (Hantrakul 2014) are a couple of examples where a LEAP sensor has been incorporated into a project as a way of controlling the functionality of the system. Although the LEAP sensor is not recommended for industrial use, researchers in medicine and engineering are pushing the sensor's capabilities to achieve more than it was designed for.

4.1 Hardware

The Leap motion sensor controller consists of two small optical sensors (small cameras) as well as three infrared light emitting diodes. The sensors are fixed and pointed in the y-axis direction. The infrared LEDs within the LEAP sensor are able to pick up light that is not within the visible light spectrum. Infrared light has a wavelength between 750 nm and 1 mm (Nave 2015). Both cameras have wide angle lenses. This enables the LEAP sensor to have a wide interactive area. This interactive space takes the form of an upside down pyramid. The interactive range is approximately 0.25 cm to 60 cm directly above the controller device. The range is limited by the infrared LEDs ability to pick up infrared light. As distance between a hand and the infrared LED sensor increases, it becomes harder to gather accurate information to determine hand position (Colgan 2014). The cameras and the infrared LEDs are the main hardware components and together they form the basis of the compact tracking device. The LEAP sensor is connected to other smart devices using a USB connection. The LEAP sensors inbuilt USB controller reads the data collected from the optical sensors and infrared LEDs. The data is stored within the device's local memory and is able to be accessed by using the LEAP motion software.

4.2 Software

The LEAP Motion software is a free service that enables the computer to process the raw sensor data images through advanced algorithms to produce a reconstructed image of the user's hand that the device sees in real time. The software uses tracking algorithms that use the information provided in the data to represent finger movement (Colgan 2014). Background objects such as heads, bodies and lights are excluded and filtering techniques are applied to improve the accuracy of a finger's inferred position.

The LEAP is able to be programmed using programming languages such as C#, Unity, C+, Java and Python. There are a number of attractive advantages from using Python over the other programming languages.

- Python is a robust language. It is used as a primary programming language in high demanding critical systems (Levin 2011). Dropbox, Reddit and Youtube all use Python. It is used to solve complex problems and has fast execution rates.
- Python provides programmers with a flexible and scalable language. Python is not driven by premade templates or generic APIs (Levin 2011). Therefore this language is more suited for rapid development for a large range of applications.
- Python is user friendly. With simple syntax and debugging, Python is an ideal language as the project is over a restricted time period. (Levin 2011)
- Python is free and has a large standard library.

The decision to use Python means that it is easy for any new student that has minimum programming experience to carry on the project and understand and develop good programming skills.

4.3 Application Programming Interface [API] Structure Overview

The LEAP Motion software uses an algorithm that maps out an internal model of a human hand that then provides predictive modelling and tracking. The algorithm software uses the visible parts of the hand and its past observations to calculate the most likely positions for the parts of the hand that are not currently visible (Developer.leapmotion.com 2015). Hand information in form of data, is required for processing the position of the hand. Some of this information is given in the form of vector co-ordinates.

The LEAP motion sensor uses a right- handed Cartesian co-ordinate system. The Cartesian plane is centred in the middle of the LEAP controller device and the y-axis is vertical pointing in the same direction as the sensors, therefore giving positive values for hand placement on this axis. The x axis and z axis lie on the horizontal plane. The z axis values will increase when the user places their hand close to their body and will become negative values when hand placement goes beyond the device (Developer.leapmotion.com 2015).

The API system also employs physical measure values. Distance is given in mm, time in μs , speed in $mm s^{-1}$ and angle in radians (Developer.leapmotion.com 2015).

The LEAP motion API provides access to all the available tracking information through different classes. Currently there are 29 different classes that the API reference provides (Developer.leapmotion.com 2015). The following sections provide a short description of the different classes utilised in this project.

4.3.1 Vector Class

The vector class is a functional class that is utilised for testing and research steps for this thesis project. As the name suggests, this class consists of x, y and z coordinates. Most classes in Leap Motions API make use of the vector class as the co-ordinates are utilized for developer projects and GUI applications. The vector class itself concentrates on the backward tilt (Pitch), left and right rotation (yaw) and left and right tilt (roll) of the hand or hands that the controller sees. Other properties that make up the vector class is the magnitude, magnitude squared, backward, down, forward, left, etc. There are also many methods on how the vector data can be displayed; for example, the angle of two specified vectors can be given in radians or a cross-product of two vectors will give the vector that is orthogonal to the two original vectors (Developer.leapmotion.com 2015).

4.3.2 Finger Class

Fingers are numerically classified by the leap motion software. The finger class data represents a tracked finger location inside the LEAP's field of view. This class, similar to the vector class, contains several coordinates that make up a vector. The class also contains information such as which hand the fingers belong to, the current position of the fingertip and which direction it is pointing. Each finger is permanently identified to a particular hand, therefore the angular order of the fingers will determine its identification (Developer.leapmotion.com 2015). As a hand and its fingers move in and out of the LEAP's range of view, it is possible but the fingers can be identified incorrectly. As a result the finger ID may be exchanged. Quantities that are derived from the API

output will have discrepancies and will not be uniform with finger ID and history of positions. The tracking properties such as velocity will be continuous and uniform.

The classification for the finger class is extensive but is simplified by categorizing each finger, the bones that make up each finger and the joint position between each bone. The class methods for fingers are classified by integers. The following integer code (Table 3) corresponds to the following generic finger names:

Table 3: Finger Name Code Representation

Integer	Finger Name
0	THUMB
1	INDEX
2	MIDDLE
3	RING
4	PINKY

The index, middle, ring and pinky fingers all have four bones and the thumb has only three bones. The finger bone index is referenced in a similar way using integers. The code for identifying the bone type is provided in Table 4.

Table 4: Bone Name Code Representation

Integer	Bone Name
0	METACARPAL
1	PROXIMAL
2	INTERMEDIATE
3	DISTAL

The joint positions are specified in mm from the Leap Motion Cartesian Plane origin. The code identifying the joint position is given in Table 5.

Table 5: Joint Name Code representation

Integer	Joint Description	LOCATION
0	JOINT_MCP	Metatarsophalangeal joint (knuckle of the finger)
1	JOINT_PIP	Proximal interphalangeal joint- middle joint of the finger
2	JOINT_DIP	Distal interphalangeal joint of the finger - joint closest to the tip of the finger
3	JOINT_TIP	The tip of the finger

The fingertip position and the Direction of the vectors give the Cartesian position of the fingertip and the general direction in which a finger is pointing. As mentioned previously, all fingers contain four bones that are ordered from the base to the tip. The standard anatomical naming system described above does not apply to the thumb as a real thumb has one less bone. For easy programming, the LEAP API models the thumb by including the zero-length metacarpal bone; therefore the thumb has the same number of bones as a finger. The thumb is labelled as a proximal phalanx (the third bone from the tip of the finger) and the anatomical proximal phalanx is labelled as the intermediate phalanx (the second bone from the tip of the finger) in the leap motion finger bone model.

4.3.3 Hand Class

The Hand class provides additional information about the identification of the hand position of a detected hand as well as that hand's arm and associated fingers.

The Hand class is similar to the finger class but has a few additional data attributes. The Sphere Radius and the Palm position are two attributes that were used for this thesis project (Developer.leapmotion.com 2015). Other attributes include Palm velocity, palm width, Pinch strength, sphere centre, identification, etc. The Sphere radius is a float data type that represents the radius of the sphere that fits the curvature of the hand detected (Developer.leapmotion.com 2015). The Sphere radius variable has two main advantages:

1. It is useful in estimating how far each finger is spread apart (although it is not effective in gauging what the overall size of the hand is) and
2. The curvature of the fingers can be estimated from this data.

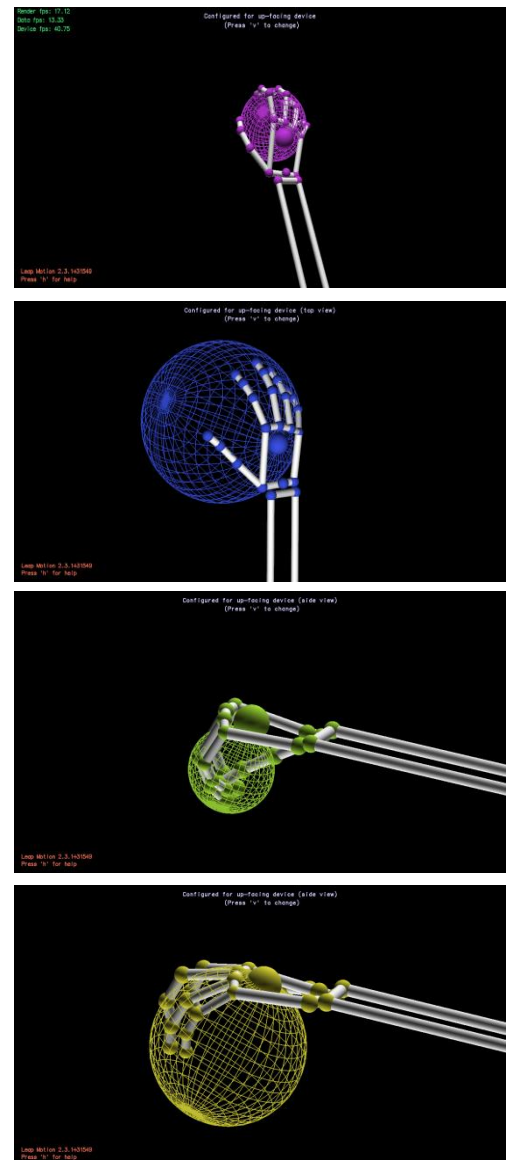


Figure 8: Diagnostic Visualiser Images of Different Sphere Radius'

The sphere is placed roughly in the palm of the hand as if it were lightly holding a ball. Therefore, the radius of the sphere decreases when the fingers curl in towards the palm of the hand. An example of different sphere radii can be seen in Figure 8. This makes programming a script easier when distinguishing between an open fingered gesture and a fully closed fist gesture.

The Palm Position is the x, y and z Cartesian co-ordinates of the hand's palm. The palm position is hence represented in a vector format. The centre of the palms position is given in millimetre in reference to the LEAP motion controller origin (Developer.leapmotion.com 2015).

4.3.4 Frame Class

The LEAP motion controller works using frames of information that are processed as soon as they are received from the LEAP controller (Developer.leapmotion.com 2015). The LEAP motion software has a fluctuating frame rate depending on processing power of the computer being used, the activity within the LEAP controller's field of view, the software tracking settings as well as other factors. A single frame represents a set of data that contains the complete set of hand and finger tracking information (Developer.leapmotion.com 2015). Fingers, tools, gestures and any other movement within the LEAP controllers field of view are represented by the frame class. The LEAP motion API allows a user to look at this data and retrieve the frames of data and even look at specific frames of data. The Frame object is essentially the root of the Leap Motion data model.

4.3.5 Listener Class and Controller Class

The Listener class is an 'event-driven' class that is registered with the Controller class. The listener class responds to various changes to the data seen by the leap controller and uses call-back functions that are used the override subclasses to respond to events that are defined in the controller class (Developer.leapmotion.com 2015).

4.4 Online Reviews

To give you, the reader and developer, a general feel for the Leap Motion as an overall general product, a handful of online review are listed below. These reviews are from users and developers who give their honest opinion.

“All in all, the Leap Motion controller is more about potential than anything else. While it provides a new means for computational control unlike anything else we've seen, it's clear that it's not cut out to replace a touchscreen or mouse as a primary input device. Not yet, anyway. Some developer may well figure out a way to take full advantage of the Leap's capabilities with a novel UI, but for now, it's best suited for creative pursuits, not productivity.” (Gorman 2015)

-engadget

“Intriguing, Unessential, but Promising. Leap Motion's Kinect-like PC motion controller has its moments of magic, but right now it's more toy than productivity tool.” (Stein 2013)

-C|net

“The Leap Motion Controller is a Piece of sci-fi futurism available today, and it's cheaper than you think. But while it's magic when it works right, it's maddening when it (frequently) doesn't.” (PCMag Australia 2013)

-PC

From these reviews and from personal usage, it can be concluded that the Leap Motion sensor is a good device but has not been perfected for precision applications. This is a primary concern as the controller is desired to be reliable and accurate. Considering that the controller is for a wheelchair and intended for individuals with limited dexterity and capabilities, safety and user experience is crucial for the design and dictates the engineering methods employed.

5. Machine Learning

Machine learning (Schapire 2008) is a subclass of computer science that is linked to artificial intelligence. By definition, machine learning is the study of pattern recognition (Bishop 2016) and computational learning (WhatIs.com 2015); in other words, it has the ability to learn and evolve without being explicitly programmed. Machine learning software is able to learn autonomously with the aim being to improve the development of computer programs. This is achieved through analysing data in order to construct an algorithm that can learn and make predictions. When these algorithms are exposed to new data, they are able to change and grow, formulating new predictions and algorithms based on the static or dynamically fed data. Machine learning is used mainly for optimisation and, in industrial scenarios, used for predictive modelling. There are many applications of machine learning including, adaptive websites, game playing, handwriting recognition, internet fraud detection (Quora.com 2016), search engines and software engineering.

5.1 Data mining

Similar to machine learning is data mining. Both machine learning and data mining methods analyse sets of data for patterns that can be applied to optimisation or predictive algorithms (Docs.oracle.com 2015). The difference between data mining and machine learning is that data mining software applications analyse data collected from a device and present the patterns in a way that is understandable for human comprehension; while machine learning uses the collected data to improve the software programs own understanding (Quora.com 2015). For example, advertisements online change to what the user's interests are through machine learning.

Generally the mathematics involved in data mining software programs are decision trees (Saedsayad.com 2016), cluster analysis (Learn Data Mining 2016) and various classification and regression analysis techniques (Docs.oracle.com 2016). There are many data mining open source

software programs available that include the following: H2O, Yooreeka, WEKA. Other data mining software that is commercial software include: MATLAB, IBM SPSS Modeler, Oracle Data Mining.

5.2 Waikato Environment for Knowledge Analysis [WEKA]

WEKA is a machine learning software developed at the University of Waikato in New Zealand. It is an open source software that is able to discover patterns in large data sets as well as being able to extract all the information and represent it in a useful way (Community.pentaho.com 2015). The software provides large selection of mathematical models for data analysis and predictive modelling. WEKA supports an array of data mining tasks such as clustering, regression, visualization and data processing. The machine learning techniques utilised by WEKA are based upon the assumption that the available data is of a single flat line or relational format. These formats describe each data point by a fixed number of attributes, normally numeric methods are utilised.

There are many algorithms implemented in WEKA: linear regression, model trees, decision trees, support vector machines, neural networks, and other. Most of them produce complicated models that are hard to translate into code. A very simple model is produced by decision trees. The classification accuracy of the decision trees generated by the WEKA implementation is very high. For this reason, this project will only use this algorithm.

5.3 Decision Trees

One of the most famous machine learning algorithm is Quinlan's C4.5 (Quinlan 2014) for generating a pruned or unpruned decision tree (Quinlan 2014). The WEKA software's version of this algorithm is called J48 (Facweb.cs.depaul.edu 2016). The decision trees generated by J48 are used for classification.

The algorithm works by examining the normalized information gain that results from choosing an attribute for splitting the data. The attribute with the highest normalized information gain is used.

The splitting procedure stops if all instances in a subset belong to the same class (Quinlan 2014) and a leaf node is created in the decision tree associated with that class. It can also happen that none of the features give any information gain. In this case J48 creates a decision node higher up in the tree using the expected value of the class.

The algorithm also performs pruning of the generated tree. The pruning is carried out from the leaves to the root. If the estimated error of a subtree is comparable with just using a leaf instead, then the tree is pruned at that branch.

6. Ergonomics and User Experience

This project aims to accommodate the user's needs. If the user is not able to adapt and benefit from a gesture control system because it is too difficult and unpredictable to use, then the system becomes useless and unnecessary. The LEAP motion controller prides itself on being a more natural and intuitive way for people to interact with computers (Sanders and Flowers). With that in mind, controlling something with subtle finger and hand movements in a 3 Dimensional space is still a foreign concept and may not be intuitive for a user to learn.

Like other household motion control devices (for example the Xbox Kinect), interacting in a new way will require practice and an understanding of how to interact with the hardware system for a new user.

The project design has a few rules of thumb when considering the user experience for the LEAP enabled application to be easy to learn and use.

6.1 Easy Gestures

The symbology must be intuitive, easy to learn and memorise. Complex hand gestures that are physically difficult can be discouraging for the user. Physically inspired gestures from real world behaviours that are more natural means less training to learn individual needs.

6.2 Exaggerated Responses and Dynamic Feedback

The response to a gesture should be exaggerated as the user should feel as if their intent is amplified for an even more positive result; for example, the small movement of a physical computer mouse should significantly move the computer cursor (Sanders and Flowers).

The more responsive feedback given to the user, the better interaction experience they will have as they will be able to interact more with the LEAP system.

6.3 Intuition

A user should be able to start to operate the system with little or no instruction. Intuitive guesses and common sense tie in to natural human behaviour, therefore using gestures that will come to the user naturally are important. Confusing logic will only discourage an individual from progressing and using the device controller.

Simple things like making all the gestures have the same orientation to the device make a significant difference to a new user as it keeps uniformity.

6.4 Resources

Any good product is accompanied by a set of clear instructions as well as online help and information. Products that are transparent yield the most positive feedback. A working knowledge of the overall system benefits the user as they will be able to operate the controller to its full capability and be able to troubleshoot errors that may occur.

7. Project Design

This project aims to control the basic functionality of a motorised wheelchair through hand gesture recognition techniques. The basic driving commands of a wheelchair include forward, reverse, left turn, right turn and stop. Being able to perform a 90° or 180° rotation to the left or right as well as a 360° turn, is also a standard feature. As well as these basic functions, the user of a wheelchair is able to change the speed at any given time. These combined features make up the basic command set of a wheelchair, therefore this project aims to distinguish a sufficient set of gestures to provide full control of a wheelchair.

The overall project design can be split into three separate areas:

1. Hand gesture controller design interface
2. Controller function and program design, and
3. Robotic testing platform

Each area requires a design that will be robust and have accuracy and precision. Safety considerations and features must be included at each stage of the project.

This project will not go to the extent of implementing the system on a wheelchair as this will be too costly, therefore this project is a proof of concept.

7.1 Hand gesture Controller Design

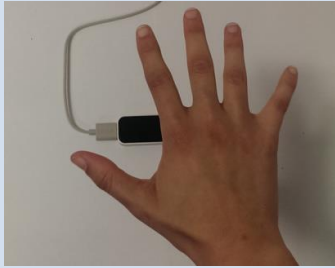
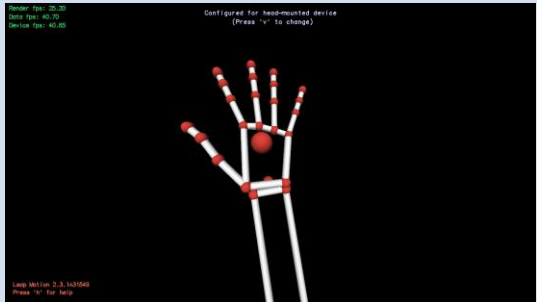





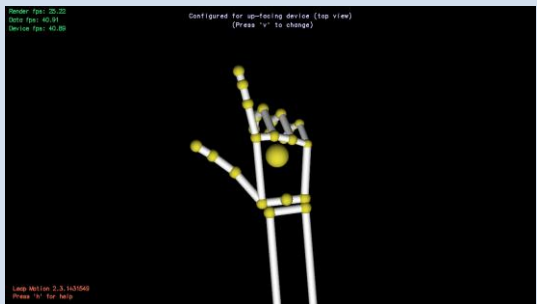

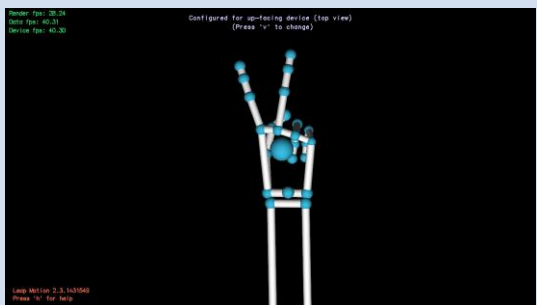
There are five general functions that are performed by a wheelchair, hence there will need to be five or more separate hand gestures that will represent each different function. As the LEAP controller is for a wheelchair, it is assumed that the controller is to be used by a person who has very limited shoulder, arm and hand dexterity and is prone to fatigue. Big gestures that involve movement and tracking (such as a swipe, circle or flick gesture) will be hard to manage, hence the use of these gestures in this project is eliminated. The hand gestures used will be mostly based around different finger positions.

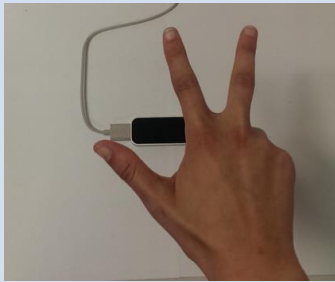

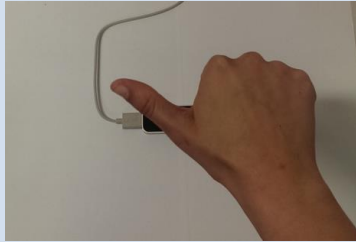
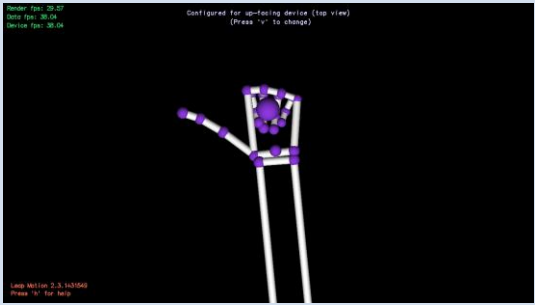

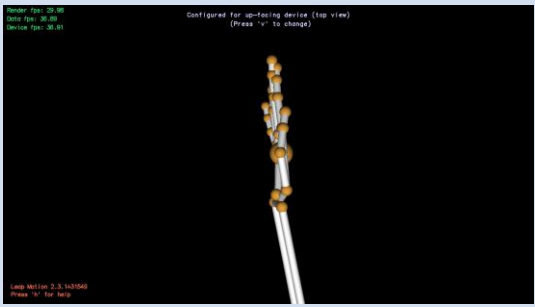
As only one hand is needed to operate the controller, there is no need for the program to be able to distinguish between the right hand and the left hand. A flaw with the LEAP motion sensor is that it sometimes struggles to distinguish between right and left hands. Robust hand gestures are better suited to eliminate as much unpredictability as possible, hence symmetrical hand gestures are preferable as they eliminate the possibility of the LEAP motion sensor confusing the left hand as the right hand and vice versa. Choosing gestures that eliminate this confusion was essential for safety reasons and reliability as it decreases the risk that the LEAP motion sensor will mistake a right hand for a left hand gesture and carry out an undesirable wheelchair command.

Above all, the gestures have to be user friendly and memorable. Gestures that are hard to perform will discourage people from using the controller.

With the above factors in mind, the following eight gestures were chosen; their names and representation are illustrated by photos in Table 6. The gestures are described by their finger position. The LEAP motion sensor is placed in a neutral position with the sensors facing directly upwards. Each hand gesture is placed approximately 15 cm or more directly about the LEAP sensor in a central location.

Table 6: Hand Gesture Representation

Hand Gesture Name	Photo	Leap Sensor Skelton Image: TOP View
Open Arm- Palm facing the LEAP. All fingers and Thumb are fully extended and separated		
Full Fist- Palm facing the LEAP. All fingers and thumb are curled		
Thumb and Pinky- Palm facing the LEAP. Thumb and Pinky finger are extended and three middle fingers are curled		
Thumb and Index- Palm facing the LEAP. Thumb and Index finger only are extended and separate. The other fingers are curled		
Index and Middle- Palm facing the LEAP. Index and Middle finger are extended and separate. Other fingers are curled.		

<p>Thumb, Index and Middle- Palm facing the LEAP. Thumb, Index and Middle fingers are extended and separate. Other fingers are curled.</p>		
<p>Thumb Only- Palm facing the LEAP. Only the thumb is extended. All fingers are curled.</p>		
<p>Axe- Palm facing towards the left (x- axis). Fingers slightly curled in comfortable position. Slight finger separation.</p>		

Each hand gesture has their own unique Cartesian and vector representation when placed inside the LEAP motions range of view. To accurately define each gesture, access to the LEAP sensor data is needed so that it can be analysed. Sections 6.1.1 and 6.1.2 detail how the data is represented, how it was collected, how it was processed and the concluding results from the testing of gestures.

7.1.1 Leap Data Sampling Application: LEAP Listener

As part of the hand gesture background research for the controller, numerous tests were done to collect data. The LEAP controller works by taking a pre-existing model of the human hand and compares this model with the data points collected from the sensors and updates in real time. The data is analysed frame by frame by the LEAP Motion software. The LEAP motion software SDK files provided a basic LEAP Listener application program located in the LEAP sample folder. This program was used as a template and then modified to collect the required hand frame data and export it as a text file. The vector, finger and hand classes discussed earlier were required to be added to the LEAP listener to collect the data from the LEAP motion sensor. Each frame of data consists of the roll, Pitch and yaw angles of the hand as well as the bone direction vector coordinate data points for five fingers. The index, middle, ring and pinky fingers have four bones each and the thumb has three bones. Each bone will have three co-ordinates, therefore, there is a total of fifty seven (57) data points. Overall, including the Pitch roll and yaw angles, there will be sixty data points per frame of data. An example of a single frame of data is given in Appendix B.

The modified Leap Listener program was used to conduct batch and continuous testing for each hand gesture. Over one hundred tests were conducted for each gesture with the same right hand. The test data was then organised into a singular data document with separate documents for each gesture. These test data files are available upon request. The number of collected data points was too large to visually interpolate any useful information, thus a data mining and machine learning analytic program called WEKA is employed and its results discussed in the following section.

The modified Leap listener program application is provided in Appendix A.

7.1.2 WEKA Results

The LEAP controller is to be used to drive a wheelchair, which means safety is a top concern, therefore, it is desirable to make the LEAP controller system perform as accurately as possible. The high level of accuracy is only achievable through testing, collecting the frames of data, analysing it and then using the outcome in the software that will predict the hand gestures to drive the motors.

WEKA consolidated and analysed all the frame data using the J48 decision tree algorithm. It was able to handle the large amounts of data as well as the large classification range that the LEAP listener sample program was instructed to print. Using the WEKA program provided a more structured approach for analysing the LEAP data, yielded better results and was more systematic than a mere trial and error method.

WEKA produced a classifier model from the data. This model provided information about every arm and bone direction. This information was used to classify the structure of each hand gesture. The testing segment of this thesis project, discussion in section 7.1, has been instrumental in understanding how each gesture is classified.

Initially, a model that will classify all eight gestures as described in Table 6 was generated using WEKA. WEKA provided statistics on each gesture. The classification matrix is given in the following table (Table 7). This model yielded 81.67% classification accuracy.

Table 7: Confusion Matrix for all Hand Gestures generated from WEKA

open arm	full fist	index and thumb	index and middle	index and thumb and middle	thumb only	axe	thumb and Pinky	classified as
5528	11	0	1	21	1351	967	0	open arm
0	2629	19	0	7	0	1	2	full fist
0	1	2693	0	14	0	0	0	index and thumb
0	0	21	2571	4	0	0	1	index and middle
1921	11	0	0	1267	0	0	0	index and thumb and middle
5	21	2	0	0	3270	0	0	thumb only
707	6	0	0	0	0	2086	0	axe
0	7	1	0	0	0	0	2694	thumb and Pinky

Three of the gestures are most frequently misclassified: Axe was sometimes classified as open arm, the index, thumb and middle gesture was misclassified also as open arm, and the open arm gesture was misclassified as the thumb only gesture. Table 8 contains the detailed accuracy by gesture. As we can see the axe and the index, thumb and middle are the two gestures that have the lowest classification accuracy. Each hand gesture has a true positive (TP) percentage, identifying the percentage of hand gestures that were identified correctly, and a false positive (FP) percentage that shows how many times the correct hand gesture was identified as incorrect.

Because we are interested in robust gestures, these two gestures were removed. More data was collected using only the open arm, fill fist, index and thumb, index and middle, thumb and pinky and thumb only gestures. The precision column shows the fraction of the total correctly retrieved

identified hand gestures to the actual relevant correctly identified hand gesture. The receiver operating characteristic (ROC) percentage is an indication of how well the decision tree classified each hand gesture.

Table 8: Detailed Accuracy by Gesture Type

Hand Gesture	TP %	FP %	Precision %	ROC %
Open_Arm	70.2	13.2	67.7	92.0
Full_Fist	98.9	0.2	97.9	99.9
Index_and_Thumb	99.4	0.2	98.4	100
Index_and_Middle	99.0	0	100	100
Thumb_Only	99.2	0.2	98.6	100
Thumb_and_Pinky	99.7	0	99.9	97.7
Thumb_Index_and_Middle	39.6	5.5	48.4	93.2
Axe	74.6	0	99.9	100

Table 9 shows a summary of results that give the total number of data sets provided to WEKA and the amount of times the J48 algorithm was able to correctly or incorrectly classify the data. The percentage of correctly classified instances was 97.16%. Ideally, the greater the correctly classified instances, the better the LEAP controller system will operate.

Table 9: Summary of Total data accuracy results formulated from WEKA J48 Algorithm

Total Number of Instances	112388	
Correctly Classified Instances	109198	97.16%
Incorrectly Classified Instances	3190	2.84%

A ROC value is also shown in Table 10 for each hand gesture. The ROC value represents the probability of a positive instance occurring more often than a negative one. The ROC values for

each gesture are very high, indicating that, from the data collected, there is a greater chance of the gesture being identified correctly.

Table 10: New Detailed Accuracy of Gesture: Generated from more collected Data

Hand Gesture	TP	FP	Precision	ROC
Open_Arm	0.984	0.007	0.976	0.998
Full_Fist	0.973	0.012	0.957	0.996
Index_and_Thumb	0.982	0.004	0.979	0.998
Index_and_Middle	0.98	0.002	0.992	0.998
Thumb_Only	0.924	0.007	0.942	0.994
Thumb_and_Pinky	0.964	0.003	0.978	0.998

A confusion matrix provides a visualization of the performance of the J48 algorithm for the selected gestures. The confusion matrix in Table 11 shows the actual number of instances that a hand gesture was confused with a different hand gesture. The columns are the classified hand gestures and the rows are the actual hand gestures. As a general overview, most hand gestures had been confused with two or more different hand gestures on more than one occasion. The thumb only gesture and the full fist gesture are the most likely to get confused with one another.

Table 11: Confusion Matrix for Selected Hand Gestures generated from WEKA

Open Arm	Full Fist	Index and Thumb	Index and Middle	Thumb Only	Thumb and Pinky	
23179	127	52	5	114	75	Open Arm
145	23493	38	6	393	60	Full Fist
18	146	18622	24	78	71	Index and Thumb
67	168	107	20150	70	0	Index and Middle
193	474	146	122	12079	54	Thumb Only
15	129	63	0	88	11675	Thumb and Pinky

The results determine what type of gesture will be used to drive the wheelchair. When WEKA generated desirable statistics, the more robust the gesture was and the less chance the program would mistake a gesture and carry out an undesired and potentially dangerous function. In summary, the axe gesture and the thumb, index and middle gesture did not have good results; therefore the use of these gesture are no longer justifiable.

This testing and data results are a key outcome of this thesis. It not only determined which hand gestures to use but also confirmed that the LEAP motion sensor was a viable option as the hand gesture device used for this application.

7.1.3 Conclusion of Hand Gesture Design Process

After the testing and analysing of results, a total of six gestures were chosen. These gestures were then allocated a task (see Table 12 for details) that represents the functions of a standard motorized wheelchair. The tasks allocated, are aimed to match intuitively and logically to a gesture. For example, the index finger can be assumed to be a Pivotal finger as the addition of a thumb or middle finger will indicate a left or right turn respectively.

Table 12: Hand Gesture Wheelchair Function Task Allocation

Description	Application Task
Open_Arm	Start, forward, Incrementally increase speed.
Thumb Only	Reverse
Index and Middle	Right Turn
Index and Thumb	Left Turn
Thumb and Pinky	Stop Immediately
Full_Fist	Controlled Stop

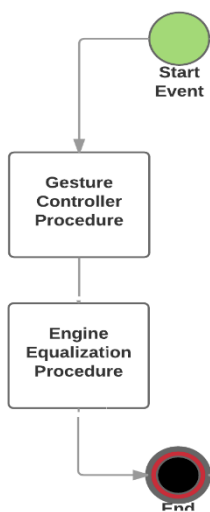
Although the chosen gestures seemed suitable for the accuracy needs of the controller, each still had wild variations from one trial to the next, which were visible through the LEAP diagnostic visualizer as well as WEKA statistics. Although the gesture tests were conducted under the same conditions, the sensor was not completely accurate 100% of the time.

7.2 Server Application Description

The complete server application code is provided in Appendix C. The application is built on top on the LEAP listener sample (Developer.leapmotion.com 2016) provided by the Leap Motion API as it has predefined call back functions that are used to respond to events dispatched by the LEAP controller. First a native listener for the LEAP motion sensor is initialized. Then, just like in the data sampling process, the server application collects all the measurements that the sensor produces. The Server Application stores these measurements, and then passes this information to a predictive function. This function is generated from the WEKA J48 algorithm. The tree that the algorithm produces is translated into if-then-else statements. This gives a very fast predictive function since the tree has very small depth of less than 20 nested if statements.

The predicted hand gesture, (one of the following gestures: open arm, full fist, index and thumb, index and middle, thumb and pinky and only thumb), is then sent as a UDP packet to the receiving end of the test platform. The leap driver application that runs on the Raspberry Pi interprets the data packet, reads the gesture being sent from the server application and then drives the motors accordingly.

7.3 Controller Function Program Design



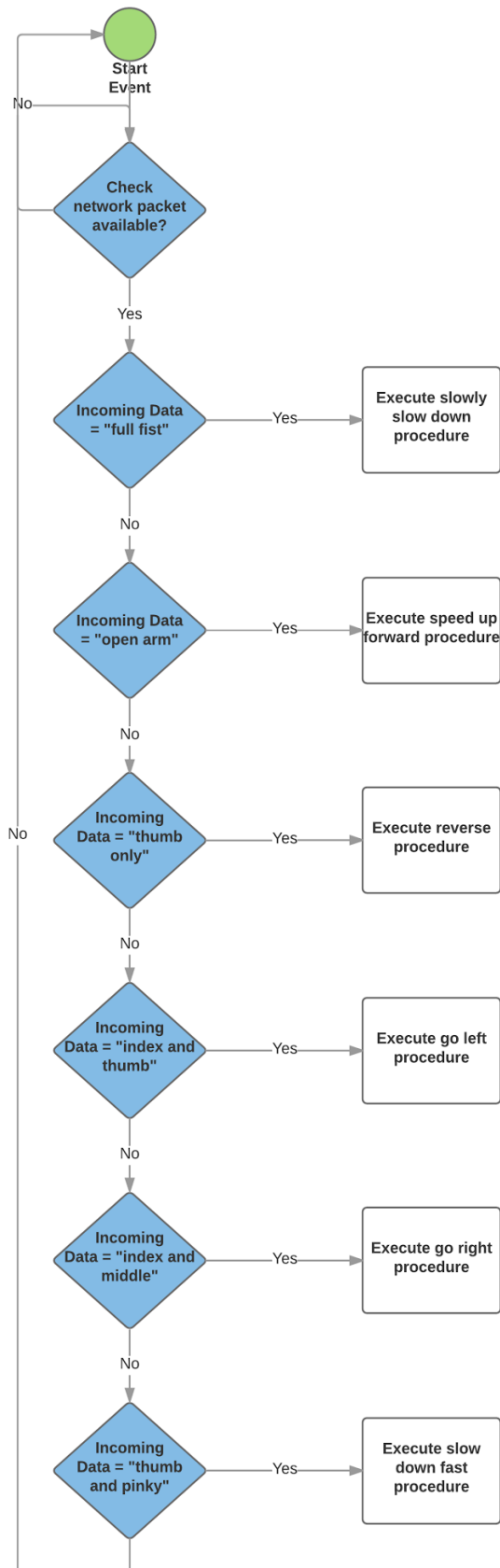
The aim of the controller function programs are to respond to hand gestures and move the wheelchair motors appropriately.

The diagram shown in Figure 9 gives an overall representation of how the program is intended to operate. When the server application program recognises a gesture, an event will start. The event will then go through a gesture control procedure to determine what action the wheelchair motors should take. After the gesture control procedure, a wheelchair 'engine equalization' procedure will start as a way to keep extra control on the

Figure 9: Overall System Flow Diagram.

wheelchair's movements (stop it from Spinning out of control). The controller function program will continuously be on standby waiting to register the next gesture until the program is told to end.

Gesture Controller Procedure



The gesture controller procedure is illustrated using a diagram as shown by Figure 10. When the program starts, it will continuously wait for a packet of data to be received. A hand gesture will be seen by the LEAP motion sensor and the frame data will be passed through the network socket. When the program receives an available network packet it will match the data with a defined gesture. The program will then proceed to carry out the intended actions using the motors.

If the same hand gesture information is being provided continuously through the LEAP sensor, then the program will persist to carry out the same action. For example, if a person were to steadily hold the same index and thumb gesture in the LEAP controller's range of view, the wheelchair will continue to turn to the left. The same applies for all other gestures.

The open arm gesture will execute a procedure that will gradually increase the duty cycle of the PWM switching signals. The longer the open arm gesture is held, the greater the speed of the wheelchair. When the open arm

Figure 10: Gesture Controller Procedure Diagram

gesture is removed and no other gesture is registered, the wheelchair motors will continue to operate at that speed.

The speed of each motor of the wheelchair is limited. The restricted maximum speed limit is a safety technique so that the person using the wheelchair is safe, comfortable with the speed and the motors of the wheelchair are protected from potential damage.

The Drive Engines procedure is briefly illustrated by a diagram in Figure 11. This procedure dictates the speed and direction of the motors when a hand gesture event is registered. When no hand gesture is being recorded, no PWM signal will be sent to the motors and both motors will be off. When a hand gesture starts an event, and the speed of one or both of the motors will increase. The rotation of the motor shaft will be in a clockwise or anticlockwise direction indicating forward and reverse movement respectively.

Drive Engines Procedure

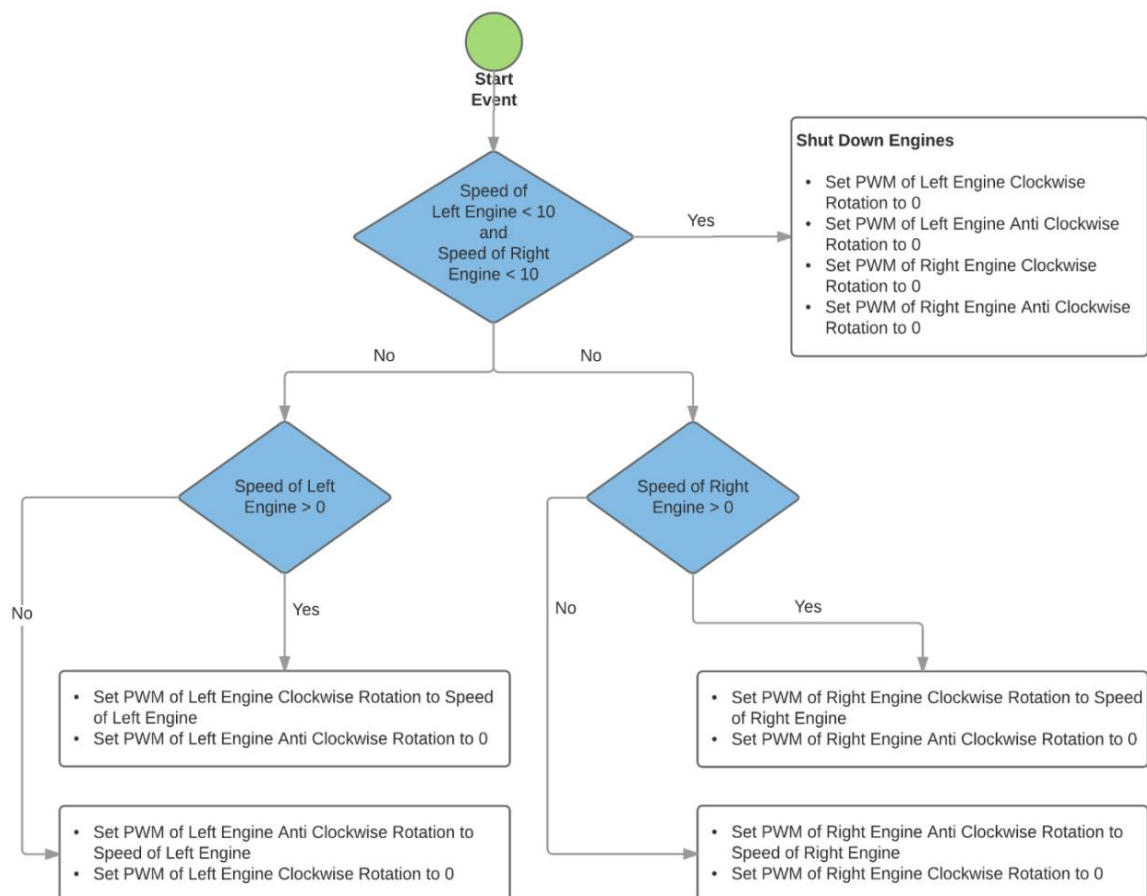


Figure 11: Drive Engine Procedure Logic Diagram

The left and right procedures (Figure 12) shows how the two different motors will react when and

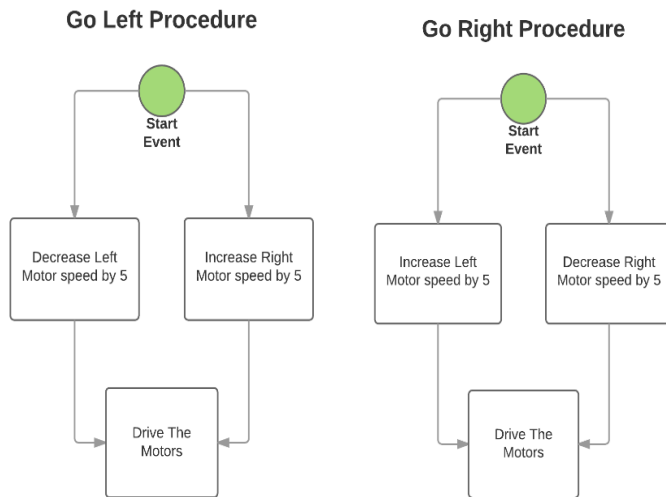


Figure 12: Index and Thumb and Index and Middle Flowchart procedure Representation

Index and middle or Index and thumb gesture is seen by the LEAP controller. It is desired for the wheelchair to turn on the spot. In general, depending on which direction to turn, one motor will speed up in a clockwise direction to move forward and the other motor will speed up, but in the opposite anti-clockwise direction. The speed of the motors are increased so that turning will be more efficient.

The Slow Down Fast Procedure given in Figure 13 is a diagram representing the thumb and Piny hand gesture triggered event. For an abrupt or emergency stop, the speeds of the motors are decreased incrementally.

Slow Down Fast Procedure

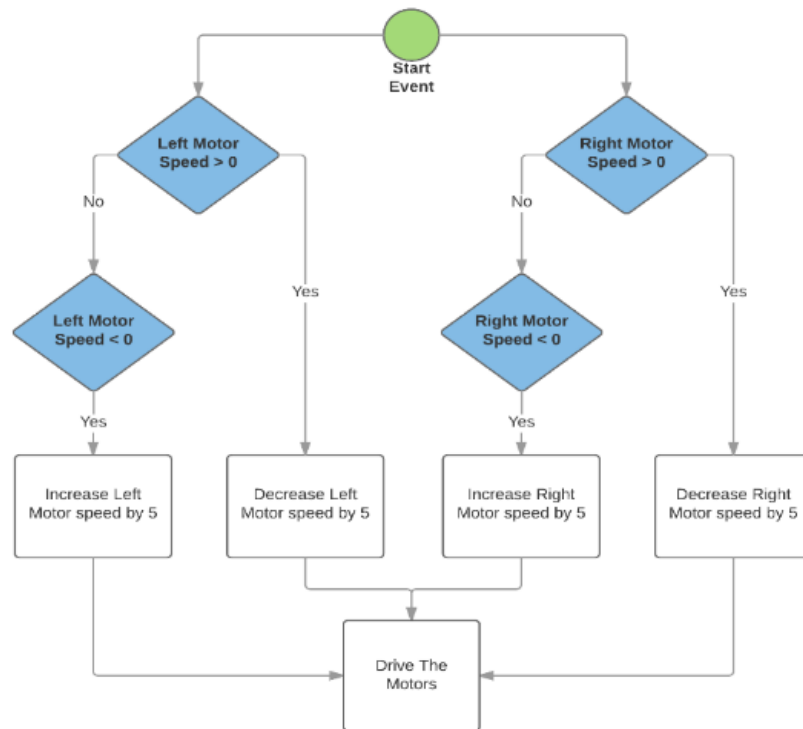


Figure 13: Thumb and Pinky Gesture Flowchart Procedure

Engine Equalization Procedure

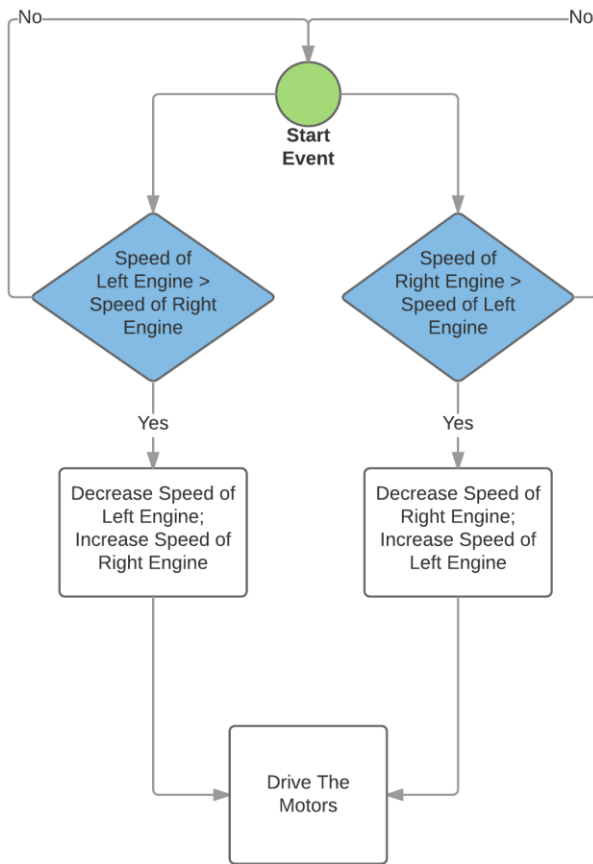


Figure 14: Engine Equalisation Procedure for Motor Speed Stabilisation

The Engine Equalization Procedure is aimed at stabilizing both the motors speed. The procedure diagram shown in Figure 14, is a comparison method that compares the speed of the two motors against each other. After a turning gesture, the motors are receiving two different PWM frequency and are rotation in opposite directions. If no gesture command is being received once the turning gesture has ceased, then the motors speed will gradually start to equalise. For example, the index and thumb gesture is used to turn left. To turn left, the speed of motor one will increase in speed and the speed of the second motor decrease and then speed up again in the opposite direction. When LEAP controller stops receiving

the index and thumb hand gesture, and no other hand gesture is registered, it is desirable for the wheelchair to continue forward in a straight direction. Therefore, once the turn has been made the motors' speeds are compared and are incrementally changed so that they rotating at the same speed in the same direction.

The motor driver program is provided in Appendix D.

7.4 Robotic Testing Platform

There were many ways to test working code. The following are some options that were considered:

- Simple breadboard with LEDs for indications
- Graphical User Interface (GUI) application
- Robot

All the individual options would have been effective visual aids, but to illustrate the controller's capability, a working robot was best suited considering it was for a wheelchair application. One of the main reasons behind this decision was that the speed variations and control of the motors of a robot mimicked that of a motorised wheelchair, making it easier to visualise in comparison to a GUI or through mere static indication methods.

Design possibilities for the robot were limited by the following factors:

1. External power source: it is desirable to have to robot mobile; therefore the battery source had to be reasonably light weight and therefore slightly limited the range of electronics that could be used. and
2. Wireless communication: the LEAP controller will only work on a computer with an Intel i3, i5, i7 or AMD processor and a graphics card, therefore the commands will have to be communicated to the robot through radio or Wi-Fi.

It was required to control two separate low voltage DC motors to drive the robot and make it perform different functions exactly like one would see a standard motorised wheelchair perform. However, a two wheeled robot base large enough to hold two batteries and a controller was not available, so a standard 4 motor robot base development kit was purchased. The four motors will act as two units, meaning the two left side motors will receive the same commands and similarly with the two right side motors.

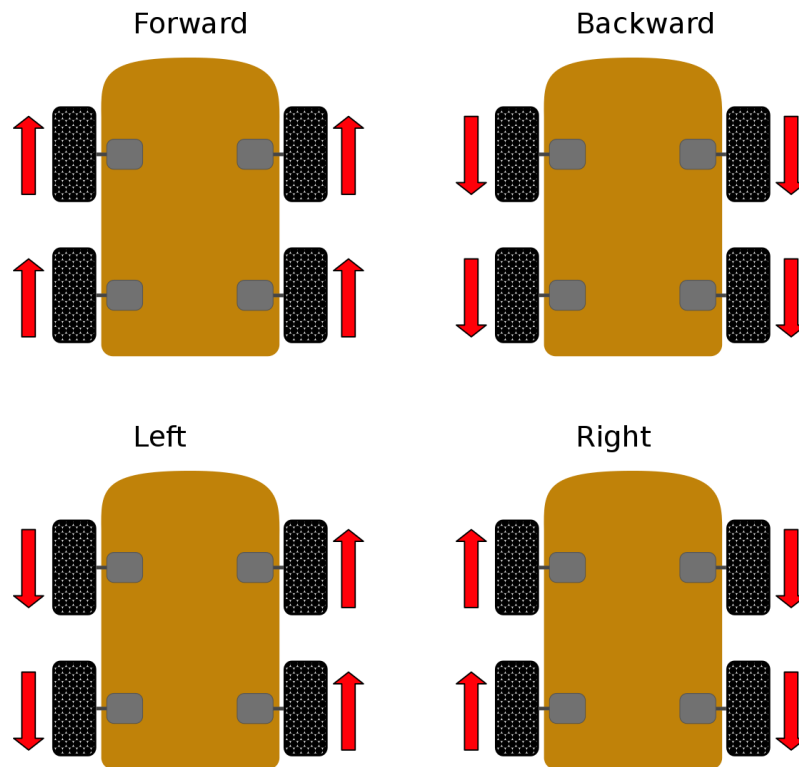


Figure 15: Diagram Representation of Test Robot Motor Rotation Direction

Using a Raspberry Pi, USB wireless dongle, L298N Arduino motor driver shield (Geekonfire.com 2015) and four motors, the test product was complete. The directional spin of the motors is indicated in Figure 15. A photo of the final test robot platform is shown in Figure 16 and representation of this circuit is illustrated in Figure 17. The Pin allocation is provided in Appendix

E.

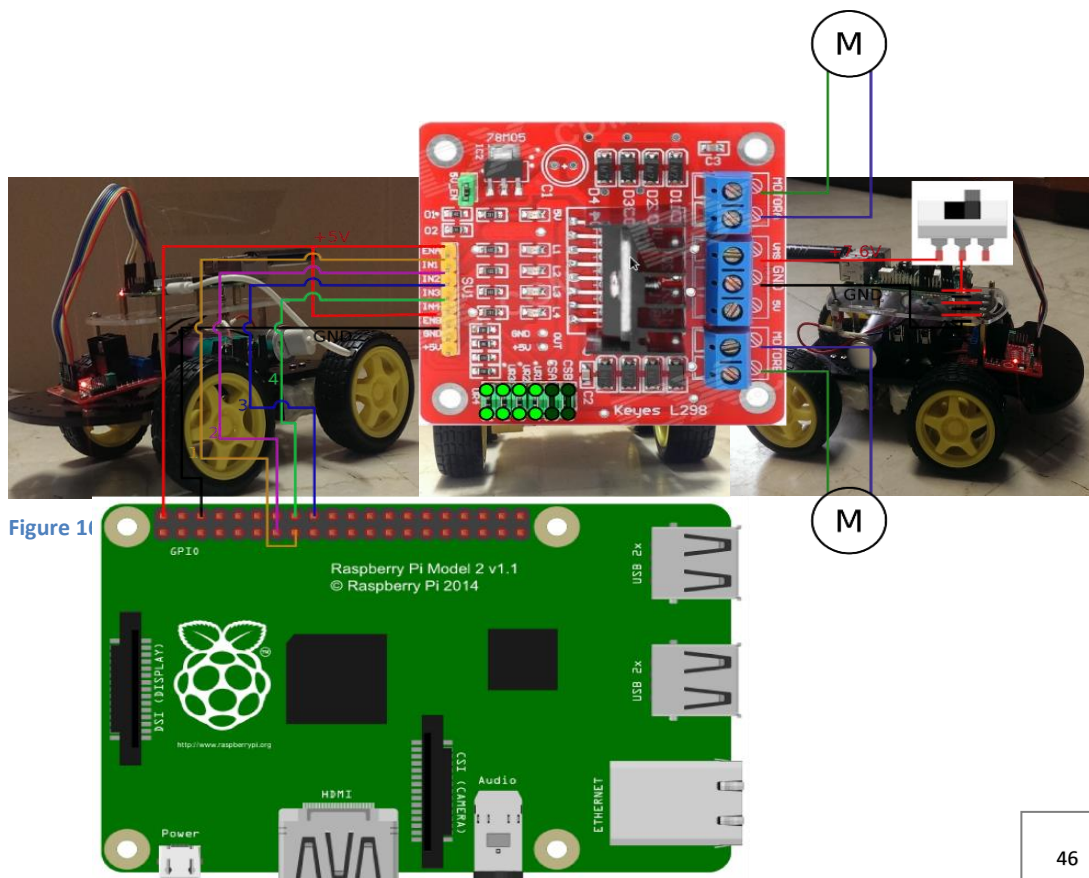


Figure 16

Figure 17: Wiring Diagram of Robot Test platform

Take note that the Raspberry Pi was version 2.0, as the Pin configuration will vary in different Pi models (Raspberry Pi 2015). The Raspberry Pi needs its own 5V, 1A power supply, so a mobile battery bank with a USB to micro USB connection cable was used as it was relatively compact and convenient to recharge. The Raspberry Pi communicated to the host computer through Wi-Fi. A Wi-Fi USB dongle (Dlink.com.au 2016) was inserted in one of four USB ports readily available on the Raspberry Pi. The dongle was configured and given an IP address so that it could receive packets of data from the host computer. The Raspberry Pi has a microSD port which is used for loading the Wi-Fi dongle configuration code and motor driver code to the Pi as well as storing data.

The Arduino motor driver PCB is supplied with 7.6V DC from two 3.8V Lithium Ion batteries in series. The power from these batteries will be used to drive the four 3V rated DC motors. The board has a L298N motor driver chip (*Dual Full-Bridge Driver* 2000) commonly used for controlling

robotics and other mechanics. The L298N chip is a high voltage, high current dual full H bridge driver capable of driving brushed DC motors or a four wire, two phase stepper motor as well as other inductive loads such as relays (L298N et al. 2015). This motor driver shield was selected for this project for numerous reasons. The L298N chip is pre-mounted onto a shield with its input going out to terminals on the shield which made it simple to change or troubleshoot when needed. The shield has an on-board voltage regulator and indication LED for each input (Geekonfire.com 2015). The main feature is that it is able to drive motors simultaneously while the motors are receiving commands to go at a different speed or change rotational direction. Additional specifications for the Arduino motor driver shield are provided Appendix E.

The code that is responsible for driving the motors and determining their rotation direction and speed is provided in Appendix D. This code defines the functionality of the robot (turn left, turn right, forward and reverse) by setting specifically allocated Raspberry Pi general input/output to the motors true or false. Pulse Width modulation (PWM) has been included allowing the duty cycle to change which in turn determines the speed of the motors.

To recreate the robot testing platform, a bill of materials (BOM) is provided in Appendix E.

8. Test Result

The testing of the overall system was relatively conventional and consisted of carrying out a series of repetitive tests. As this project did not have ethics approval from the required organisation, the tests were limited to a few willing participants. An explanation of how the controller works was verbally communicated to the participant before commencing any testing. The participants were asked to test the controller system's usability by means of some simple tests and a general free run of the controller so that the general feel for the controller could be assessed.

There were two main categories that the gesture controller system was being marked against: accuracy and user friendliness.

User friendliness was given through general feedback from individuals. In summary, the leap gestures were intuitive but it took practice to grasp the responsiveness of the system. It was not easy to navigate the test robot straight away on the first try. The novelty of a gesture recognition controller wore off as some individuals became frustrated with the lack of controller sensitivity.

The participants involved captured a large age group as the age range was started from 8 years old to 69 years. All participants were asked to try and complete the following instructions:

- Test 1: move forward
- Test 2: Stop: gesture of your choice
- Test 3: reverse
- Test 4: turn left or turn right 90 degrees and continue with straight forward movement :
participant's choice of left or right turn
- Test 5: complete a full circle on the spot then stop: direction is participants choice

Each participant was given three attempts to complete the task. The results are shown in Table 13 that indicate if the participant was able to complete the task in one of the three attempts.

Table 13: Results if the tasks were able to be completed

Participant	Test 1	Test 2	Test 3	Test 4	Test 5
1	Yes	Yes	Yes	Yes	No
2	Yes	Yes	Yes	No	No
3	Yes	Yes	Yes	Yes	No
4	Yes	Yes	Yes	Yes	No
5	Yes	Yes	No	No	No
6	Yes	Yes	Yes	Yes	Yes
7	Yes	Yes	Yes	No	No
8	Yes	Yes	Yes	Yes	No
9	Yes	Yes	Yes	Yes	Yes
10	Yes	Yes	Yes	No	No
11	Yes	Yes	Yes	No	Yes
12	Yes	Yes	Yes	No	No
Percentage	100%	100%	91.67%	41.67%	25%

It is clear from the participant testing that harder tasks are not easily achievable straight away as the rate of correct movements decline as difficulty rate increases.

Almost all participants were able to complete the first three simple tests, start, stop and reverse. Turning required more skill. As there was a small amount of latency and the testing platform was not build to turn easily, turning left or right required some practice and guess work as to when would be the best moment to release the hand gesture making the robot motor equalisation sequence take over.

9. Problems Encountered

Each stage of this thesis project encountered some minor problems that initially cause some setbacks. Through further development and troubleshooting all of the problems were resolved. Noteworthy problems that were encountered are briefly discussed and the ways these issues can be resolved.

9.1 LEAP motion Connectivity

Problem: The LEAP motion sensor would not register hand gestures. It was observed that the LEAP motion was receiving power from the computer as the green power light was on, but the small infrared sensors were not observed to be on. The LEAP motion diagnostic visualizer would not display any hand skeletons, hand tracking or photo captures. The cause of this re-occurring error was unknown and would happen on any computer.

Solution: Disconnecting and reconnecting the LEAP motion sensor did not resolve the issue. The LEAP Motion control Panel offers some troubleshooting techniques. When this problem occurred, recalibrating the device through the 'Re-calibration Device' option located in the LEAP motion settings, would force the infrared sensors on. The

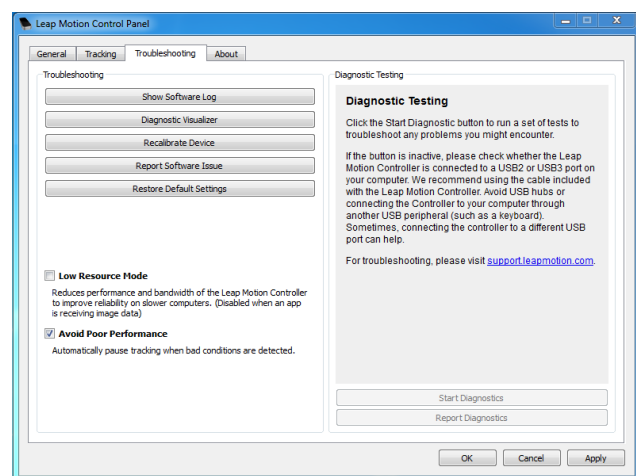


Figure 18: LEAP motion Troubleshooting Tab in Settings-Recalibration

diagnostic tool would then give instruction on how to recalibrate the LEAP sensor. Further troubleshooting techniques for the LEAP motion controller can be found on the frequently asked questions section of the LEAP motion Support web page (Leap Motion Support 2015).

Restarting the host processor was the most reliable option.

9.2 Robot Test platform Troubleshooting

Problem: Initially there were minor issues with the robot car testing platform. For example the motors would not rotate in the required direction or not rotate at all. There were also instances where the Raspberry Pi would appear not to be communicating with the server application.

Solution: Physical layer troubleshooting fixed most problems. Checking the wiring found that sometimes the female to male connection would be accidentally pulled apart. The battery supply may also be causing issues if it is running low on power. For any other problems, a 'test motors' program was created to individually test the clockwise and anticlockwise rotation of each motor in sequence. This was extremely useful for troubleshooting errors.

9.3 Latency Issues

Problem: On occasion, latency of controller system was noticeable. The frame rate per second of the LEAP motion sensor was very low and as a result the controller system became unresponsive.

Solution: This is an unfortunate flaw of the LEAP motion sensor. The frame rate per second of the LEAP motion sensor cannot be set manually or to poll the LEAP motion for frame data as the programs would be alternately tested on desktop computer and laptops with different capabilities. Polling for frames of data risks receiving the same frame twice consecutively if the application frame rate exceeds the LEAP frame rate. The frame rate of the LEAP controller can be anywhere from 20 to 200 fps depending on the user's settings and available computing power (Developer.leapmotion.com 2015). The LEAP software is open source, making the latency issue a possible future work for students.

10. Future Improvements

There will always be room for improvement on this particular thesis project as it is easy to want to strive for excellence. There are numerous areas that need improving: Gesture recognition accuracy, user friendly interface and better communication between software and hardware interfaces are a few examples.

The follow list potential future improvements that can be made for this project:

Additional Mode Settings: Most new EPW's have a joystick controller as well as additional buttons and a LCD screen. The buttons allow the user to change mode settings. The modes are changed when the wheelchair changes terrain, like going up a small hill, or changes if a different speed setting is required. The LEAP controlled wheelchair is programmed to change the PWM frequency increasingly the longer the forward hand gesture is held for. The option for having different set speed modes would potentially be more desirable for market use.

Real time feedback/ new testing platform: A GUI would be beneficial to implement for testing and debugging new code. A simple indication method for correct hand gestures or a more elaborate wheelchair driving simulator can be made for better visualization. A GUI will also provide additional information regarding the controller interface and performance to the user. Another advantage is that participants testing out new systems would be more comfortable watching their gestures control a virtual indication method on a screen. This a highly recommended future improvement.

Additional sensors: Additional proximity sensors would assist in the user friendliness and overall navigation of the wheelchair. The proximity sensors could be ultrasonic sensors that will collectively act as collision avoidance system. A GPS system can also be implemented for

autonomous navigation. This would work in coincide with the proximity sensors to ensure the user reaches the destination and that the wheelchair is protected from potential collisions.

Addition of accelerometers, encoders and gyroscopes could also be considered.

WEKA Data/Larger testing sample: Collecting and analysing more hand gesture data of different types of gestures would complement and improve the research on this paper. As all the data was collected from the same right hand from the same person. It would be beneficial to obtain data for the same hand gesture from a large number of participants, therefore WEKA would produce an algorithm that would potentially better suited as a general template to suit any person's individual interpretation and physical capability of the same gesture.

Alternative Gesture Controllers: There were notable disadvantages with the LEAP controller. The LEAP device needs a graphics card as well as a minimum of an i3 processor. Extra development would be required to accommodate the LEAP motion device if it were to be implemented and tested in a working motorised wheelchair. It is opined that the GestIC technology be used if this project were to be restarted as it has a good development system with GUI interface and would proceed to be able to evolve and integrate into a smart wheelchair system.

This project provides a basic platform for hand gesture recognition controllers. A continuation of this project would result in further tests and eventually an implementation of the controller on a motorized wheelchair. Implementation on an actual motorised wheelchair or mobility scooter would be the final finishing and testing stage of this project before making this alternative controller a commercial product.

11. Conclusion

This thesis project aimed to design an alternative controller for a wheelchair for people with limited arm and hand dexterity. An interactive hand gesture controller system was designed and tested. The LEAP motion sensor was the hand recognition device used, and although its online review forewarned of the flaws of the controller, the overall accuracy of the hand gesture recognition system was satisfactory. The hand gesture recognition testing showed that 97.16% of the hand gestures selected for this thesis were correctly identified. In comparison to other alternative wheelchair controller research articles available online, 97.16% accuracy for this gesture controller is on par, exceeding the reliability of other alternative controllers.

The controller is a non-invasive, discrete and accurate, but there are hurdles to cross when it comes to the user ergonomics and user experience. Elderly users who rarely use computers or smart phones, found the gesture controller less intuitive to use than that of other participants. Although the gesture controller is designed so that a gesture is only required to be held in the LEAP motion range of view for a short time, the hand gestures are still hard to manage. The positive side is that the controller can be customised to have any robust gesture that better suits the user.

In conclusion, this project has been a proof of the concept that an interactive hand gesture controller can be designed and programmed to be an accurate and reliable alternative controller for a motorised wheelchair. There are many improvements to be made to the system architecture of this project and this gesture recognition controller is not a commercially viable solution for an alternative wheelchair controller.

12. References

- Abarca, Edwin. 2014. "Decoding 3D Printing: Sculpteo @ Autodesk". *Leap Motion Blog*.
<http://blog.leapmotion.com/decoding-3d-printing-sculpteo-autodesk/>. (Accessed: 14/01/2016)
- Atwiki.assistivetechnet.net,. 2015. 'Alternative Wheelchair Control - Atwiki'.
http://atwiki.assistivetechnet.net/index.php/Alternative_wheelchair_control. (Accessed: 04/09/2015)
- Bishop, Christopher. 2016. "Christopher M. Bishop | PRML". *Research.Microsoft.Com*.
<http://research.microsoft.com/en-us/um/people/cmbishop/PRML/>. (Accessed: 14/01/2016)
- Brown, Paul. 2015. 'Hover Board » Raspberry Pi Geek'. *Raspberry Pi Geek*. <http://www.Raspberry-Pi-geek.com/Archive/2014/07/Using-the-Hover>. (Accessed: 11/09/2015)
- Cardinal, David. 2012. 'Gestic Brings 3D, Camera-Free Gesture Recognition To Low-Cost Devices | Extremetech'. *Extremetech*. <http://www.extremetech.com/extreme/140286-gestic-brings-3d-camera-free-gesture-recognition-to-low-cost-devices>. (Accessed: 15/09/2015)
- Colgan, Alex. 2014. 'How Does The Leap Motion Controller Work?'. *Leap Motion Blog*.
<http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/>. (Accessed: 11/10/2015)
- Commons.wikimedia.org,. 2011. 'File:Xbox-360-Kinect-Standalone.Png - Wikimedia Commons'.
<https://commons.wikimedia.org/wiki/File%3AXbox-360-Kinect-Standalone.png>. (Accessed: 13/09/2015)
- Community.pentaho.com,. 2015. 'Weka | Pentaho Community'.
<http://community.pentaho.com/projects/data-mining/>. (Accessed: 14/01/2016)
- Corp., Origin. 2015. 'Sip/Puff Switch'. *Orin.Com*. http://www.orin.com/access/sip_puff/.
(Accessed: 04/10/2015)
- Dlink.com.au,. 2016. "Wireless AC1200 Dual Band USB Adapter". <http://www.dlink.com.au/home-solutions/wireless-ac1200-dual-band-usb-adapter>. (Accessed: 14/01/2016)
- Dev.windows.com,. 2015. 'Kinect - Windows App Development'. <https://dev.windows.com/en-us/kinect>. (Accessed: 14/01/2016)

Developer.leapmotion.com,. 2016. "Listener — Leap Motion Python SDK V2.3 Documentation".
<https://developer.leapmotion.com/documentation/python/api/Leap.Listener.html>. (Accessed:
14/01/2016)

Developer.leapmotion.com. 2015. 'API Overview " Leap Motion Python SDK V2.3 Documentation'.
https://developer.leapmotion.com/documentation/python/devguide/Leap_Overview.html.
(Accessed: 06/10/2015)

Developer.leapmotion.com. 2015. 'API Reference " Leap Motion Python SDK V2.3
Documentation'.
https://developer.leapmotion.com/documentation/python/aPi/Leap_Classes.html.
(Accessed: 12/10/2015)

Developer.leapmotion.com. 2015. 'Vector " Leap Motion Python SDK V2.3 Documentation'.
<https://developer.leapmotion.com/documentation/python/aPi/Leap.Vector.html>.
(Accessed: 12/10/2015)

Developer.leapmotion.com. 2015. 'Finger " Leap Motion Python SDK V2.3 Documentation'.
<https://developer.leapmotion.com/documentation/python/aPi/Leap.Finger.html>.
(Accessed: 23/09/2015)

Developer.leapmotion.com. 2015. 'Hand " Leap Motion Python SDK V2.3 Documentation'.
<https://developer.leapmotion.com/documentation/python/aPi/Leap.Hand.html>. (Accessed:
20/10/2015)

Developer.leapmotion.com. 2015. 'Frame " Leap Motion Python SDK V2.3 Documentation'.
<https://developer.leapmotion.com/documentation/python/aPi/Leap.Frame.html>.
(Accessed: 15/09/2015)

Developer.leapmotion.com. 2015. 'Listener " Leap Motion Python SDK V2.3 Documentation'.
<https://developer.leapmotion.com/documentation/python/aPi/Leap.Listener.html>.
(Accessed: 21/09/2015)

Developer.leapmotion.com. 2016. "Skeletal Tracking | Leap Motion Developers".
<https://developer.leapmotion.com/>. (Accessed: 22/09/2015)

Dev.windows.com. 2016. "Kinect - Windows App Development". <https://dev.windows.com/en-us/kinect>. (Accessed: 14/09/2015)

Dictionary.com. 2015. 'The Definition Of Data-Mining'.

- <http://dictionary.reference.com/browse/data-mining>. (Accessed: 14/01/2016)
- Docs.oracle.com. 2015. 'What Is Data Mining?'.
http://docs.oracle.com/cd/B28359_01/datamine.111/b28129/process.htm. (Accessed: 14/01/2016)
- Docs.oracle.com. 2016. "Regression".
https://docs.oracle.com/cd/B28359_01/datamine.111/b28129/regress.htm. (Accessed: 14/01/2016)
- Doherty, Jay. 2012. "Drive Controls & Programming A Power Chair -- Mobility Management".
Mobilitymgmt.Com. <https://mobilitymgmt.com/Articles/2012/11/01/Drive-Controls.aspx>.
 (Accessed: 14/01/2016)
- Dual Full-Bridge Driver*. 2000. Ebook. 1st ed. STM Microelectronics.
<http://www.st.com/web/en/resource/technical/document/datasheet/CD00000240.pdf>.
 (Accessed: 14/01/2016)
- Duo3d.com. 2015. 'DUO - A Compact USB Camera For Sensing Space.'. <https://duo3d.com/>.
 (Accessed: 04/09/2015)
- Duo3d.com. 2015. 'DUO Comparison'. <https://duo3d.com/compare>. (Accessed: 05/09/2015)
- Engadget. 2016. "Leap Motion Controller Review". <http://www.engadget.com/2013/07/22/leap-motion-controller-review/>. (Accessed: 14/01/2016)
- Facweb.cs.depaul.edu. 2016. "Classification Via Decision Trees In WEKA".
<http://facweb.cs.depaul.edu/mobasher/classes/ect584/WEKA/classify.html>. (Accessed: 14/01/2016)
- Geekonfire.com. 2015. 'Dual H-Bridge Motor Driver - GOF_Wikidual H-Bridge Motor Driver - Open Source, Open Minded'. http://www.geekonfire.com/wiki/index.php?title=Dual_H-Bridge_Motor_Driver. (Accessed: 16/10/2015)
- Gorman, Michael. 2015. 'Leap Motion Controller Review'. *Engadget*.
<http://www.engadget.com/2013/07/22/leap-motion-controller-review/>. (Accessed: 02/10/2015)
- Hantrakul, Lamtharn. 2014. "Organically Creating Music With Your Hands And Fingers". *Leap Motion Blog*. <http://blog.leapmotion.com/organically-creating-music-with-your-hands-and-fingers/>. (Accessed: 17/10/2015)
- Hiremath, Shivayogi V., Dan Ding, and Rory A. Cooper. 2013. 'Development And Evaluation Of A

- Gyroscope-Based Wheel Rotation Monitor For Manual Wheelchair Users'. *The Journal Of Spinal Cord Medicine* 36 (4): 347-356. doi:10.1179/2045772313y.0000000113. (Accessed: 03/10/2015)
- Hover Labs Co. 2015. 'HOME'. <http://www.hoverlabs.co/#shop>. (Accessed: 19/10/2015)
- Instructables.com. 2016. "Robotic Hand Controlled By Gesture With Arduino + Leap Motion". <http://www.instructables.com/id/Robotic-Hand-controlled-by-Gesture-with-Arduino-Le/>. (Accessed: 14/01/2016)
- "Introducing The World's First E Field Based 3D Gesture Controller". 2016. <http://ww1.microchip.com/downloads/en/DeviceDoc/41660a.pdf> (Accessed: 14/01/2016)
- Intel. 2015. 'Intel Realsense Camera'. <http://www.intel.com/content/www/us/en/architecture-and-technology/realsense-depth-camera.html>. (Accessed: 04/10/2015)
- Kim-Tien, Nguyen, Nguyen Truong-Thinh, and Trinh Duc Cuong. 2013. 'A Method For Controlling Wheelchair Using Hand Gesture Recognition'. *Advances In Intelligent Systems And Computing*, 961-970. doi:10.1007/978-3-642-37374-9_93. (Accessed: 17/09/2015)
- Kouroupetroglou, Christos. *Enhancing The Human Experience Through Assistive Technologies And E-Accessibility*. (Accessed: 14/01/2016)
- Kyrnin, Mark. 2013. 'Leap Motion Controller Adds Motion Controls To Any Mac Or Windows Computer - Tech For Anyone'. *Tech For Anyone*. <http://www.techforanyone.com/leap-motion-controller-adds-motion-controls-mac-windows-computer/> (Accessed: 16/10/2015)
- L298N, Full-Bridge, SparkFun Shield, SparkFun Kit, Motor IRF7862PBF, Motor H-Bridge, SparkFun Driver, Heatsink TO-220, and SparkFun L298N. 2015. 'Full-Bridge Motor Driver Dual - L298N - COM-09479 - Sparkfun Electronics'. *Sparkfun.Com*. <https://www.sparkfun.com/products/9479>. (Accessed: 26/09/2015)
- Leap Motion Support. 2015. <https://support.leapmotion.com/home>. (Accessed: 14/01/2016)
- Learn.sparkfun.com. 2016. "Leap Motion Teardown - Learn.Sparkfun.Com". <https://learn.sparkfun.com/tutorials/leap-motion-teardown>. (Accessed: 16/09/2015)
- Levin, Mike. 2011. 'Python Programming Language Advantages And Disadvantages (My Love Letter To Python)... - Mike Levin SEO Consultant NYC'. *Mike Levin SEO Consultant NYC*.

- <http://mikelev.in/2011/01/python-programming-language-advantages/>. (Accessed: 19/10/2015)
- Lopresti EF, et al. 2016. "Performance Testing Of Collision-Avoidance System For Power Wheelchairs. - Pubmed - NCBI". *Ncbi.Nlm.Nih.Gov*.
<http://www.ncbi.nlm.nih.gov/pubmed/21674403>. (Accessed: 25/09/2015)
- Microchip.com. 2015. 'Gestic Technology - Touch And Input Sensing | Microchip Technology Inc.'. http://www.microchip.com/pagehandler/en_us/technology/gestic. (Accessed: 16/09/2015)
- Microchip.com. 2015. 'MGC3130 - Capacitive Touch Sensors'.
<http://www.microchip.com/wwwproducts/Devices.aspx?product=MGC3130>. (Accessed: 18/10/2015)
- Microchipdirect.com. 2015. 'Product Search'.
<http://www.microchipdirect.com/ProductSearch.aspx?Keywords=dm160226>. (Accessed: 14/09/2015)
- Motion, Leap. 2015. 'Leap Motion'. *Leapmotion.Com*. <https://www.leapmotion.com/>. (Accessed: 23/09/2015)
- Motion, Leap. 2016. "Leap Motion App Store | Apps For The Leap Motion Controller". *Apps.Leapmotion.Com*. <https://apps.leapmotion.com/categories/games>. (Accessed: 04/10/2015)
- Motion, Leap. 2016. "Leap Motion". *Leapmotion.Com*. <https://www.leapmotion.com/> (Accessed: 14/01/2016)
- Nave, R. 2015. 'Electromagnetic Spectrum'. *Hyperphysics.Phy-Astr.Gsu.Edu*.
<http://hyperphysics.phy-astr.gsu.edu/hbase/ems3.html>. (Accessed: 20/10/2015)
- Newdisability.com,. 2015. 'WHEELCHAIR: Wheelchair Accessories: Innovative And Cool Wheelchair Accessories By Rehadesign.'. <http://www.newdisability.com/wheelchairstatistics.htm>. (Accessed: 27/09/2015)
- Ni.com,. 2016. "Labview System Design Software - National Instruments".
<http://www.ni.com/labview/>. (Accessed: 14/01/2016)
- Pajkanovic, Aleksandar, and Branko Dokic. 2013. 'Wheelchair Control By Head Motion'. *Serb J Electr Eng* 10 (1): 135-151. doi:10.2298/sjee1301135p. (Accessed: 26/09/2015)

- Pande, Vishal V, Nikita S Ubale, Darshana P Masurkar, Nikita R Ingole, and Pragati P Mane. 2014. *Hand Gesture Based Wheelchair Movement Control For Disabled Person Using MEMS..* Ebook. 4th ed. Thane: Mumbai University.
http://www.ijera.com/papers/Vol4_issue4/Version%204/Y04404152158.pdf. (Accessed: 09/09/2015)
- PCMag Australia,. 2013. 'Leap Motion Controller'. <http://au.pcmag.com/leap-motion-controller/5086/review/leap-motion-controller>. (Accessed: 07/09/2015)
- Quinlan, J. Ross. 2014. *C4.5: Programs For Machine Learning*. Ebook. 1st ed. San Mateo California: Morgan Kaufmann Publishers.
https://books.google.com.au/books?id=b3ujBQAAQBAJ&printsec=frontcover&source=gbg_summary_r&cad=0#v=onepage&q&f=false. (Accessed: 05/10/2015)
- Quora.com,. 2015. 'What Is The Difference Between Data Analytics, Data Analysis, Data Mining, Data Science, Machine Learning, And Big Data? - Quora'. <https://www.quora.com/What-is-the-difference-between-Data-Analytics-Data-Analysis-Data-Mining-Data-Science-Machine-Learning-and-Big-Data-1>. (Accessed: 14/01/2016)
- Quora.com,. 2016. "What Are Some Interesting Possible Applications Of Machine Learning? - Quora". <https://www.quora.com/What-are-some-interesting-possible-applications-of-machine-learning>. (Accessed: 14/01/2016)
- Raspberry Pi,. 2015. 'Raspberry Pi 2 Model B'. <https://www.RaspberryPi.org/products/Raspberry-Pi-2-model-b/>. (Accessed: 20/08/2015)
- Raspberry Pi,. 2016. "Raspberry Pi - Teach, Learn, And Make With Raspberry Pi".
<https://www.raspberrypi.org/>. (Accessed: 29/08/2015)
- Realsense, Intel. 2013. 'INTEL REALSENSE'. *Intelrealsense.Bemyapp.Com*.
<http://intelrealsense.bemyapp.com/>. (Accessed: 23/09/2015)
- Saedsayad.com. 2016. "Decision Tree". http://www.saedsayad.com/decision_tree.htm. (Accessed: 14/01/2016)
- Sanders, Brandon, and Woodie Flowers. *Mastering Leap Motion*. (Accessed: 01/09/2015)
- Schapire, Rob. 2008. *Theoretical Machine Learning*. Ebook. 1st ed.
http://www.cs.princeton.edu/courses/archive/spr08/cos511/scribe_notes/0204.pdf. (Accessed: 14/01/2016)

- Stein, Scott. 2013. 'Leap Motion Controller Review: Waving Your Hands At The Future'. *CNET*.
<http://www.cnet.com/au/products/leap-motion-controller/>. (Accessed: 15/10/2015)
- Simpson, Richard C. 2008. *How Many People Would Benefit From A Smart Wheelchair?*. Ebook. 1st ed. Journal of Rehabilitation Research and Development.
<http://www.rehab.research.va.gov/JOUR/08/45/1/pdf/simpson.pdf>. (Accessed: 14/01/2016)
- Software.intel.com,. 2016. "Overview Of Intel® Realsense™ SDK | Intel® Developer Zone".
<https://software.intel.com/en-us/intel-realsense-sdk>. (Accessed: 11/09/2015)
- Support.xbox.com,. "Kinect Components | Xbox 360". N.p., 2016. Web. (Accessed: 14/01/2016)
- Upfront Analytics,. 2015. 'Data Mining Vs Artificial Intelligence Vs Machine Learning - Upfront Analytics'. <http://upfrontanalytics.com/data-mining-vs-artificial-intelligence-vs-machine-learning/>. (Accessed: 17/10/2015)
- Visnjic, Filip. 2016. "10 Most Exciting New Experiments With Leap Motion".
Creativeapplications.Net. <http://www.creativeapplications.net/processing/10-most-exciting-new-experiments-with-leap-motion/>. (Accessed: 29/09/2015)
- WhatIs.com,. 2015. 'What Is Machine Learning? - Definition From WhatIs.Com'.
<http://whatis.techtarget.com/definition/machine-learning>. (Accessed: 14/09/2015)
- Wikipedia,. 2015. 'Motorized Wheelchair'.
https://en.wikipedia.org/wiki/Motorized_wheelchair#/media/File:Electric-powered_wheelchair_Belize2.jpg. (Accessed: 14/09/2015)
- www.tutorialspoint.com,. 2016. "Data Mining Cluster Analysis". Learn Data Mining. Data Pattern Evaluation. http://www.tutorialspoint.com/data_mining/dm_cluster_analysis.htm. (Accessed: 14/01/2016)

Appendices

Appendix A

LEAP Data Sampling Application

The python code for the data sampling application:

```
import sys
import Leap, thread, time
from Leap import CircleGesture, KeyTapGesture, ScreenTapGesture,
SwipeGesture

class SampleListener(Leap.Listener):
    finger_names = ['Thumb', 'Index', 'Middle', 'Ring', 'Pinky']
    bone_names = ['Metacarpal', 'Proximal', 'Intermediate', 'Distal']

    def on_init(self, controller):
        print "Initialized"

    def on_connect(self, controller):
        print "Connected"

    def on_disconnect(self, controller):
        print "Disconnected"

    def on_exit(self, controller):
        print "Exited"

    # Code that I wrote to collect the data
    def on_frame(self, controller):
        # Get the most recent frame and report some basic information
        frame = controller.frame()

        # Get hands
        for hand in frame.hands:
            #handType = "Left hand" if hand.is_left else "Right hand"
            #print "  %s, id %d, position: %s" % (handType, hand.id,
            hand.palm_position)

            # Get the hand's normal vector and direction
            normal = hand.palm_normal
            direction = hand.direction
            # Get arm bone
            arm = hand.arm

            # Calculate the hand's Pitch, roll, and yaw angles
            if len(hand.fingers) == 5:
```

```

        # If the sensor sees all 5 fingers then we are sure that
        the whole hand is there, and we can collect the sample. The first part of
        the sample consist of the palm normal (go check leap sensor page ref
        here), then hand direction, and arm position.
        print round(direction.Pitch * Leap.RAD_TO_DEG,2),
        round(normal.roll * Leap.RAD_TO_DEG,2), round(direction.yaw *
        Leap.RAD_TO_DEG,2),
        print round(arm.direction[0],2),
        round(arm.direction[1],2), round(arm.direction[2],2), arm.wrist_position,
        arm.elbow_position,
        # Get fingers
        mystr = ""
        for finger in hand.fingers:
            for b in range(0, 4):
                bone = finger.bone(b)
                # For each finger, for each bone get the bone
                direction (5 fingers, 4 bones, 3 coordinates per direction = 60 values)
                mystr += str(round(bone.direction[0],2) ) + " "
            +str(round(bone.direction[1],2) ) + " " +str(round(bone.direction[2],2)
            )+ " "

        print mystr

def main():
    # Create a sample listener and controller
    listener = SampleListener()
    controller = Leap.Controller()

    # Have the sample listener receive events from the controller
    controller.add_listener(listener)

    # Keep this process running until Enter is pressed
    print "Press Enter to quit..."
    try:
        sys.stdin.readline()
    except KeyboardInterrupt:
        pass
    finally:
        # Remove the sample listener when done
        controller.remove_listener(listener)

if __name__ == "__main__":
    main()

```

Appendix B

WEKA Data File

An example of the data that is being collected from the LEAP sensor using the data sampling application is provided below. First attributes are the Pitch, roll, and yaw of the arm, then the arm direction, the wrist and the elbow position and finally the fingers. Each finger is described by 4 bones and each bone by its angular direction. The last attribute is the event - the position that our arm was in that produced this data values. This data is then fed to a machine learning algorithm that will try to predict the event from the data values. The resulting model is transformed into code that predicts the event in real time using values that come from the LEAP sensor.

```
@RELATION leap
```

```
@ATTRIBUTE Pitch NUMERIC
@ATTRIBUTE roll NUMERIC
@ATTRIBUTE yaw NUMERIC
@ATTRIBUTE arm_direction0 NUMERIC
@ATTRIBUTE arm_direction1 NUMERIC
@ATTRIBUTE arm_direction2 NUMERIC
@ATTRIBUTE arm_wrist_position0 NUMERIC
@ATTRIBUTE arm_wrist_position1 NUMERIC
@ATTRIBUTE arm_wrist_position2 NUMERIC
@ATTRIBUTE arm_elbow_position0 NUMERIC
@ATTRIBUTE arm_elbow_position1 NUMERIC
@ATTRIBUTE arm_elbow_position2 NUMERIC
# finger id, bone id, three coordinate point
@ATTRIBUTE bone_direction_0_0_0 NUMERIC
@ATTRIBUTE bone_direction_0_0_1 NUMERIC
@ATTRIBUTE bone_direction_0_0_2 NUMERIC
@ATTRIBUTE bone_direction_0_1_0 NUMERIC
@ATTRIBUTE bone_direction_0_1_1 NUMERIC
@ATTRIBUTE bone_direction_0_1_2 NUMERIC
@ATTRIBUTE bone_direction_0_2_0 NUMERIC
@ATTRIBUTE bone_direction_0_2_1 NUMERIC
@ATTRIBUTE bone_direction_0_2_2 NUMERIC
@ATTRIBUTE bone_direction_0_3_0 NUMERIC
@ATTRIBUTE bone_direction_0_3_1 NUMERIC
@ATTRIBUTE bone_direction_0_3_2 NUMERIC
@ATTRIBUTE bone_direction_1_0_0 NUMERIC
@ATTRIBUTE bone_direction_1_0_1 NUMERIC
@ATTRIBUTE bone_direction_1_0_2 NUMERIC
@ATTRIBUTE bone_direction_1_1_0 NUMERIC
@ATTRIBUTE bone_direction_1_1_1 NUMERIC
@ATTRIBUTE bone_direction_1_1_2 NUMERIC
@ATTRIBUTE bone_direction_1_2_0 NUMERIC
@ATTRIBUTE bone_direction_1_2_1 NUMERIC
@ATTRIBUTE bone_direction_1_2_2 NUMERIC
```


@ATTRIBUTE bone_direction_1_3_0 NUMERIC
 @ATTRIBUTE bone_direction_1_3_1 NUMERIC
 @ATTRIBUTE bone_direction_1_3_2 NUMERIC
 @ATTRIBUTE bone_direction_2_0_0 NUMERIC
 @ATTRIBUTE bone_direction_2_0_1 NUMERIC
 @ATTRIBUTE bone_direction_2_0_2 NUMERIC
 @ATTRIBUTE bone_direction_2_1_0 NUMERIC
 @ATTRIBUTE bone_direction_2_1_1 NUMERIC
 @ATTRIBUTE bone_direction_2_1_2 NUMERIC
 @ATTRIBUTE bone_direction_2_2_0 NUMERIC
 @ATTRIBUTE bone_direction_2_2_1 NUMERIC
 @ATTRIBUTE bone_direction_2_2_2 NUMERIC
 @ATTRIBUTE bone_direction_2_3_0 NUMERIC
 @ATTRIBUTE bone_direction_2_3_1 NUMERIC
 @ATTRIBUTE bone_direction_2_3_2 NUMERIC
 @ATTRIBUTE bone_direction_3_0_0 NUMERIC
 @ATTRIBUTE bone_direction_3_0_1 NUMERIC
 @ATTRIBUTE bone_direction_3_0_2 NUMERIC
 @ATTRIBUTE bone_direction_3_1_0 NUMERIC
 @ATTRIBUTE bone_direction_3_1_1 NUMERIC
 @ATTRIBUTE bone_direction_3_1_2 NUMERIC
 @ATTRIBUTE bone_direction_3_2_0 NUMERIC
 @ATTRIBUTE bone_direction_3_2_1 NUMERIC
 @ATTRIBUTE bone_direction_3_2_2 NUMERIC
 @ATTRIBUTE bone_direction_3_3_0 NUMERIC
 @ATTRIBUTE bone_direction_3_3_1 NUMERIC
 @ATTRIBUTE bone_direction_3_3_2 NUMERIC
 @ATTRIBUTE bone_direction_4_0_0 NUMERIC
 @ATTRIBUTE bone_direction_4_0_1 NUMERIC
 @ATTRIBUTE bone_direction_4_0_2 NUMERIC
 @ATTRIBUTE bone_direction_4_1_0 NUMERIC
 @ATTRIBUTE bone_direction_4_1_1 NUMERIC
 @ATTRIBUTE bone_direction_4_1_2 NUMERIC
 @ATTRIBUTE bone_direction_4_2_0 NUMERIC
 @ATTRIBUTE bone_direction_4_2_1 NUMERIC
 @ATTRIBUTE bone_direction_4_2_2 NUMERIC
 @ATTRIBUTE bone_direction_4_3_0 NUMERIC
 @ATTRIBUTE bone_direction_4_3_1 NUMERIC
 @ATTRIBUTE bone_direction_4_3_2 NUMERIC

@ATTRIBUTE event

{open_arm,full_fist,index_and_thumb,index_and_middle,index_and_thumb_and_middle,thumb_only,axe,thumb_and_Pi
nky}

@data

-0.0,-1.47,6.0,0.15,-0.47,-0.87,-10.9301,121.59,97.0169,-46.3337,230.104,295.776,0.0,0.0,0.0,0.56,-
 0.02,0.83,0.45,0.05,0.89,0.65,-0.08,0.75,0.07,0.14,0.99,0.19,0.14,0.97,0.19,0.2,0.96,0.19,0.25,0.95,-0.07,0.15,0.99,-
 0.04,0.1,0.99,-0.03,0.18,0.98,-0.02,0.24,0.97,-0.22,0.15,0.96,-0.17,0.05,0.99,-0.12,0.22,0.97,-0.08,0.36,0.93,-
 0.36,0.11,0.93,-0.54,0.09,0.84,-0.48,0.23,0.85,-0.43,0.35,0.84,open_arm
 -0.01,-6.5,3.16,0.07,-0.5,-0.86,-14.4209,117.234,90.3736,-31.2888,231.197,288.52,0.0,0.0,0.0,0.63,-
 0.03,0.78,0.58,0.01,0.81,0.72,-0.11,0.69,0.13,0.13,0.98,0.26,0.09,0.96,0.26,0.16,0.95,0.27,0.22,0.94,-0.01,0.14,0.99,-
 0.02,0.07,1.0,-0.0,0.14,0.99,0.02,0.2,0.98,-0.16,0.16,0.97,-0.15,0.04,0.99,-0.12,0.11,0.99,-0.11,0.16,0.98,-0.3,0.13,0.94,-
 0.5,0.07,0.86,-0.49,0.11,0.87,-0.47,0.15,0.87,open_arm
 -0.01,-72.86,1.72,-0.43,0.48,-0.76,-9.47284,175.887,40.9209,83.311,74.124,204.328,0.0,0.0,0.0,0.15,-0.77,0.63,0.19,-
 0.67,0.72,0.28,-0.23,0.93,0.16,-0.11,0.98,0.24,-0.21,0.95,0.34,-0.18,0.92,0.42,-0.15,0.9,0.12,0.02,0.99,0.21,-
 0.06,0.98,0.34,-0.04,0.94,0.43,-0.02,0.9,0.08,0.16,0.98,0.18,0.06,0.98,0.31,0.07,0.95,0.4,0.08,0.91,-
 0.0,0.28,0.96,0.09,0.06,0.99,0.23,0.05,0.97,0.33,0.04,0.94,open_arm

-0.02,-1.48,6.02,0.15,-0.47,-0.87,-10.9451,121.609,97.0155,-46.3382,230.188,295.741,0.0,0.0,0.0,0.56,-
 0.02,0.83,0.45,0.05,0.89,0.65,-0.08,0.75,0.07,0.14,0.99,0.19,0.14,0.97,0.19,0.2,0.96,0.19,0.25,0.95,-0.07,0.15,0.99,-
 0.04,0.1,0.99,-0.03,0.18,0.98,-0.02,0.24,0.97,-0.22,0.15,0.96,-0.17,0.05,0.99,-0.12,0.22,0.97,-0.08,0.36,0.93,-
 0.36,0.11,0.93,-0.54,0.09,0.84,-0.48,0.23,0.85,-0.42,0.35,0.84,open_arm
 -0.02,-72.52,1.72,-0.44,0.48,-0.76,-9.51658,175.901,41.0012,83.5612,74.1046,204.22,0.0,0.0,0.0,0.16,-0.76,0.63,0.19,-
 0.68,0.71,0.28,-0.3,0.91,0.16,-0.11,0.98,0.24,-0.21,0.95,0.34,-0.17,0.92,0.42,-0.14,0.9,0.12,0.02,0.99,0.21,-
 0.05,0.98,0.34,-0.03,0.94,0.43,-0.02,0.9,0.08,0.16,0.98,0.18,0.06,0.98,0.31,0.07,0.95,0.4,0.08,0.91,-
 0.01,0.28,0.96,0.1,0.07,0.99,0.23,0.06,0.97,0.34,0.05,0.94,open_arm
 -0.03,-1.48,6.04,0.15,-0.47,-0.87,-10.9597,121.628,97.014,-46.3423,230.271,295.706,0.0,0.0,0.0,0.56,-
 0.02,0.83,0.45,0.05,0.89,0.66,-0.08,0.75,0.07,0.14,0.99,0.19,0.14,0.97,0.19,0.2,0.96,0.19,0.25,0.95,-0.07,0.15,0.99,-
 0.04,0.1,0.99,-0.03,0.18,0.98,-0.02,0.24,0.97,-0.22,0.15,0.96,-0.17,0.05,0.99,-0.12,0.22,0.97,-0.08,0.36,0.93,-
 0.36,0.11,0.93,-0.54,0.09,0.84,-0.48,0.23,0.85,-0.42,0.35,0.84,open_arm
 -0.03,-6.53,3.16,0.07,-0.5,-0.86,-14.4228,117.241,90.3678,-31.2952,231.202,288.515,0.0,0.0,0.0,0.63,-
 0.03,0.78,0.58,0.01,0.81,0.72,-0.11,0.69,0.13,0.13,0.98,0.26,0.09,0.96,0.26,0.16,0.95,0.27,0.22,0.94,-0.01,0.14,0.99,-
 0.02,0.07,1.0,-0.0,0.14,0.99,0.02,0.2,0.98,-0.16,0.16,0.97,-0.15,0.04,0.99,-0.13,0.1,0.99,-0.11,0.16,0.98,-0.3,0.14,0.94,-
 0.5,0.07,0.86,-0.49,0.11,0.87,-0.47,0.15,0.87,open_arm
 -0.04,-1.49,6.06,0.15,-0.47,-0.87,-10.9742,121.647,97.012,-46.3455,230.355,295.671,0.0,0.0,0.0,0.56,-
 0.02,0.83,0.45,0.06,0.89,0.66,-0.08,0.75,0.07,0.14,0.99,0.19,0.14,0.97,0.19,0.2,0.96,0.19,0.25,0.95,-0.07,0.15,0.99,-
 0.04,0.11,0.99,-0.03,0.18,0.98,-0.02,0.24,0.97,-0.22,0.15,0.96,-0.17,0.05,0.99,-0.12,0.22,0.97,-0.08,0.36,0.93,-
 0.36,0.11,0.93,-0.54,0.09,0.84,-0.48,0.23,0.85,-0.42,0.35,0.84,open_arm
 -0.04,-72.83,1.71,-0.43,0.48,-0.76,-9.46689,175.888,40.9251,83.3139,74.0771,204.304,0.0,0.0,0.0,0.16,-0.77,0.62,0.19,-
 0.68,0.71,0.28,-0.26,0.92,0.16,-0.11,0.98,0.24,-0.21,0.95,0.34,-0.17,0.92,0.42,-0.14,0.9,0.12,0.02,0.99,0.22,-
 0.06,0.97,0.34,-0.04,0.94,0.43,-0.02,0.9,0.08,0.16,0.98,0.18,0.06,0.98,0.31,0.07,0.95,0.4,0.08,0.91,-
 0.0,0.28,0.96,0.09,0.06,0.99,0.23,0.05,0.97,0.33,0.04,0.94,open_arm
 -0.05,-6.35,3.22,0.07,-0.5,-0.87,-14.539,117.38,90.3421,-31.1154,231.067,288.671,0.0,0.0,0.0,0.62,-
 0.03,0.78,0.58,0.01,0.81,0.72,-0.12,0.68,0.13,0.13,0.98,0.26,0.1,0.96,0.26,0.17,0.95,0.27,0.22,0.94,-0.01,0.14,0.99,-
 0.02,0.07,1.0,-0.0,0.15,0.99,0.02,0.21,0.98,-0.16,0.16,0.97,-0.15,0.05,0.99,-0.13,0.12,0.99,-0.11,0.17,0.98,-
 0.3,0.13,0.94,-0.51,0.07,0.86,-0.49,0.11,0.87,-0.47,0.14,0.87,open_arm
 -0.05,-6.55,3.16,0.07,-0.5,-0.86,-14.4249,117.247,90.362,-31.3017,231.206,288.51,0.0,0.0,0.0,0.63,-
 0.03,0.78,0.58,0.01,0.81,0.72,-0.11,0.69,0.13,0.13,0.98,0.26,0.09,0.96,0.26,0.16,0.95,0.27,0.22,0.94,-0.01,0.14,0.99,-
 0.02,0.06,1.0,-0.0,0.14,0.99,0.02,0.2,0.98,-0.16,0.16,0.97,-0.15,0.04,0.99,-0.13,0.1,0.99,-0.11,0.16,0.98,-0.3,0.14,0.94,-
 0.5,0.07,0.86,-0.49,0.11,0.87,-0.47,0.15,0.87,open_arm
 -0.05,-72.76,1.7,-0.43,0.48,-0.76,-9.46533,175.891,40.9371,83.3624,74.0433,204.266,0.0,0.0,0.0,0.16,-0.77,0.62,0.19,-
 0.68,0.71,0.28,-0.28,0.92,0.16,-0.11,0.98,0.24,-0.21,0.95,0.34,-0.17,0.92,0.42,-0.14,0.9,0.12,0.02,0.99,0.22,-
 0.06,0.97,0.34,-0.04,0.94,0.43,-0.02,0.9,0.08,0.16,0.98,0.18,0.06,0.98,0.31,0.07,0.95,0.4,0.08,0.91,-
 0.0,0.28,0.96,0.1,0.06,0.99,0.23,0.05,0.97,0.33,0.05,0.94,open_arm
 -0.06,-1.5,6.09,0.15,-0.47,-0.87,-10.9881,121.665,97.0105,-46.3476,230.432,295.639,0.0,0.0,0.0,0.56,-
 0.02,0.83,0.45,0.06,0.89,0.66,-0.08,0.75,0.07,0.14,0.99,0.19,0.14,0.97,0.19,0.2,0.96,0.19,0.25,0.95,-0.07,0.15,0.99,-
 0.04,0.11,0.99,-0.03,0.18,0.98,-0.02,0.24,0.97,-0.22,0.15,0.96,-0.17,0.05,0.99,-0.12,0.22,0.97,-0.08,0.37,0.93,-
 0.36,0.11,0.93,-0.54,0.09,0.84,-0.48,0.23,0.85,-0.42,0.35,0.84,open_arm
 -0.07,-1.52,6.11,0.15,-0.47,-0.87,-11.003,121.669,97.0058,-46.3454,230.493,295.606,0.0,0.0,0.0,0.56,-
 0.02,0.83,0.45,0.06,0.89,0.65,-0.08,0.75,0.07,0.15,0.99,0.19,0.14,0.97,0.19,0.2,0.96,0.19,0.25,0.95,-0.07,0.15,0.99,-
 0.04,0.11,0.99,-0.03,0.18,0.98,-0.02,0.24,0.97,-0.22,0.15,0.96,-0.17,0.05,0.99,-0.12,0.23,0.97,-0.08,0.37,0.93,-
 0.36,0.11,0.93,-0.54,0.09,0.84,-0.48,0.23,0.85,-0.42,0.35,0.84,open_arm
 -0.07,-6.59,3.15,0.07,-0.5,-0.86,-14.427,117.253,90.3553,-31.3085,231.211,288.503,0.0,0.0,0.0,0.63,-
 0.03,0.78,0.58,0.01,0.81,0.72,-0.11,0.69,0.13,0.13,0.98,0.26,0.09,0.96,0.26,0.16,0.95,0.27,0.22,0.94,-0.01,0.15,0.99,-
 0.02,0.06,1.0,-0.0,0.14,0.99,0.02,0.2,0.98,-0.16,0.16,0.97,-0.15,0.04,0.99,-0.13,0.1,0.99,-0.11,0.16,0.98,-0.3,0.14,0.94,-
 0.5,0.07,0.86,-0.49,0.11,0.87,-0.47,0.15,0.87,open_arm
 -0.07,-72.53,1.71,-0.44,0.48,-0.76,-9.51279,175.901,41.0241,83.7155,74.2601,204.254,0.0,0.0,0.0,0.12,-0.75,0.65,0.14,-
 0.67,0.73,0.23,-0.27,0.93,0.16,-0.11,0.98,0.24,-0.2,0.95,0.34,-0.17,0.92,0.42,-0.14,0.9,0.12,0.02,0.99,0.22,-
 0.05,0.98,0.34,-0.03,0.94,0.43,-0.02,0.9,0.08,0.16,0.98,0.18,0.07,0.98,0.31,0.08,0.95,0.4,0.08,0.91,-
 0.01,0.28,0.96,0.1,0.07,0.99,0.23,0.06,0.97,0.34,0.05,0.94,open_arm
 -0.08,-1.53,6.12,0.15,-0.47,-0.87,-11.0146,121.678,97.0018,-46.3417,230.542,295.583,0.0,0.0,0.0,0.56,-
 0.02,0.83,0.45,0.06,0.89,0.65,-0.08,0.75,0.07,0.15,0.99,0.19,0.14,0.97,0.19,0.2,0.96,0.19,0.25,0.95,-0.07,0.15,0.99,-
 0.04,0.11,0.99,-0.03,0.18,0.98,-0.02,0.24,0.97,-0.22,0.15,0.96,-0.17,0.05,0.99,-0.12,0.23,0.97,-0.08,0.37,0.93,-
 0.36,0.11,0.93,-0.54,0.09,0.84,-0.48,0.23,0.85,-0.42,0.35,0.84,open_arm
 -0.09,-1.54,6.14,0.15,-0.48,-0.87,-11.0279,121.691,96.9972,-46.3362,230.6295,557,0.0,0.0,0.0,0.56,-
 0.02,0.83,0.45,0.06,0.89,0.65,-0.08,0.76,0.06,0.15,0.99,0.19,0.14,0.97,0.19,0.2,0.96,0.19,0.25,0.95,-0.07,0.15,0.99,-

0.04,0.11,0.99,-0.03,0.18,0.98,-0.02,0.24,0.97,-0.22,0.15,0.96,-0.17,0.05,0.99,-0.12,0.23,0.97,-0.08,0.37,0.93,-
0.36,0.11,0.93,-0.54,0.09,0.84,-0.48,0.23,0.85,-0.42,0.35,0.84,open_arm
-0.1,-6.62,3.15,0.07,-0.5,-0.86,-14.4275,117.257,90.3491,-31.318,231.216,288.496,0.0,0.0,0.0,0.63,-
0.03,0.78,0.58,0.01,0.81,0.72,-0.11,0.69,0.13,0.13,0.98,0.26,0.09,0.96,0.26,0.16,0.95,0.27,0.22,0.94,-0.01,0.15,0.99,-
0.02,0.06,1.0,-0.0,0.14,0.99,0.02,0.2,0.98,-0.16,0.16,0.97,-0.15,0.04,0.99,-0.13,0.1,0.99,-0.11,0.16,0.98,-0.3,0.14,0.94,-
0.5,0.07,0.86,-0.48,0.11,0.87,-0.47,0.15,0.87,open_arm

Learned Predictive Model

This is the model generated by J48 classification algorithm. The algorithm is implemented in weka.

Actual WEKA output from the J48 classification algorithm:

=== Run information ===

Scheme:weka.classifiers.trees.J48 -C 0.25 -M 150
Relation: leap
Instances: 112388
Attributes: 73

Test mode:10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree

```
bone_direction_1_3_2 <= 0.49
| bone_direction_2_3_2 <= 0.72
| | bone_direction_4_2_2 <= 0.54
| | | bone_direction_0_3_1 <= 0
| | | | arm_elbow_position0 <= 48.0034
| | | | | bone_direction_2_3_0 <= 0.75
| | | | | bone_direction_4_2_2 <= 0.09
| | | | | bone_direction_0_1_1 <= 0.18
| | | | | | bone_direction_2_3_2 <= -0.52
| | | | | | bone_direction_2_1_0 <= -0.35: full_fist (248.0/74.0) #
| | | | | | bone_direction_2_1_0 > -0.35
| | | | | | | roll <= -56.78: full_fist (157.0/35.0)
| | | | | | | roll > -56.78
| | | | | | | | arm_elbow_position1 <= 152.298
| | | | | | | | | bone_direction_0_2_1 <= -0.13: thumb_only (228.0/8.0)
| | | | | | | | | bone_direction_0_2_1 > -0.13: full_fist (314.0/4.0)
| | | | | | | | | arm_elbow_position1 > 152.298: thumb_only (7589.0/353.0)
| | | | | | | | | bone_direction_2_3_2 > -0.52: full_fist (384.0/55.0)
| | | | | | | | | bone_direction_0_1_1 > 0.18: full_fist (930.0/52.0)
| | | | | | | | | bone_direction_4_2_2 > 0.09
| | | | | | | | | bone_direction_1_2_2 <= -0.23: thumb_and_Pinky (421.0/4.0)
| | | | | | | | | bone_direction_1_2_2 > -0.23: full_fist (257.0/59.0)
| | | | | | | | | bone_direction_2_3_0 > 0.75
| | | | | | | | | bone_direction_4_1_0 <= -0.51: thumb_and_Pinky (257.0)
| | | | | | | | | bone_direction_4_1_0 > -0.51: index_and_thumb (151.0/29.0)
| | | | | | | | | arm_elbow_position0 > 48.0034
| | | | | | | | | bone_direction_2_2_1 <= 0.5
| | | | | | | | | bone_direction_0_2_1 <= -0.4: open_arm (157.0/65.0)
| | | | | | | | | bone_direction_0_2_1 > -0.4: thumb_only (997.0/44.0)
| | | | | | | | | bone_direction_2_2_1 > 0.5
| | | | | | | | | bone_direction_0_2_1 <= -0.18
| | | | | | | | | arm_wrist_position2 <= 80.48: open_arm (2512.0/36.0)
| | | | | | | | | arm_wrist_position2 > 80.48: thumb_only (162.0/5.0)
| | | | | | | | | bone_direction_0_2_1 > -0.18: full_fist (231.0/62.0)
| | | | | | | | | bone_direction_0_3_1 > 0
| | | | | | | | | bone_direction_0_2_0 <= -0.65: thumb_only (1201.0/45.0)
```

```

| | | | bone_direction_0_2_0 > -0.65
| | | | | bone_direction_2_2_2 <= 0.7
| | | | | | bone_direction_3_1_0 <= -0.46
| | | | | | | bone_direction_2_2_0 <= -0.29: full_fist (401.0/3.0)
| | | | | | | bone_direction_2_2_0 > -0.29: thumb_and_Pinky (273.0/15.0)
| | | | | | | bone_direction_3_1_0 > -0.46
| | | | | | | bone_direction_1_1_0 <= -0.47: thumb_and_Pinky (147.0/79.0)
| | | | | | | bone_direction_1_1_0 > -0.47
| | | | | | | | bone_direction_0_2_0 <= 0.77
| | | | | | | | | arm_wrist_position0 <= 127.004: full_fist (18139.0/342.0)
| | | | | | | | | arm_wrist_position0 > 127.004: thumb_only (55.0/12.0)
| | | | | | | | | bone_direction_0_2_0 > 0.77
| | | | | | | | | bone_direction_4_0_0 <= 0.46: thumb_only (565.0/5.0)
| | | | | | | | | bone_direction_4_0_0 > 0.46
| | | | | | | | | | bone_direction_0_1_2 <= 0.08: thumb_only (209.0/48.0)
| | | | | | | | | | bone_direction_0_1_2 > 0.08: full_fist (1087.0/5.0)
| | | | | | | | | | bone_direction_2_2_2 > 0.7: index_and_thumb (206.0/26.0)
| | | | | | | | | | bone_direction_4_2_2 > 0.54
| | | | | | | | | | | bone_direction_3_2_2 <= 0.06
| | | | | | | | | | | | arm_elbow_position0 <= 73.5734: thumb_and_Pinky (10595.0/153.0)
| | | | | | | | | | | | arm_elbow_position0 > 73.5734: open_arm (150.0/26.0)
| | | | | | | | | | | | bone_direction_3_2_2 > 0.06
| | | | | | | | | | | | | bone_direction_3_3_1 <= -0.04: index_and_thumb (244.0/115.0)
| | | | | | | | | | | | | bone_direction_3_3_1 > -0.04
| | | | | | | | | | | | | | bone_direction_2_2_1 <= 0.36: open_arm (560.0/4.0)
| | | | | | | | | | | | | | bone_direction_2_2_1 > 0.36
| | | | | | | | | | | | | | | bone_direction_4_1_1 <= -0.04: open_arm (292.0/120.0)
| | | | | | | | | | | | | | | bone_direction_4_1_1 > -0.04
| | | | | | | | | | | | | | | | bone_direction_0_1_1 <= -0.24: open_arm (228.0/107.0)
| | | | | | | | | | | | | | | | bone_direction_0_1_1 > -0.24: full_fist (1823.0/103.0)
| | | | | | | | | | | | | | | | bone_direction_2_3_2 > 0.72
| | | | | | | | | | | | | | | | | bone_direction_3_2_2 <= 0.86: index_and_thumb (2174.0/20.0)
| | | | | | | | | | | | | | | | | bone_direction_3_2_2 > 0.86
| | | | | | | | | | | | | | | | | | arm_direction2 <= -0.93: index_and_middle (691.0)
| | | | | | | | | | | | | | | | | | arm_direction2 > -0.93: open_arm (202.0/1.0)
| | | | | | | | | | | | | | | | | | bone_direction_1_3_2 > 0.49
| | | | | | | | | | | | | | | | | | | bone_direction_2_2_2 <= 0.31
| | | | | | | | | | | | | | | | | | | | bone_direction_1_0_1 <= -0.4: full_fist (73.0)
| | | | | | | | | | | | | | | | | | | | bone_direction_1_0_1 > -0.4
| | | | | | | | | | | | | | | | | | | | | bone_direction_4_1_1 <= -0.24: open_arm (47.0)
| | | | | | | | | | | | | | | | | | | | | bone_direction_4_1_1 > -0.24: index_and_thumb (14371.0/25.0)
| | | | | | | | | | | | | | | | | | | | | bone_direction_2_2_2 > 0.31
| | | | | | | | | | | | | | | | | | | | | | bone_direction_3_3_2 <= 0.26
| | | | | | | | | | | | | | | | | | | | | | | bone_direction_3_3_0 <= -0.68: open_arm (274.0/106.0)
| | | | | | | | | | | | | | | | | | | | | | | bone_direction_3_3_0 > -0.68
| | | | | | | | | | | | | | | | | | | | | | | | bone_direction_4_2_2 <= 0.44
| | | | | | | | | | | | | | | | | | | | | | | | | bone_direction_2_3_2 <= 0.44: index_and_thumb (347.0/52.0)
| | | | | | | | | | | | | | | | | | | | | | | | | bone_direction_2_3_2 > 0.44
| | | | | | | | | | | | | | | | | | | | | | | | | | bone_direction_4_1_2 <= 0.8
| | | | | | | | | | | | | | | | | | | | | | | | | | | arm_elbow_position1 <= 82.7269
| | | | | | | | | | | | | | | | | | | | | | | | | | | | arm_direction2 <= -0.89: thumb_only (634.0)
| | | | | | | | | | | | | | | | | | | | | | | | | | | | arm_direction2 > -0.89: index_and_middle (171.0/70.0)
| | | | | | | | | | | | | | | | | | | | | | | | | | | | arm_elbow_position1 > 82.7269
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | bone_direction_0_1_0 <= 0.68
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | bone_direction_0_1_1 <= -0.09
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | bone_direction_2_3_0 <= -0.37: thumb_only (460.0/4.0)
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | bone_direction_2_3_0 > -0.37: index_and_middle (1012.0/7.0)
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | bone_direction_0_1_1 > -0.09

```

```

| | | | | | | | | bone_direction_1_2_0 <= -0.59: index_and_thumb (50.0/3.0)
| | | | | | | | | bone_direction_1_2_0 > -0.59: index_and_middle (16855.0/22.0)
| | | | | | | | | bone_direction_0_1_0 > 0.68
| | | | | | | | | bone_direction_1_1_2 <= 0.82: thumb_only (259.0)
| | | | | | | | | bone_direction_1_1_2 > 0.82
| | | | | | | | | bone_direction_2_2_2 <= 0.98: index_and_thumb (319.0)
| | | | | | | | | bone_direction_2_2_2 > 0.98: index_and_middle (205.0/1.0)
| | | | | | | | | bone_direction_4_1_2 > 0.8: index_and_thumb (452.0/91.0)
| | | | | | | | | bone_direction_4_2_2 > 0.44
| | | | | | | | | bone_direction_0_2_0 <= 0: thumb_only (160.0/73.0)
| | | | | | | | | bone_direction_0_2_0 > 0: index_and_thumb (716.0/18.0)
| | | | | | | | | bone_direction_3_3_2 > 0.26
| | | | | | | | | bone_direction_4_1_1 <= 0.64
| | | | | | | | | bone_direction_4_0_0 <= 0.53
| | | | | | | | | bone_direction_3_2_1 <= 0.5
| | | | | | | | | bone_direction_0_1_0 <= -0.52
| | | | | | | | | bone_direction_0_3_2 <= 0.64: thumb_only (204.0)
| | | | | | | | | bone_direction_0_3_2 > 0.64: open_arm (183.0)
| | | | | | | | | bone_direction_0_1_0 > -0.52
| | | | | | | | | bone_direction_4_2_0 <= 0.37: open_arm (18468.0/58.0)
| | | | | | | | | bone_direction_4_2_0 > 0.37
| | | | | | | | | bone_direction_1_0_0 <= -0.1: thumb_and_Pinky (159.0/1.0)
| | | | | | | | | bone_direction_1_0_0 > -0.1: open_arm (373.0/14.0)
| | | | | | | | | bone_direction_3_2_1 > 0.5
| | | | | | | | | bone_direction_2_3_1 <= 0.31: index_and_middle (151.0/10.0)
| | | | | | | | | bone_direction_2_3_1 > 0.31
| | | | | | | | | bone_direction_1_0_1 <= 0.07: full_fist (191.0)
| | | | | | | | | bone_direction_1_0_1 > 0.07: open_arm (182.0/5.0)
| | | | | | | | | bone_direction_4_0_0 > 0.53
| | | | | | | | | bone_direction_1_2_2 <= 0.92: full_fist (215.0/63.0)
| | | | | | | | | bone_direction_1_2_2 > 0.92
| | | | | | | | | bone_direction_3_1_1 <= 0.07: index_and_middle (359.0)
| | | | | | | | | bone_direction_3_1_1 > 0.07: open_arm (201.0/87.0)
| | | | | | | | | bone_direction_4_1_1 > 0.64
| | | | | | | | | arm_elbow_position2 <= 298.214: index_and_middle (910.0)
| | | | | | | | | arm_elbow_position2 > 298.214: thumb_only (150.0/66.0)

```

Number of Leaves : 66

Size of the tree : 131

Time taken to build model: 26.05 seconds

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	109198	97.1616 % #with cross validation
Incorrectly Classified Instances	3190	2.8384 %
Kappa statistic	0.9655	
Mean absolute error	0.0113	
Root mean squared error	0.0758	
Relative absolute error	5.4846 %	
Root relative squared error	23.6409 %	
Total Number of Instances	112388	#number of data points

=== Detailed Accuracy By Class === #the accuracy for each class

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.984	0.007	0.976	0.984	0.98	0.998	open_arm
	0.973	0.012	0.957	0.973	0.965	0.996	full_fist
	0.982	0.004	0.979	0.982	0.98	0.998	index_and_thumb
	0.98	0.002	0.992	0.98	0.986	0.998	index_and_middle
	0.924	0.007	0.942	0.924	0.933	0.994	thumb_only
	0.964	0.003	0.978	0.964	0.971	0.998	thumb_and_Pinky
Weighted Avg.	0.972	0.006	0.972	0.972	0.972	0.997	

=== Confusion Matrix ===

a	b	c	d	e	f		<-- classified as
23179	127	52	5	114	75		a = open_arm
145	23493	38	6	393	60		b = full_fist
18	146	18622	24	78	71		c = index_and_thumb
67	168	107	20150	70	0		d = index_and_middle
193	474	146	122	12079	54		e = thumb_only
157	129	63	0	88	11675		f = thumb_and_Pinky

Appendix C

Server Application Program

```
import sys
import Leap, thread, time
from Leap import CircleGesture, KeyTapGesture, ScreenTapGesture, SwipeGesture

import socket
def sendUdpMsg(ip, port, msg):
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.sendto(msg, (ip, port))
    sock.close()

class SampleListener(Leap.Listener):

    def on_init(self, controller):
        print "Initialized"

    def on_connect(self, controller):
        print "Connected"

    def on_disconnect(self, controller):
        print "Disconnected"

    def on_exit(self, controller):
        print "Exited"

    def on_frame(self, controller):
        # Get the most recent frame and report some basic information
        frame = controller.frame()

        # Get hands
        for hand in frame.hands:

            normal = hand.palm_normal
            direction = hand.direction
            # Get arm bone
            arm = hand.arm

            completestr = ""
            if len(hand.fingers) == 5:
                # Calculate the hand's Pitch, roll, and yaw angles
                completestr = str((round(direction.Pitch * Leap.RAD_TO_DEG,2), round(normal.roll *
Leap.RAD_TO_DEG,2), round(direction.yaw * Leap.RAD_TO_DEG,2), round(arm.direction[0],2),
round(arm.direction[1],2), round(arm.direction[2],2), str(arm.wrist_position), str(arm.elbow_position))))+ " "
                # Get fingers
                mystr = ""
                for finger in hand.fingers:
                    # Get bones
                    for b in range(0, 4):
                        bone = finger.bone(b)
                        mystr += str(round(bone.direction[0],2) ) + " " +str(round(bone.direction[1],2) ) + " "
+str(round(bone.direction[2],2) )+ " "
                completestr += mystr

            # The variable completestr has all the measurements just like in the sampling process. This measurements are
            passed to the predictive function generated from the WEKA J48 algorithm that will give us the event gesture. This
            gesture is then send with a udp message to the Raspberry Pi. The listener there will interpret it and drive the motors
            accordingly.

            pred = predictPosition(completestr)
            print pred
            sendUdpMsg("192.168.10.30", 10000, pred)
```



```

def predictPosition(x):
    x = x.replace(")", " ").replace("(", " ").replace("'", " ").strip().replace(" ", ",").replace(",", ";").replace(";",
";").replace(",,", ";").replace(";;", ";")
    #print x
    values = x.split(",")

Pitch,roll,yaw,arm_direction0,arm_direction1,arm_direction2,arm_wrist_position0,arm_wrist_position1,arm_wrist_positi
on2,arm_elbow_position0,arm_elbow_position1,arm_elbow_position2,bone_direction_0_0_0,bone_direction_0_0_1,bon
e_direction_0_0_2,bone_direction_0_1_0,bone_direction_0_1_1,bone_direction_0_1_2,bone_direction_0_2_0,bone_dire
ction_0_2_1,bone_direction_0_2_2,bone_direction_0_3_0,bone_direction_0_3_1,bone_direction_0_3_2,bone_direction_
1_0_0,bone_direction_1_0_1,bone_direction_1_0_2,bone_direction_1_1_0,bone_direction_1_1_1,bone_direction_1_1_2
,bone_direction_1_2_0,bone_direction_1_2_1,bone_direction_1_2_2,bone_direction_1_3_0,bone_direction_1_3_1,bone
_direction_1_3_2,bone_direction_2_0_0,bone_direction_2_0_1,bone_direction_2_0_2,bone_direction_2_1_0,bone_direc
tion_2_1_1,bone_direction_2_1_2,bone_direction_2_2_0,bone_direction_2_2_1,bone_direction_2_2_2,bone_direction_2
_3_0,bone_direction_2_3_1,bone_direction_2_3_2,bone_direction_3_0_0,bone_direction_3_0_1,bone_direction_3_0_2,
bone_direction_3_1_0,bone_direction_3_1_1,bone_direction_3_1_2,bone_direction_3_2_0,bone_direction_3_2_1,bone_
direction_3_2_2,bone_direction_3_3_0,bone_direction_3_3_1,bone_direction_3_3_2,bone_direction_4_0_0,bone_directi
on_4_0_1,bone_direction_4_0_2,bone_direction_4_1_0,bone_direction_4_1_1,bone_direction_4_1_2,bone_direction_4_
2_0,bone_direction_4_2_1,bone_direction_4_2_2,bone_direction_4_3_0,bone_direction_4_3_1,bone_direction_4_3_2 =
tuple([float(myval) for myval in values])
    if bone_direction_2_3_2 <= 0.64:

if bone_direction_2_3_2 <= 0.41:
    if bone_direction_1_3_2 <= 0.52:
        if bone_direction_4_2_2 <= 0.54:
            if bone_direction_0_3_1 <= 0:
                if arm_elbow_position0 <= 48.0034:
                    if bone_direction_2_3_0 <= 0.75:
                        if bone_direction_1_2_2 <= -0.21:
                            if bone_direction_4_2_2 <= 0.09:
                                if arm_wrist_position1 <= 115.574: r="full_fist"
                                else:
                                    if bone_direction_1_1_0 <= -0.54: r = "thumb_and_Pinky"
                                    else:
                                        if arm_elbow_position1 <= 159.182:
                                            if bone_direction_0_3_2 <= 0.73: r="thumb_only"
                                            else: r="full_fist"
                                        else:
                                            if bone_direction_1_1_1 <= -0.31: r="open_arm"
                                            else:
                                                if arm_elbow_position2 <= 413.745:
                                                    if arm_direction1 <= 0.14:
                                                        if bone_direction_3_2_2 <= -0.11:
                                                            if bone_direction_0_1_1 <= -0.5:
                                                                if bone_direction_1_1_1 <= 0.88: r="thumb_only"
                                                                else: r="thumb_and_Pinky"
                                                            else: r="thumb_only"
                                                            else: r="thumb_and_Pinky"
                                                        else:
                                                            if bone_direction_4_1_2 <= 0.56: r="thumb_only"
                                                            else: r="open_arm"
                                                            else: r="thumb_and_Pinky"
                                                    else: r="thumb_and_Pinky"
                                                else:
                                                    if bone_direction_0_2_2 <= 0.58: r="thumb_only"
                                                    else:
                                                        if bone_direction_0_1_1 <= -0.13: r="index_and_thumb"
                                                        else: r="full_fist"
                                                    else:
                                                        if bone_direction_1_1_1 <= 0.3: r="index_and_thumb"
                                                        else: r="thumb_and_Pinky"
                                                    else:
                                                        if bone_direction_2_2_1 <= 0.5:
                                                            if arm_direction1 <= 0.27: r="thumb_only"
                                                            else:
                                                                if arm_wrist_position2 <= 42.9115: r="open_arm"

```

```

        else: r="thumb_only"
    else:
        if bone_direction_0_1_1 <= -0.04:
            if bone_direction_0_2_2 <= 0.87:
                if arm_wrist_position2 <= 80.48:
                    if bone_direction_0_3_2 <= 0.02: r="thumb_only"
                    else: r="open_arm"
                else: r="thumb_only"
            else: r="full_fist"
        else: r="full_fist"
    else:
        if bone_direction_0_2_0 <= -0.65: r="thumb_only"
    else:
        if bone_direction_0_3_0 <= 0.67:
            if arm_elbow_position1 <= 234.934:
                if bone_direction_2_1_1 <= -0.21: r="index_and_thumb"
            else:
                if arm_wrist_position0 <= 48.5787:
                    if arm_elbow_position2 <= 385.774:
                        if bone_direction_0_2_2 <= 0.2: r="thumb_only"
                    else:
                        if bone_direction_3_2_1 <= 0.15:
                            if bone_direction_1_0_1 <= 0.38: r="thumb_and_Pinky"
                            else: r="full_fist"
                        else:
                            if arm_direction2 <= -0.83:
                                if bone_direction_1_2_2 <= -0.92:
                                    if bone_direction_0_2_1 <= 0.23: r="thumb_only"
                                    else: r="full_fist"
                                else:
                                    if bone_direction_4_0_2 <= 0.97: r="full_fist"
                                    else:
                                        if bone_direction_0_1_2 <= 0.84: r="thumb_only"
                                        else: r="full_fist"
                            else:
                                if bone_direction_3_2_2 <= -0.41: r="thumb_only"
                                else: r="full_fist"
                        else:
                            if arm_elbow_position1 <= 176.617: r="full_fist"
                            else: r="index_and_thumb"
                    else:
                        if arm_direction2 <= -0.96: r="thumb_only"
                        else: r="full_fist"
                else: r = "thumb_and_Pinky"
            else:
                if bone_direction_1_1_2 <= 0.92:
                    if bone_direction_1_3_1 <= 0.5:
                        if arm_elbow_position0 <= -18.4167: r = "thumb_and_Pinky"
                        else: r = "thumb_only"
                    else: r = "full_fist"
                else: r = "index_and_thumb"
    else:
        if bone_direction_3_2_2 <= 0.06:
            if arm_elbow_position0 <= 81.6317:
                if bone_direction_0_1_0 <= -0.66: r = "index_and_thumb"
            else:
                if bone_direction_0_3_1 <= 0.6: r = "thumb_and_Pinky"
            else:
                if bone_direction_1_1_0 <= -0.62: r = "thumb_and_Pinky"
                else: r = "open_arm"
        else: r = "open_arm"
    else:
        if arm_direction0 <= -0.26:
            if bone_direction_3_1_1 <= 0.72: r = "open_arm"
            else: r = "thumb_and_Pinky"
        else:
            if bone_direction_3_2_1 <= -0.15: r = "index_and_thumb"

```

```

else:
    if bone_direction_1_1_0 <= -0.46: r = "thumb_and_Pinky"
    else:
        if bone_direction_0_3_1 <= -0.22: r = "thumb_and_Pinky"
        else: r = "full_fist"
else:
    if yaw <= -53.78: r = "full_fist"
    else:
        if bone_direction_3_3_2 <= 0.01: r = "index_and_thumb"
        else: r = "open_arm"
else:
    if bone_direction_3_3_2 <= 0.26:
    if bone_direction_1_3_2 <= 0.5: r = "index_and_thumb"
    else:
        if bone_direction_4_2_2 <= 0.47:
        if bone_direction_2_2_1 <= -0.2:
            if bone_direction_1_2_0 <= 0.16: r = "index_and_thumb"
            else: r = "index_and_middle"
        else:
            if bone_direction_3_1_2 <= 0.75:
            if arm_elbow_position1 <= 82.7269:
                if arm_direction2 <= -0.88: r = "thumb_only"
                else: r = "index_and_middle"
            else:
                if bone_direction_0_1_2 <= 0.56:
                if bone_direction_1_1_2 <= 0.79: r = "thumb_only"
                else: r = "index_and_middle"
            else:
                if bone_direction_3_3_0 <= 0.6:
                if bone_direction_0_1_1 <= -0.24:
                    if bone_direction_2_3_2 <= 0.92: r = "thumb_only"
                    else: r = "index_and_middle"
                else:
                    if bone_direction_0_1_0 <= 0.68:
                    if bone_direction_4_1_2 <= 0.83:
                        if arm_wrist_position1 <= 148.039:
                        if bone_direction_1_2_2 <= 0.98: r = "index_and_middle"
                        else: r = "thumb_only"
                    else: r = "index_and_middle"
                    else: r = "index_and_thumb"
                else:
                    if bone_direction_1_1_0 <= 0.18: r = "index_and_thumb"
                    else: r = "index_and_middle"
            else:
                if arm_direction2 <= -0.92:
                if bone_direction_1_3_2 <= 0.83: r = "index_and_thumb"
                else: r = "index_and_middle"
            else: r = "thumb_only"
        else:
            if bone_direction_1_0_0 <= 0.06:
            if arm_direction2 <= -0.86: r = "thumb_only"
            else: r = "open_arm"
        else:
            if bone_direction_3_1_0 <= 0.19: r = "index_and_middle"
            else: r = "index_and_thumb"
    else:
        if bone_direction_1_1_2 <= 0.85: r = "thumb_only"
        else: r = "index_and_thumb"
else:
    if bone_direction_4_1_1 <= 0.64:
    if bone_direction_1_2_2 <= 0.27:
    if bone_direction_4_1_2 <= 0.97:
        if bone_direction_2_2_2 <= 0.81: r = "thumb_only"
        else: r = "index_and_thumb"
    else: r = "index_and_middle"
    else:
        if bone_direction_4_0_0 <= 0.53:

```

```

    if bone_direction_3_2_1 <= 0.5:
        if bone_direction_0_1_0 <= -0.52:
            if bone_direction_0_1_1 <= -0.13: r = "thumb_only"
            else: r = "open_arm"
        else:
            if arm_wrist_position2 <= 120.873:
                if arm_elbow_position1 <= 234.957: r = "open_arm"
                else:
                    if bone_direction_1_0_1 <= 0.08: r = "index_and_middle"
                    else: r = "open_arm"
            else:
                if bone_direction_4_0_0 <= 0.02: r = "open_arm"
                else: r = "thumb_and_Pinky"
        else:
            if bone_direction_2_3_2 <= 0.95:
                if bone_direction_4_1_0 <= -0.15:
                    if bone_direction_1_2_2 <= 0.7: r = "full_fist"
                    else: r = "open_arm"
                else: r = "full_fist"
            else: r = "index_and_middle"
        else:
            if bone_direction_1_2_2 <= 0.92:
                if arm_direction0 <= -0.33: r = "index_and_thumb"
                else: r = "full_fist"
            else:
                if bone_direction_3_2_2 <= 0.82: r = "thumb_only"
                else:
                    if arm_direction2 <= -0.99: r = "open_arm"
                    else: r = "index_and_middle"
    else:
        if bone_direction_2_22 <= 0.71:
            if bone_direction_2_2_2 <= 0.64: r = "full_fist"
            else: r = "open_arm"
        else:
            if arm_direction2 <= -0.99: r = "thumb_only"
            else: r = "index_and_middle"
return r

```

```

def main():
    # Create a sample listener and controller
    listener = SampleListener()
    controller = Leap.Controller()

    # Have the sample listener receive events from the controller
    controller.add_listener(listener)

    # Keep this process running until Enter is pressed
    print "Press Enter to quit..."
    try:
        sys.stdin.readline()
    except KeyboardInterrupt:
        pass
    finally:
        # Remove the sample listener when done
        controller.remove_listener(listener)

if __name__ == "__main__":
    main()

```

Appendix D

LEAP Driver Application Program

```
import time
import threading
import RPi.GPIO as GPIO
# define clockwise and anticlockwise movements to Pi GPIO Pins
GPIO.setmode(GPIO.BCM)
clock_wise_en1=23
anti_clock_wise_en1 = 24

clock_wise_en2=22
anti_clock_wise_en2 = 27

GPIO.setup(clock_wise_en1, GPIO.OUT)
GPIO.setup(anti_clock_wise_en1, GPIO.OUT)

GPIO.setup(clock_wise_en2, GPIO.OUT)
GPIO.setup(anti_clock_wise_en2, GPIO.OUT)
# Set l-left and r-right to 0
# the clockwise and anticlockwise speed is restricted to 1000
l=0
r=0
p_cw_en1 = GPIO.PWM(clock_wise_en1, 1000)
p_acw_en1 = GPIO.PWM(anti_clock_wise_en1, 1000)
p_cw_en1.ChangeDutyCycle(0)
p_acw_en1.ChangeDutyCycle(0)

p_cw_en2 = GPIO.PWM(clock_wise_en2, 1000)
p_acw_en2 = GPIO.PWM(anti_clock_wise_en2, 1000)
p_cw_en2.ChangeDutyCycle(0)
p_acw_en2.ChangeDutyCycle(0)

p_cw_en1.start(0)
p_acw_en1.start(0)
p_cw_en2.start(0)
p_acw_en2.start(0)

#
myr=250
# Move engine loop to control the PWM of both motors
def moveen(rr,ll):
    if abs(rr) < 10 and abs(ll) < 10 :
        p_acw_en1.ChangeDutyCycle(0)
        p_cw_en1.ChangeDutyCycle(0)
        p_acw_en2.ChangeDutyCycle(0)
        p_cw_en2.ChangeDutyCycle(0)
    elif rr>0:
        p_cw_en1.ChangeDutyCycle(round(rr/(myr+1.0)*100))
        p_acw_en1.ChangeDutyCycle(0)
    elif rr<0:
        p_cw_en1.ChangeDutyCycle(0)
        p_acw_en1.ChangeDutyCycle(-round(rr/(myr+1.0)*100))
    if ll>0:
        p_cw_en2.ChangeDutyCycle(round(ll/(myr+1.0)*100))
        p_acw_en2.ChangeDutyCycle(0)
    elif ll<0:
        p_cw_en2.ChangeDutyCycle(0)
        p_acw_en2.ChangeDutyCycle(-round(ll/(myr+1.0)*100))
```

```

import socket
##Equalisation of Motor sequence to even out the motors when no gesture is
#given and the motors have different speeds

def equalize_motors():
    global l,r
    while True:
        if r>l+1:
            r-=1
            l+=1
        if l>r+1:
            l-=1
            r+=1

        moveen(l,r)
        print "eq", l, r
        time.sleep(0.0125)

# Gesture Controller procedure: gives instnction to the left and right motor depending on the gesture
def listener():
    global l,r
    r,l=0,0
    # Recieve UDP packed from the socket IP address port 10000
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
    sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    sock.bind(('192.168.10.30', 10000))
    mlist = []
    while True:
        data, addr = sock.recvfrom(2048) # buffer size is 1024 bytes
        if data=="full_fist":
            if l>0: l-=1
            elif l<0: l+=1
            if r>0: r-=1
            elif r<0: r+=1
        elif data == "open_arm":
            l+=1
            r+=1
            if l>myr: l=myr
        elif data == "thumb_only":
            l-=1
            r-=1
            if l<-myr: l=-myr
            if r<-myr: r=-myr
            # increase scalar values for l and r if motors are desired to rotate faster in either direction
        elif data == "index_and_thumb":
            l-=5
            r+=5
            if l<-myr: l=-myr
            if r>myr: r=myr
        elif data == "index_and_middle":
            l+=5
            r-=5
            if l>myr: l=myr
            if r<-myr: r=-myr
        elif data == "thumb_and_pinky":
            if l>0: l-=5
            elif l<0: l+=5
            if r>0: r-=5

```

```

        elif r<0: r+=5

    print "data:", data, l, r
    moveen(l,r)

NUM_OF_THREADS = 1
def init():
    global NUM_OF_THREADS
    thread_on_listener = threading.Thread(target = listener, args = ( ))
    thread_on_listener.daemon=True
    thread_on_listener.start()
    NUM_OF_THREADS += 1
    thread_on_eq = threading.Thread(target = eqlize_motors, args = ( ))
    thread_on_eq.daemon=True
    thread_on_eq.start()
    NUM_OF_THREADS += 1
#test_en2()
#test_en1()
#listener()

if __name__ == "__main__":
    init()
    while threading.active_count() > 0:
        time.sleep(0.1)
        if threading.active_count() < NUM_OF_THREADS: sys.exit()
    GPIO.cleanup()

```

Appendix E

Full Bridge Motor Driver L298N

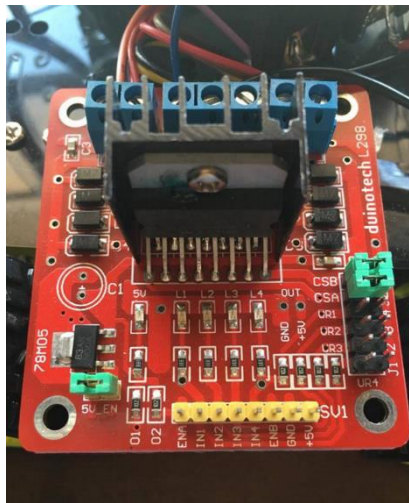


Figure 19: L298N Stepper Motor Driver Controller Board for Arduino

L298N Motor Driver Board Features:

- High Voltage High Current capability
- Heavy Load Heat Sink protection
- 4 x Pullup switches
- Motor Driver Indication LEDs and 5V power indicator LED
- Screw Terminals for Motors
- Light Weight
- Small Dimensions

L298N Motor Driver Board Specifications:

- Driver Chip: L298N
- Input Voltage Range: 6V to 35V
- Peak Current: 2A
- Logic Voltage: 5V to 7V
- Logic Current: 0.36 mA
- Max Driver Power: 25W at 70°C

Raspberry Pi 2 Specs



Raspberry Pi 2.0 Specifications:

- 1 GB RAM
- 900MHz quad core ARM CPU
- 40 GPIO Pins
- 4 USB ports
- HDMI and Ethernet port
- Micro SD card slot
- Input Voltage 5V
- Input Current 1A

Table 14: Robot Platform GPIO to L298N Arduino Motor Driver Board Pin Connections

Raspberry Pi connections	Raspberry Pi 2 Pins	L298N motor driver full H Bridge Pins
+5V	40	EN1
+5V	39	EN2
Ground	38	GND
GPIO 23	33	IN4
GPIO 24	34	IN3
GPIO 22	8	IN2
GPIO 27	7	IN1

Bill of Materials [BOM]

Table 15: BOM for Robot Platform Base

Item	Description	Cost	Quantity	Total Cost
LEAP Motion Sensor	Controller used for project	1	\$79.99	\$79.99

Laptop with processor and graphics card	Essential for the LEAP motion sensor to work efficiently	1	-	-
Raspberry Pi 2.0		1	\$49.99	\$49.99
L298N Motor Driver PCB Board	contains motor driver chip L298N	1	\$13.99	\$13.99
Robot Base	hold the Raspberry Pi and is the receiving and responsive end test product	1	\$49.99	\$49.99
Small DC Motors	3V	4	\$2.95	\$11.80
Ribbon Cable	jumper cable for clean and easy development	1	\$3	\$3
2600mAh Lithium Batteries	Supply the motors with 2 - 4 Amps	2	\$19.95	\$39.90
Battery Bank 5V 1A	Power to the Raspberry Pi	1	\$15.00	\$15.00
Wi-Fi Dongle	communication between the Raspberry Pi and the laptop with Leap motion Sensor	1	\$30.00	\$30.00
SD card	For Raspberry Pi; contains the motor driver code	1	\$15.90	\$15.90
Camera	monitoring system	2	\$3.00	\$6.00
Total				

Table 16: Expendable materials list for Robot Platform

Expendable items/ Tools required	Description
Screw Driver and various tools	assembling the robot base
Thin Solder	securing connections on the robot base
Soldering Iron	Raspberry Pi shield development
stackable headers	Raspberry Pi shield and Motor Driver board connection
USB connectors	Battery bank to Raspberry Pi connection
Wire	motors and battery connection leads