Automatic
Control &
Systems
Engineering.

Gesture control with LeapMotion

Kshitij Goel

9th May 2016

Supervisor: Linda Gray

A dissertation submitted in partial fulfilment of the requirements for the degree of Mechatronic and Robotic Engineering

Executive Summary

Introduction

This document provides an example of a proof of concept project which involves the theory of human-computer interaction. It involves the usage of hand gestures as an input to a computer and uses it for manipulation of a globally remote controlled ground vehicle.

The project aims to link different modules to work in coherence of each other and perform a task as required and ordered by the user. The task is completed by using a hardware called LeapMotion to capture and recognize hand gestures and then further process it to show a textual feedback of a successful recognition to the used with the help of a screen on a computer. The computer then transmits signals to a remotely located ground vehicle with access to the internet. The transmission of signals is done with the help of third party solution as no native support for such type of communication is available at either of the devices. A corporation called Pubnub is used which helps easy transmission of small data between devices using a secure connection. The ground vehicle comprises of a small computing device which is a Raspberry Pi. The Raspberry Pi receives the signals sent from the computer and processes it to control four motors to perform dedicated tasks. A piece of hardware is present between the Raspberry Pi and the motors called motor controller as the signals from the Raspberry cannot be directly understood by the motor.

The main deliverables of the project are the remote ground vehicle, the code for each module and the complete working system.

Aims and Objectives

The goal of the project was to create a proof of concept system which uses hand gestures as an input and manipulates a remote hardware as an output.

The objectives of the project can be broadly classified into:

1. Research on human-computer interfacing.
2. Research on hand gestures as input.
3. Receive data from the LeapMotion controller.

4. Categorizing the data received from the controller and further processing it to recognize hand gestures.
5. Research on ways to communicate with the devices.
6. Transmitting appropriate signal to the remote controlled device.
7. Receiving the signal on the remote controlled device and further processing it to perform required function.
8. Test the complete system and present it at the oral presentation.

Achievements

The research was carried on ways of human computer interfacing and devices that can be used to accomplish the task. Researching on an appropriate hardware to use for the capturing of hand gestures. Successfully developed the program for gesture recognition and giving textual feedback to the user. Developed a globally remote controlled ground vehicle and successfully tested the complete system and optimized the process for accurate results.

Conclusion

The project has successfully created a working model for the proof of concept of using hand gestures as an input and further extending the capability to manipulate a globally remote controlled vehicle as an output. The project provided important insight on ways in which humans can communicate with a computer other than traditional mouse and keyboard. It also helped many protocols which need to be followed when communicating between two devices like Secure Socket Layer (SSL). The project can be further improved by generating a video stream as feedback to the user which would be necessary before the project can be deployed for use in the field.

Abstract

The project is a proof of concept for the theory of human-computer interfacing which uses the input of hand gestures for manipulation of a globally remote controlled ground vehicle as intended. The project intends to diminish the distinction between the passive graphical user interfaces, and the real world and apply the theory to make more intuitive and immersive environments for working for individuals. The project looks at the problems faced by creative developers who use software environments like Maya, SolidWorks and many more and face problems for the manipulation of work environment due to lack of proper input devices.

The project uses the hardware called LeapMotion for gesture capturing and recognition and uses the web services to communicate with the globally remote controlled ground vehicle powered by a Raspberry Pi. The hand gestures manipulate the movement of the robot in four basic directions which are moving forward, backward, turning right and turning left. The Raspberry Pi communicates with the motor controller as a middle-man device which further communicates with the motors for appropriate movements. The remote ground vehicle has the functionality of rotating on the spot rather than taking a huge circle for turning provide it advantages of better maneuverability. Since the computer to which the LeapMotion controller is connected, and the ground vehicle are not on the same network, third party solutions are used for communication. An enterprise named Pubnub is utilized for the communication process which provides the features of secure transmission of signals between the devices using Secure Socket Layer (SSL) encryption.

Acknowledgement

My experience while performing the project was amazing due to the presence of so many people around me who helped me in various ways. I would like to start by thanking my supervisor Mrs. Linda Gray who helped me and supported me at every decision and stage of the project while motivating me to work harder when faced with a difficulty. I would also like to thank lab technician Craig Bacon who helped me in the process of building the ground vehicle while his constant support with the maintaining the hardware. My second reader, Dr. Robin Purshouse who motivated me to apply for further funding for the resources needed for the project and also providing me feedback at different instances of time. I would then like to thank my parents for their constant moral and emotional support towards my achievements. My friends who guided me through various processes for obtaining different kinds of information and helped me structure this report.

Thank you everyone.

Table of Contents

1. Introduction

1.1 Background and Motivation

The aim of the project is to create a proof of concept for the theory of human-computer interaction by using hand gestures as an input and globally remote control a ground vehicle. This is accomplished by using a LeapMotion controller which is connected to the computer and is used for the tracking of hand movements and converting them to recognizable gestures. When the gestures are recorded, an appropriate message is sent to the ground vehicle for further processing. A computing module on the ground vehicle receives the messages from the laptop and further communicates with the motor driver and motors for performing the specified action. The wireless communication between the laptop and the remote computing module Raspberry Pi is done with the help of connectivity through the Internet. The communication is done with the support of a third party solution called Pubnub which allows global connectivity between the two devices.

The project utilizes the concept of human-computer interfacing to diminish the difference between passive computer interfaces and the real world by making use of the natural hand movements as an extension of the arm. The result of such a process is a user interface that is more intuitive, natural and more immersive so that the experience of using a computing device is more enjoyable for the user while providing better functionality.

It is inspired by the imagination of, interacting with computing devices using holographs as an interface. The project builds to this idea and provides with a working model for hand gesture as an input to a computing device. The input of these hand gestures can be used by various creative developmental environments and for manipulation of physical objects when the technology can be fully developed to perform such tasks.

1.2 Problem Definition

The point of the project is to provide an example of using hand gestures as an input to a computing device and show that such an input is better than the old-fashioned inputs through mouse and a keyboard. The input of hand gestures stands on the base of human computer interfacing and brings the world of computing and natural use of arm as an input closer. The project uses the application of this principle by manipulating the movement of a globally remote vehicle, which shows that the input is not limited to a local machine and can be used to manipulate any object worldwide provided that the user has access to it.

1.3 Aims and Objectives

The project can be broken down into two most important aspects namely, the gesture recognition and the control transfer to prove its recognition by a physical movement. The aim of the project is to provide a working scenario of recognition of hand gestures as an input and can be clearly pointed out as:

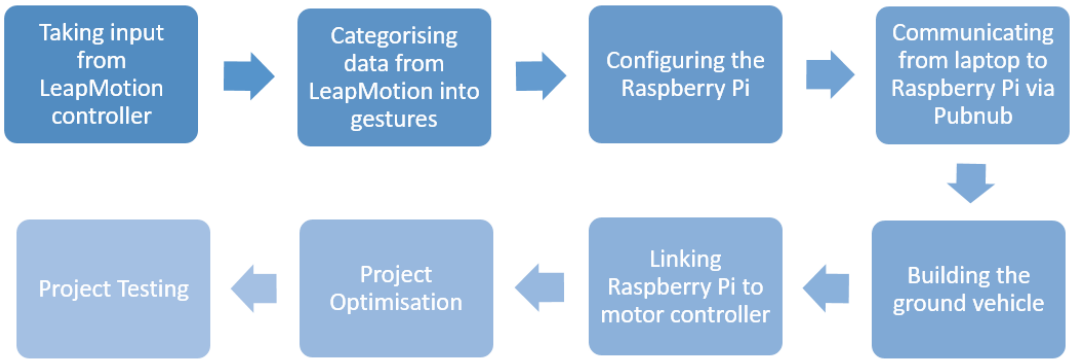
1. Performing background research on gesture control and displaying relevant data received from interfacing device to the laptop.
2. Specifying the configuration of different parts of the hardware i.e. the interfacing and the computing module on the remote ground vehicle.
3. Defining the types of movement of the remote ground vehicle.
4. Defining the system control flow with an accurate description of each module of the project.
5. Cost estimation after including all building material and hardware and proposing appropriate UK supplier for each.
6. Connecting the LeapMotion hardware to the laptop and processing data received from it and thus, generating a little textual feedback to the user.
7. Analyzing the received data from the LeapMotion and categorizing it into different gestures.
8. Connecting the LeapMotion to the internet and uploading the relevant commands for the globally remote ground vehicle.
9. Building the remote ground vehicle according to the configuration already decided.

10. Connecting the remote computing module to the internet for accessing the commands sent from the LeapMotion.
11. Writing and compiling software code for each module according to secure socket layer protocol.
12. Testing and optimizing the system for uninterrupted operation.

1.4 Project Management

The project was well managed in terms of task and time with the help of project planning at the start of the project. A Gantt chart was created at the beginning of the project which was followed strictly to overcome any delays in deliverables and task completion. A log book was generated to record the progress of the project and was shared with the supervisor using Google Drive. The log book recorded the events of the weekly meeting with the supervisor, task to be completed until next meeting and deliverables for each meeting.

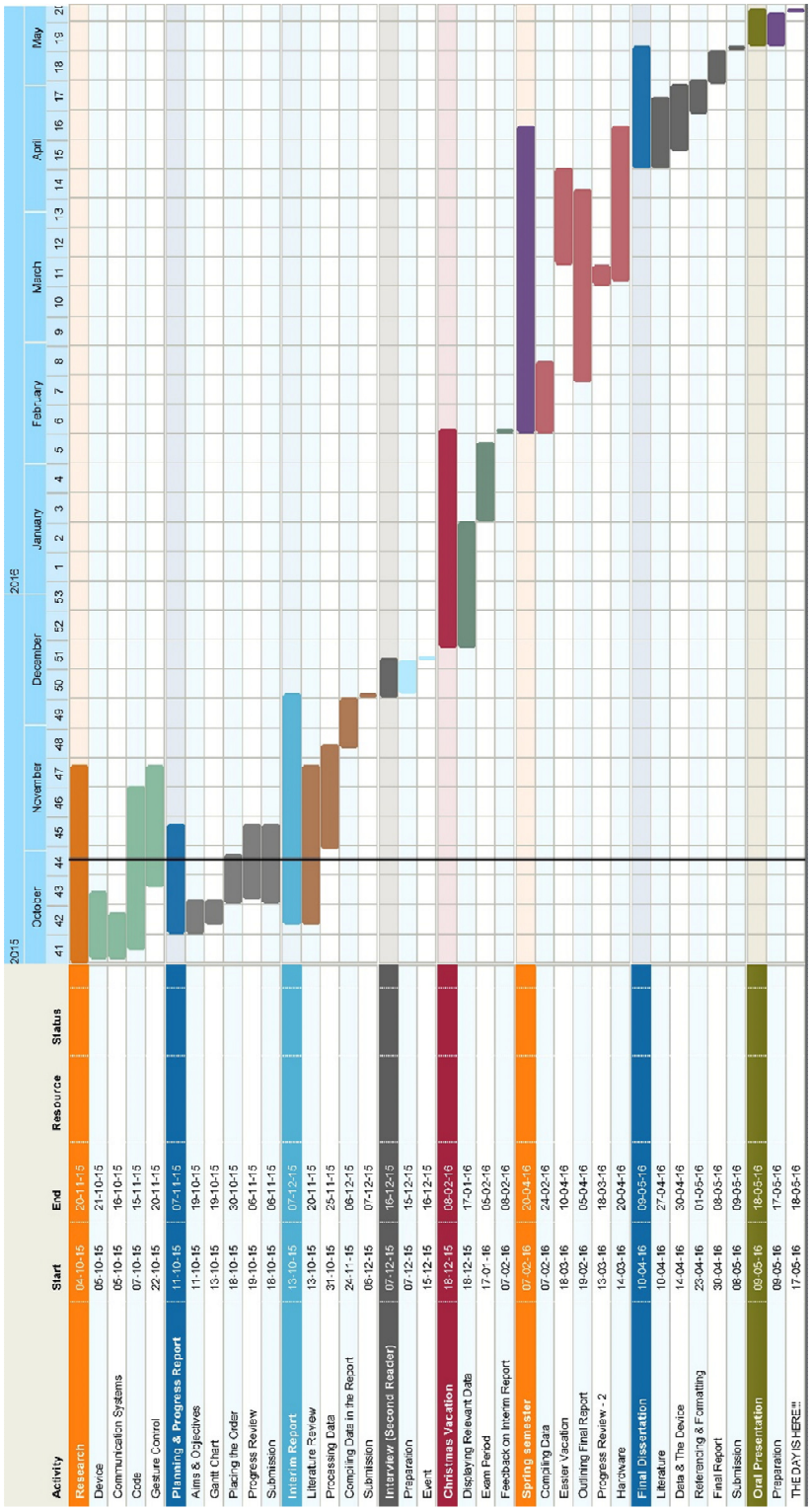
The complete project was time-lined according to the flow chart shown below.



The main deliverables for the project were:

- Aims and Objectives form
- Gantt Chart
- Progress Review form 1
- Code for communication with LeapMotion controller
- Interim Report
- Progress Review form 2
- Code for Raspberry Pi processes
- Log Book
- Dissertation

The Gantt chart used for the project planning remained constant since its creation till project completion[1]. The project followed all the deadlines and dates for task completion. Some tasks were deliberately left vague in the Gantt chart to overcome any setback or delay faced which helped to keep the project on track of its timeline.



2. Literature Review

The project is a proof of concept which made it harder to lay ground work on previously done projects or published papers. Initially, fresh ideas for interfacing between different kinds of devices were brainstormed which revealed many ways in which humans interact with the machines around them. Each method of interaction represents how unique and different each method is while building on the concept that the application of the same is far more tedious and gruesome due to the complications in the application. Proper protocols need to be followed while developing a new method of interaction as it not only helps the consumer interact more easily but also lets future developments add more functionality to the system. We interact with ticket collection tills at railway stations, touch screens on our mobile phones, motion mimicking prosthetics, movement understanding gaming systems, mind controlled devices and many more types of systems on a day to day basis. Each of these kinds of interactions implements a different strategy for the interpretation of the inputs and have various kinds of outputs accordingly[2]. Some are based on embedded systems to perform the task allotted while others depend on microprocessors which can be tweaked according to the need of the software.

2.1 Human Computer Interaction

Human-computer interaction is a branch of computer science dedicated for the ease of use of software or hardware for an end user, customer or a client where the lines of distinction between the machine operated interfacing and natural human movements tend to diminish. The interaction of two entirely different types of systems is an idea that makes the modern graphical user interfaces intuitive, responsive and natural on any device, be it stationary or portable. Human-computer interaction thus, brings close the computer and the human world to a level where the homo-sapiens see the computer as an extension of their body rather than an external device[3]. Our society has evolved from an era of carving paintings on the wall as a primal instinct to using a mouse and a keyboard for interaction. Some decades ago, we used changing the position of wires which then evolved to turning switches ON/OFF to command line processors and finally has reached the position where we see it today at a graphical user interface. This change has been drastic and has been following Moore's law for a long time, but we have reached a moment in our history where we have stopped developing new and

better ways of interaction with the machines. A new breakthrough is needed as the current systems are still not perfect for interacting with a device.

Many disadvantages come to mind when we think of current technology and how it can be made better, faster and smarter for a more immersive and enjoying experience for the user as the lack of natural instinct is not yet fulfilled as compared to drawing with bare hands. This proposition has many positive aspects to it as it will help extend the natural hand movements.

The theory of human-computer interaction is not limited to its use by a particular set of people who interact with computing devices more than normal. It can apply vastly to humans ranging from physically fit to neck-down physically challenged. Special systems have been developed to make everyday life easier for everyone. A physically impaired person interacts with a prosthetic actively for better performance than just mechanical ones. Eye tracking is an application for human-computer interaction which allows individuals who are handicapped below their necks to interact with systems around them[4]. Systems like these help people to lead their lives in a simpler and better way and act as a small step for the betterment of the world. The systems that we have developed so far provide a limited point of perspective to global advancement in the area of human-computer interaction, and much development is needed for the systems to reach a perfect level of intuitiveness to create an immersive environment for interaction[5]. This project focuses on a particular aspect of human-computer interfacing called hand gestures to establish a benchmark for future projects to set upon.

2.2 Hand gestures

In the modern world, the types of the need of the end user vary a lot based on the latter's interests and hobbies. A regular office personnel will use a computer for documents, presentations, and data handling, a photographer would use it as a media editing suite while a gamer would use it for gaming and entertainment. Although we do not see any gaps in communication in these scenarios, a major industrial usage of creative development environments is always left out which with the current set of input devices and communication methods, makes it very hard to perform at optimum capacity. This issue is actively faced by graphic designers, 3-D modeling and animation personnel and people who perform simulation in either real time of passively. A new demand for a way of communication has been rising

in the market for quite some time, which suggests that more ways of interactions need to be developed so that the man-hours wasted while trying to manipulate a 3-Dimensional object in these creative development environments on a 2-Dimensional screen can be invested more productively. These situations give rise to a new scenario where a 3-Dimensional control of objects come into play, as it makes it easier to manipulate the components of the project in progress, while having a primitive advantage of acting as an extension of the user's arm. The need of such a 3-Dimensional control is fulfilled with the help of hand gestures[6]. These hand gestures can be used in various upcoming areas, and with the combination of other ways of communication like holography and virtual reality to make a more immersive and natural experience for working[7]. The application includes the creation and manipulation of a 3-D model on popularly known creative developmental software products like Maya, SolidWorks, Unity and many more. These can also be integrated with virtual reality systems or holographic systems to bridge the futuristic ways of communication, to the present[2]. These types of interaction open up new areas of research and application, which would bring the variable of dynamic into play with modern interacting devices.

Although the use of hand gestures has many advantages to the currently used set of input devices to a computer, many complications arise with its application like the recognition of the gesture, the integration with the application and the security such that only the specified user can make inputs.

2.3 Gesture Recognition

Every detail of the project had to be thought of, in a way that compliance with other modules will not create an issue at later stages of the development. This meant that the fact of choosing a language for programming of the modules, had the same amount of weight as choosing the proper configuration of the motor driver. For capturing hand gestures and making them computer understandable, an important part was the hardware that would be used while being in accordance with the budget available. Many types of cameras and imaging devices were taken into consideration, but proper software development kit and environments were not available for them all. For the project, we needed a hardware that can easily

understand the movements of hands so that easy manipulation of the data received from the hardware, can be performed.

Many types of hardware are available for the consumer to customize according to specific needs which include 3-D cameras[8], movement recognizing mouse, high-performance grade movement recognizing infrared cameras like Kinect[9]. Each of these modules has an advantage and disadvantage connected to itself which were considered while comparing it to the scope of the project. The Xbox Kinect sensor is majorly known for high-performance gaming environment application while being a closed software development environment due to the limitation in the application in different settings, as compared to other such hardware[10]. Although the Kinect sensor is gaming grade performance device, another disadvantage linking to it is the slow frame capture rate and availability of extra features which proves it to be less useful as these features add to the cost of the hardware. The 3-D camera, on the other hand, is better in the application with the open development environment, allowing more space for creative use and a wide range of coverage, still doesn't fit the profile for the project, as it provides a 3-D image as the result[11]. This makes the project more concentrated on the image processing rather than gesture-interpreting.

In the end, after specifying the pros and cons for each type of hardware taken into consideration, the LeapMotion controller was finalized by the availability of the software development kit as well as the cost and the features it provides for hand tracking[12]. The research was performed on projects done by people which used LeapMotion controller but did not provide many results as it is a proof of concept which made it even harder to start with the project[13]. The aspect of communication from the LeapMotion controller to the globally available remote device was still lacking due to lack of choices and knowledge of appropriate services.

2.4 Communication

Two different types of ways of communication exist which can be applied to this project. The first case is when the two modules of the project are on the same network and the second case when they are on different network.

2.4.1 Local Network

Communication between the laptop and the ground vehicle computing module i.e. Raspberry Pi is done within the presence of an existing virtual network where both the devices are connected to the same network and communicate to each other using the local internet protocol addresses. The communication is done via a secure shell i.e. ssh protocol which is an encrypted network protocol operating at the layer seven i.e. Application layer of the Open Systems Interconnection model (OSI model)[14]. This protocol of communication is necessary for a remote login and conducting other network related or local computational procedures. This protocol was designed to replace Telnet and other such old protocols on a UNIX based system as they sent the password for remote login over the connection in the form of plaintext.

The facility of ssh is not natively supported by Windows operating system. Thus, third party software can be used to extend these functionalities to the system. A well-known software for this is purpose is PuTTY, which provides the functionality to run commands in the terminal or even install software from repositories and run them.

2.4.2 Remote Access

Communication between the Raspberry Pi and the laptop or any other computing device is done with the help of an internet connection. This type of communication has emerged recently with the evolution of Internet of Things (IoT) and has been very helpful for remote connections. We have emerged from a time when remote connections can be made only via satellites or long range infrared communication methods. This evolved to the development of different ways which included the intranet, dedicated short range communications (DSRC) and Radio Frequency Identification (RFID). These methods, although were better than no connectivity, had their set of flaws and setbacks which were later overcome by the presence of global connectivity by the internet by either Internet Service Provider or Global Systems for Mobile Communications (GSM). With the presence of internet at any location, messages can be sent to the remote device and user requested task can be performed.

The idea of remote application to the project makes it completely different as it expands the scope to a whole new level of complication. Many hours were invested in finding a solution for this situation which resulted in the discovery of new protocols which try to accomplish this task but fail. A new approach was needed to overcome this and thus led to the application of third party solutions. An enterprise named Pubnub was used for the process due to the suitable features available for the communication between the laptop and the remote ground vehicle. The communication provides a maximum latency of 45ms between the messages hosting device and the devices subscribed for receiving it. The Raspberry Pi and the laptop were installed with the software development kits for the Pubnub enterprise, and the secure socket layer (SSL) protocols were followed for the communication[15].

2.5 Raspberry Pi

A Raspberry Pi is a credit card-sized single board computing device based on the stripped down version of Linux operating system. It has a Broadcom system on chip (SOC) with an ARM compatible CPU, clocking frequency ranging from 700 MHz to 1.2 GHz, a graphics processing unit and a random access memory ranging from 256 MB to 1 GB depending on the different version of the hardware. It uses a MicroSDHC card for its primary storage of the operating system and the user files and also has an RJ45 port for internet connection and 4 USB ports for other peripherals. It also has various I/O pins for communication to different sub-peripherals like motors, sensors or connection to 4-pin connectors while separating the display to either an Audio/Video port or HDMI 1.0 port. It can also be connected to a camera feed input via a dedicated port while still leaving the functionality to attach a touch screen device for communication via a display port[16].

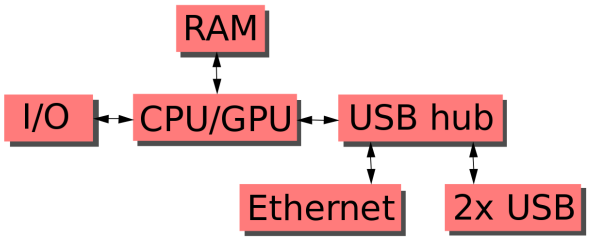


Figure 1: Components of the Raspberry Pi (Block Function) [15]

A Raspberry Pi 2 was used for the purpose of the project to run the basic python script for connection with the motors while running a background service for connection to the Pubnub server for real-time communication with the laptop. A Raspberry Pi is a very moldable piece of hardware which can be used for any purpose thinkable provided that it not be very resource consuming due to the limitation of a small processing unit[17]. The stripped-down version of the Linux on the Raspberry Pi comes pre-loaded with native support to many programming languages like C, C++, Python, Ruby, Java and Perl while having native hardware support for H264 codec for video and audio compression which adds up to the advantages for its application in the project.

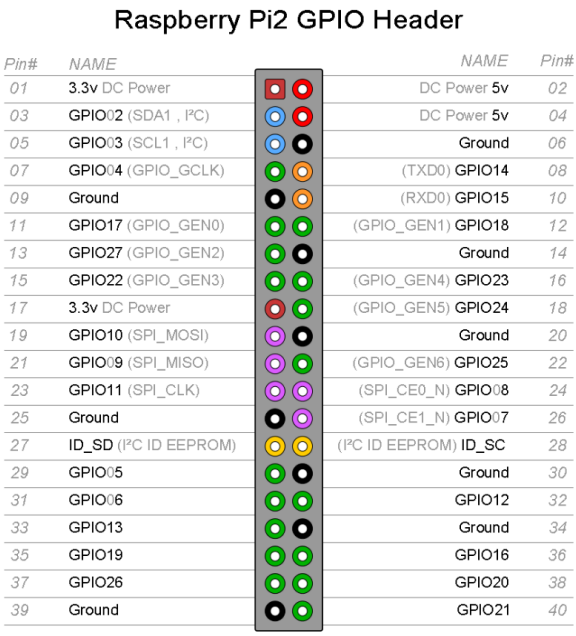


Figure 2: Raspberry Pi 2 GPIO configuration [19]

2.6 Remote Ground Vehicle

The remote ground vehicle is comprised of two different types of hardware:

1. Mechanical
2. Electrical
3. Power Source

The mechanical components include the 4 motors which control the 4 wheels for 4 distinctive types of movement which are:

1. Moving Forward
2. Moving Backward
3. Turn Right (Clockwise)
4. Turn Left (Anti-Clockwise)

The Electrical components of the ground vehicle include the computing module i.e. the Raspberry Pi and the motor controller while the power source refers to 2 components being a Li-Po battery for powering the four motors for each wheel and the source of energy for the Raspberry Pi.

The ground vehicle is built upon a base of a steel frame with hardware support for the suspension to all four wheels and brackets available for placing the camera in front of it. It also has mounting pillars for the computing module and the motor controller while leaving enough space for the two power sources. The motor controller was chosen according to the wheel movement requirement specifications. The wheel movement management is done as:

1. Move Forward – All four wheels move forward
2. Move Backward – All four wheels move backward
3. Turn Right (Clockwise) – The front left and left hind wheels to move forward while the front right and hind right wheels move backward
4. Turn Left (Anti-Clockwise) – The front left and left hind wheels to move backward while the front right and hind right wheels move forward.

This type of configuration provides a lot more flexibility and mobility for maneuvering the ground vehicle in tight places as it allows the vehicle to turn on the spot rather than take a circular turn with a large radius as adopted by many vehicles and robots being used. The two power sources are maintained for powering the device rather than one as the input for the computing module varies a lot than the input required by the motors to provide enough torque for the movement. The computing module takes a 5V 1A input whereas the wheels take 7.4V variable current input. Thus, choice of 2 different power sources helps maintain a stable working environment for all components of the vehicle.

The flow of the input from Raspberry Pi to motors include the presence of a motor driver in the middle as a direct connection would pose a fire threat to the components. The reason for this fire threat is that if the motors are connected to the Raspberry Pi, then they will draw power from it as well which would lead to a meltdown of other electrical components on the computing board. Thus, the power for the motors goes via the motor controller and the computing board only sends the signal to the motor controller for the wheels connection and direction of motion. Since each motor is manipulated individually, the motor controller that we use in the project is MDD10A by Cytron Technologies which is a Dual Channel 10A DC Motor Driver. The advantage of using this motor controller is the presence of test

buttons so that testing of a successful connection between the motor controller and the motors can be performed.

3. Procedure (Materials and Methods)

The project follows the flow of control as shown in the image below:

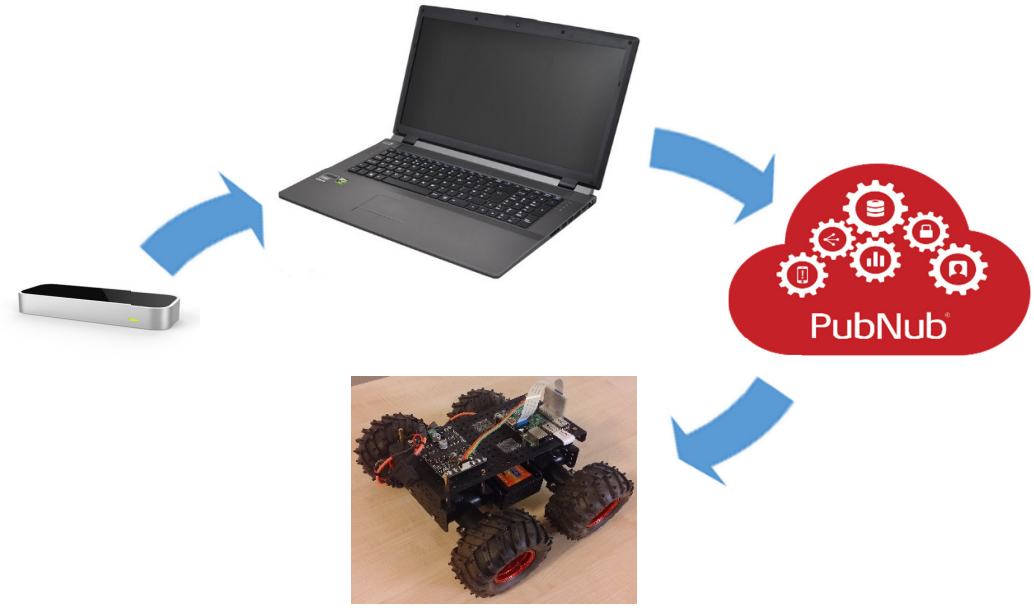


Figure 3: Control Flow diagram of the system [25]

The project thus, can be broken down into 3 different modules which are:

- LeapMotion
- Communication
- Remote Ground Vehicle

Each of the modules have several specifications that need to be fulfilled before the code for the project could be written. Programming languages that comply with the development kit while being supported by the computing module were tallied, and thus, a common language Python [26] was used for the project. Firstly, the software development kit for LeapMotion should be downloaded and installed to the processing computer. Then, the source path for the location of the development kit is to be included in the reference path of the python development environment. When the code has been written, for the LeapMotion controller and the recognition of hand gestures, the software development kit for Pubnub is downloaded and installed for use. The path of the installation of this developmental kit is then included in the reference path of the

python development environment. These steps make the python code ready for compilation and running processes. Further specifics of each module have been described below.

3.1 LeapMotion

Documentation for the LeapMotion controller is available on the official web-site which helps build the python program for hand tracking and gesture recognition. The program uses a complex algorithm for tapping to the device for capturing each frame from the hardware which registers the position. The documentation provided was a huge help as connecting the hardware after installing the required software development kit was confusing. Many problems were faced during this stage as the hardware needs specific USB 3.0 ports to function without delay due to high data transfer rates [21]. The controller takes a data input from the environment 200 times per second and is accurate up to 3mm for finger movements. The hardware performs all these tasks with the help of two infra-red cameras and three LED lights which help with the image capturing[18]. The data captured by the controller is sent to the laptop, and it is this data that contains specifics about the position of hands in 3-D space in the observable area. It is this data that is used to formulate and accurately recognize the relative position of hands from previous data capture and is then further processed. The data is stored in an object frames [23] which has containers of hands, pointables, fingers and tool and can be observed in the following code snippet:

```
frame = controller.frame()
hands = frame.hands
pointables = frame.pointables
fingers = frame.fingers
tools = frame.tools
```

Figure 4: Handler Call-back

The hands class has many attributes related to its structure which include hand identification, palm direction, position, normal to palm, palm velocity, confidence, etc. These attributes help in identifying which hand is being recognized by the controller, the number of fingers on each hand which is in an erect position and their spatial position [28]. The pointables class provides the

information about the type of object being detected by the controller i.e. if the object can be used to sketch or draw. This is majorly used when a pen, a pencil, a single finger or any sharp object is detected and used for drawing on a blank canvas[5]. The tolls class further breaks this down into categorizing it into fingers or other sharp objects. Although these attributes tell much about the hands and their position on the observable area, detection of gestures cannot be processed. Thus, the following code snippet is used for the purpose:

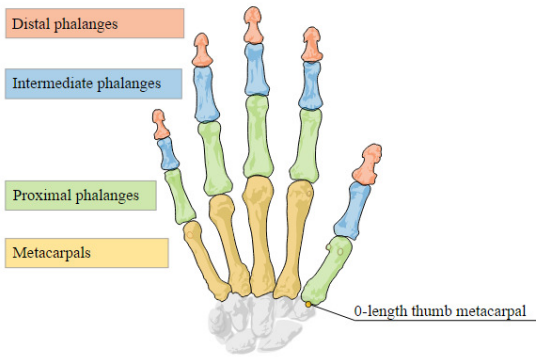


Figure 5: Bone Joints used for relative hand position[28]

```
if gesture.type==Leap.Gesture.TYPE_CIRCLE:
    circle=CircleGesture(gesture)

    # Determine clock direction using the angle between the pointable and the circle normal
    if circle.pointable.direction.angle_to(circle.normal)<=Leap.PI/2:
        clockwiseness="clockwise"
    else:
        clockwiseness="counterclockwise"
```

Figure 6: Circle Gesture Detection

The clockwiseness variable is used to determine if the circle gesture made in the observable area in a clockwise direction or anti-clockwise as each of these gestures are linked with an action on the remote ground vehicle [29]. The clockwise circle gesture makes the vehicle turn right while the anti-clockwise circle gesture makes the vehicle turn left.

Another gesture that is defined in the python script is the swipe gesture which is activated as shown in the following code snippet:

```
if gesture.type==Leap.Gesture.TYPE_SWIPE:
    swipe=SwipeGesture(gesture)
    swdir=swipe.direction
```

Figure 7: Swipe Gesture Detection

The swipe gesture is linked to the forward and backward movement of the ground vehicle while the swdir variable indicates the movement of the hand in 3-D space [30]. The front swipe is linked to move the remote ground vehicle move forward, and the back swipe makes the vehicle move back. The data provided by swdir variable is a set of 3-D space co-ordinates which by themselves make no sense. Thus, to properly disintegrate the data into the X, Y and Z-axis movements, the image on the right from documentation helped [30]. The dis-integration of data was thus done by swdir.z as its value ranged from -180 to +180 where it has the value 0 at the origin of the space observable by the controller.

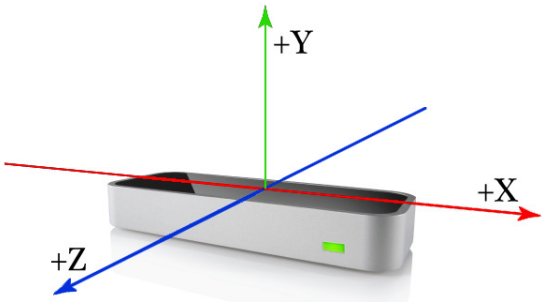


Figure 8: LeapMotion Co-ordinate System [30]

3.2 Communication

The communication between the devices was done with the help of third party solution which provided real-time data transmission and processing. The services of an enterprise called Pubnub were used for which a sign-up was required. The services of the company could only be used after installing their software development kit which was available for many programming languages and various situations. Although, the software development kit for Windows operating system was hard to install due to much server data being processed on a Linux-based operating system. A recompilation of the files was performed for Intel based chips and some dependencies were installed which included cryptology and encryption base files.

The service uses a publisher-subscriber method for real-time data streaming and device signaling which lets the user establish and maintain a persistent socket connection with a latency of less than 250ms. The user then gets a unique set of 2 keys namely publisher key and subscriber key. In the case of this project since the laptop sends the signal to the remote device, the laptop is the publisher, and the remote device is the subscriber. The messages are sent on a specified channel which can be named anything and then the messages are packed into

JSON packages. A JSON package is a JavaScript Object Notation, and it uses human readable data packets to transmit data[19]. When the subscriber sends a message on a particular channel, the message is forwarded to the server, and a time-stamp is added to it. This message is then transmitted to whoever has subscribed to that channel with the provided subscriber key receives it. If either of the subscriber key or the name of the channel is wrong, then the data sent would not be received as each data is unique to each subscriber key and channel name.

This method of communication can have many cases:

1. Unicast (1:1) – In this case there is one publisher and one subscriber
2. Multicast (1: Many) – In this case 1 publisher is present but many subscribers
3. Allcast (Many: Many) – In this case many publishers communicate with many subscribers. In this case, a publisher can also subscribe to either himself or another channel.

Since our project lies within the scope of a Unicast, when the laptop publishes a message, the remote ground vehicle receives it and acts accordingly. Whenever a gesture as defined in the python script at the laptop is recorded from the LeapMotion controller, the script publishes a message to a particular channel as shown in the code below:

```
pubnub.publish('test', 'Move Forward', callback=callback, error=callback)
```

Figure 9: Pubnub Message Publishing

The error and callback parameters of the function define the current state of the connection between the publisher and the server. If at any point, the publisher gets disconnected from the server, the program will show a message stating an error in connection. After the program has completed, both the subscriber and publisher need to unsubscribe from the channel else it would lead to a security flaw on both ends of the system.

3.3 Remote Ground Vehicle

The setup of the remote ground vehicle was a lengthy and tedious job and filled with problems and error. For the setup, the image of the stripped down Linux operating system called Raspbian was downloaded and flashed to a MicroSDHC card. The file system on this card is supposed to be ext4 rather than FAT32 or NTFS file systems of a Windows machine. Special software is needed to flash an image properly on the MicroSDHC card. Once, the image was flashed, the MicroSDHC card was inserted into the computing board, and all the necessary cables were connected, but no output could be seen via the HDMI port. To solve this, the config.txt file in the root of the MicroSDHC card was modified to force display using the HDMI port and then the screen resolution was set to 720p. After many sets of tweaks, the display finally showed on the monitor and a successful login was performed. The next step was to install the Pubnub software development kit on the Raspberry Pi for which the procedure for installation was followed from the website of the enterprise but kept ending with an error. When checked for the dependencies i.e. base files of cryptology and encryption, the full structural integrity of the system existed. After much research and stumbling on solutions, a developer version of Python is required on the Raspberry Pi for the Pubnub to work properly. When this was satisfied, Pubnub installed without any hassle and the system was connected. The next step was to mount the Raspberry Pi on the chassis along with the motor controller and the motors. Camera was connected and screwed to its bracket, and the Raspberry Pi was connected to the motor controller and the motors were attached to

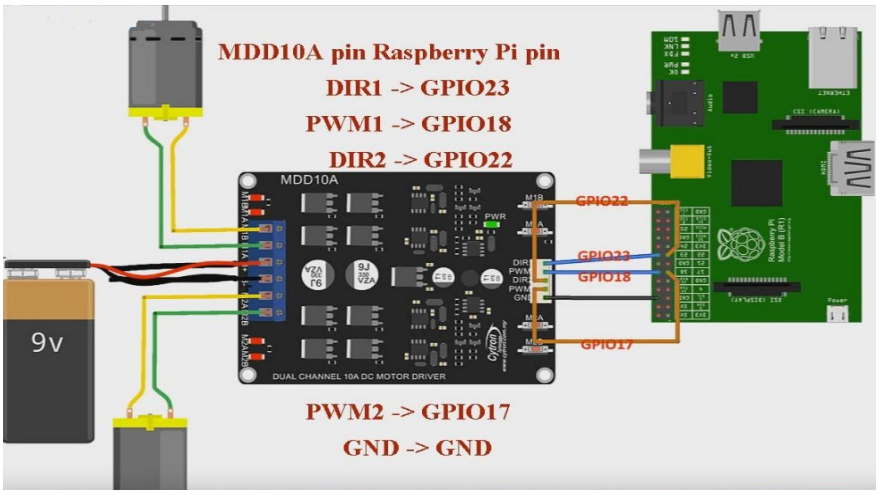


Figure 10: Motor Controller connection [31]

the motor controller according to the following diagram:

Next, the main python script was written which would handle all the call for the motors to run in a particular direction while communication with the Pubnub server in the background. For this task to be accomplished, the GPIO pins on the Raspberry Pi need to be setup and configured to communicate with the motor controller efficiently. This is done after defining the board connection configuration i.e. either PCM or BOARD [31]. The project is using BOARD configuration as it makes it easier to remember which wire is connected to which pin number. The code snippet is shown below:

```
GPIO.setmode(GPIO.BOARD)

dir1=16
mtr1=12
dir2=15
mtr2=11

GPIO.setup(dir1,GPIO.OUT)
GPIO.setup(mtr1,GPIO.OUT)
GPIO.setup(dir2,GPIO.OUT)
GPIO.setup(mtr2,GPIO.OUT)
```

Figure 11: GPIO setup for Raspberry Pi in

The python code is written such that when a message is received from the publisher to move forward, all wheels move forward for 0.5 sec and then stop. This timed approach to movement provides more precision to the current location of the remote ground vehicle while also providing better maneuverability.

3.4 Optimization

After the above-mentioned steps are completed for the development of the code for each module, some gaps remain before the project can be deployed[20]. The problems that can be traced down include:

- Normalizing the gesture recognition in a particular direction specifically for Move Forward and Move Backward.
- Normalizing the input from the LeapMotion controller to prevent overload of commands to motor controller
- Running the Python script on the Raspberry Pi at boot.

3.4.1 Normalizing the gesture recognition

The frames [23] recorded from the LeapMotion controller for the gestures of swipe front and swipe back have a lot of freedom to move in the other two axes namely X and Y axes. This makes the recognition of gesture for just the movement in the Z-direction less relevant. This problem is solved with the help of the following if and elif statements:

```
swdir=swipe.direction
if (swdir.z>0 and math.fabs(swdir.z)>math.fabs(swdir.y) and math.fabs(swdir.z)>math.fabs(swdir.x)):
    bck=(bck+1)
    print "Move Backward"
    if bck==25:
        pubnub.publish('test', 'Move Backward', callback=callback, error=callback)
        bck=0
elif (swdir.z<0 and math.fabs(swdir.z)>math.fabs(swdir.y) and math.fabs(swdir.z)>math.fabs(swdir.x)):
    fwd=(fwd+1)
    print "Move Forward"
    if fwd==25:
        pubnub.publish('test', 'Move Forward', callback=callback, error=callback)
        fwd=0
```

Figure 12: Swipe Gesture Normalization

The swdir.z attribute compares the movement of the motion of hands in the z direction and compares it the motion of the hands the X and Y axes and the control goes inside the if statement if that condition is fulfilled. The same procedure is repeated for each of the swipe directions.

3.4.2 Normalizing the Input from LeapMotion controller

The other problem faced when taking input from the LeapMotion controller is when the frames [23] object of the class is used for recognition of gestures, an instance of successful recognition is passed from the software even when the full gesture has not been performed. A situation which helps explain such a condition is when the user is performing a clockwise circle gesture, and he starts drawing the circle at the uppermost point. Till the time user has completed approximately forty degrees in the clockwise direction, the software recognizes it as a successful full clockwise circle gesture. For this problem, a unique solution was applied such that when the software makes thirty-five successful calls for the circle gesture, it is recorded as a single input and the message is sent to the remote ground vehicle accordingly. The following code snippets help understand the solution applied.

```
if clockwiseness=="clockwise":
    cl=(cl+1)
    if cl==35:
        pubnub.publish('test', 'Turn Right', callback=callback, error=callback)
        cl=0
```

Figure 13:LeapMotion input normalization

The variable cl acts as a counter for the number of occurrences of a successful call for the clockwise circle gesture, and when thirty-five of such calls have been made by the software, it publishes the message to be received by the remote ground vehicle to process the information and thus, turn right.

3.4.3 Running Python script at boot on Pi

When the python script for this project is created, it runs through the graphical user interface of the Raspberry Pi. However, when deploying the project in the field, it is necessary that the python script runs by itself and performs the task required. The solution to this problem is to create a system service in the raspberry pi which gets initiated when the Raspberry Pi boots up and keeps running in the background. The pre-requisite of this script is a connection to the internet as the Raspberry receives messages to process through the web services of Pubnub.

A system daemon is a part of multitasking operating systems which runs in the background and cannot be directly controlled by the user[21]. The service is created in the following manner for deployment.

- Place the script file in the /home/pi directory for easy navigation as this is one of the few directories which get loaded into the memory
- Create a configuration file that informs the system daemon to perform the task using the following command:

  sudo nano /lib/systemd/system/script.service
- Change the permission of the configuration file so that it is read only by the system
- Reload the system daemon and then enable the script using systemctl (controls the daemon) command[32]

Once the script is running in the background, the project can be deployed in the field for testing.

4. Conclusion and Future Work

The completion of the project provides a LeapMotion controller as a platform for the recognition and manipulation of hand gestures which can be applied in various areas as an input to a computer. The other deliverable for the project is a globally remote controlled ground vehicle powered by a Li-Po battery with Raspberry Pi as the computing module. The LeapMotion controller is used as an input for hand gestures and messages the remote vehicle through the Pubnub servers to move accordingly. The successful movement of the remote ground vehicle suggests that the applications of such type of an input device are countless and the functionality of LeapMotion controller can be extended to manipulate the input to other computing devices. The application of hand gestures, thus expands the application of the theory of human-computer interaction. The hand gesture input thus allows us to globally remote control a ground vehicle powered by a computing module, Raspberry Pi.

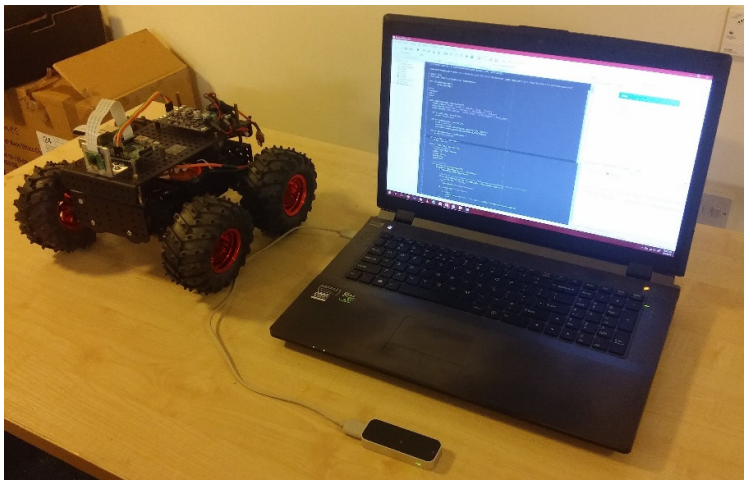


Figure 14: Hand Gesture recognition and remote vehicle

The aims of the project were completed on time making the project planning a successful tool for the completion of the project. The project proved to be very challenging and an exciting experience as it allowed the learning of vast amounts knowledge in topics related to human-computer interfacing and gesture control. The way of communication between the devices was the most exciting part as it allowed the exploration of different protocols of communication and data transmission without compromising on the level of security.

The project covered many aspects for final deployment in the field, but some features were still lacking that could be modified and developed for use in the future. The remote ground vehicle lacked the capability of video streaming through its native camera which would result in a perfect solution for using the robot in various real life cases. Many attempts were made to have a video streaming available for the project but resulted in failed outputs. The video streaming is a huge feature as it is one of the key

components for the robot to be really globally remote controlled. A hand manipulator can be attached to the robot and sent out for mapping or rescue missions in the field. Such modifications can be made to the robot making it a truly adaptable piece of hardware for output.

5. References

[1] A. Bednjanec and M. F. Tretinjak, "Application of Gantt Charts in the Educational Process," pp. 631–635.

[2] X. Liang, Q. Li, X. Zhang, S. Zhang, and W. Geng, "Performance-driven motion choreographing with accelerometers," *Comput. Animat. Virtual Worlds, 20*, vol. 2, no. 3, pp. 89–99, 2009.

[3] T. Tullis, W. Albert, and M. Kaufmann, "Human-Computer Interaction," no. May, pp. 1–4, 2008.

[4] C. Sharma, "and HCI Perspective," pp. 607–612, 2014.

[5] M. Nowicki, O. Pilarczyk, and J. Wąsikowski, "Gesture Recognition Library for Leap Motion Controller," 2014.

[6] R. C. Johnson, "3-D gesture control breaks out of the game box," pp. 1–7, 2015.

[7] M. Alcoverro, X. Suau, J. R. Morros, A. López-Méndez, A. Gil, J. Ruiz-Hidalgo, and J. R. Casas, "Gesture control interface for immersive panoramic displays," *Multimed. Tools Appl.*, pp. 1–27, 2013.

[8] D. Ionescu, V. Suse, C. Gadea, B. Solomon, B. Ionescu, and S. Islam, "A new infrared 3D camera for Gesture Control," *Conf. Rec. - IEEE Instrum. Meas. Technol. Conf.*, pp. 629–634, 2013.

[9] Y. She, Q. Wang, Y. Jia, T. Gu, Q. He, and B. Yang, "A Real-Time Hand Gesture Recognition Approach Based on Motion Features of Feature Points," *2014 IEEE 17th Int. Conf. Comput. Sci. Eng.*, pp. 1096–1102, 2014.

[10] G. Marin, F. Dominio, and P. Zanuttigh, "Hand gesture recognition with leap motion and kinect devices," *2014 IEEE Int. Conf. Image Process.*, pp. 1565–1569, 2014.

[11] S. Kuchinskas, "Naked gaming," *Sci Am*, vol. 302, no. 2, p. 24, 2010.

[12] E. MARKOWITZ, "Wave Your Hand, Control Your Computer.," *Inc.*, vol. 35, no. 5, pp. 112–113, 2013.

[13] C. Providers and J. Forces, "Great Leap Forward in Gesture-Control

Technology," *IEEE Comput. Soc.*, no. July, pp. 21–22, 2012.

[14] W. Cui and D. Li, "Research based on OSI model," pp. 554–557, 2011.

[15]]"Mis experiencias con la Raspberry Pi: Modelos de la RasPi", *Muyraspi.blogspot.co.uk*, 2016. [Online]. Available: http://muyraspi.blogspot.co.uk/2013/02/modelos-de-la-raspi.html. [Accessed: 04- Jan- 2016].

[16] A. Choi, "Senior Project Design Report Spirit : A Home Automation System," pp. 1–21.

[17] C. Bysani and S. Chundi, "Council for Innovative Research," vol. 11, no. 2, pp. 2250–2255.

[18] E. Sarajevo, "Raspberry Pi as a Wireless Sensor Node : Performances and Constraints," no. May, pp. 26–30, 2014.

[19] P. 2, "Plug a 5V fan to GPIO 1 and 2", *Raspberrypi.stackexchange.com*, 2016. [Online]. Available: http://raspberrypi.stackexchange.com/questions/32837/plug-a-5v-fan-to-gpio-1-and-2/33412. [Accessed: 29- Mar- 2016].

[20] Y. Yin and R. Davis, "Real-time continuous gesture recognition for natural human-computer interaction," *2014 IEEE Symp. Vis. Lang. Human-Centric Comput.*, pp. 113–120, 2014.

[21]"Gestures — Leap Motion Python SDK v2.3 documentation", *Developer.leapmotion.com*, 2016. [Online]. Available: https://developer.leapmotion.com/documentation/python/devguide/Leap_Gestures.html#id13. [Accessed: 31- Oct- 2015].

[22] B. Date, C. S. Level, and B. Course, "Degree project Extending Lego Mindstorms NXT Functionality Using Internet- Connected Android Device," 2012.

[23]"Frames — Leap Motion Python SDK v2.3 documentation", *Developer.leapmotion.com*, 2016. [Online]. Available: https://developer.leapmotion.com/documentation/python/devguide/Leap_Frames.html. [Accessed: 29- Oct- 2015].

[24] J. Han and N. Gold, “Lessons Learned in Exploring the Leap Motion(TM) Sensor for Gesture-based Instrument Design,” *Proc. Int. Conf. New Interfaces Music. Expr.*, pp. 371–374, 2014.

[25] J. Hanson, D. Nugent and I. Jennings, "PubNub Support and API Integration Help", *Airpair.com*, 2016. [Online]. Available: https://www.airpair.com/pubnub/posts/pubnub-support-and-api-integration-help. [Accessed: 05- Apr- 2016].

[26]"Python SDK Documentation — Leap Motion Python SDK v2.3 documentation",*Developer.leapmotion.com*, 2016. [Online]. Available: https://developer.leapmotion.com/documentation/python/index.html. [Accessed: 18- Nov- 2015].

[27] D. Management and U. Systemd, “Daemon Management Under Systemd,” no. June, pp. 28–34, 2015.

[28]"System Architecture — Leap Motion Python SDK v2.3 documentation",*Developer.leapmotion.com*, 2016. [Online]. Available: https://developer.leapmotion.com/documentation/python/devguide/Leap_Architecture.html. [Accessed: 29- Nov- 2015].

[29]"CircleGesture — Leap Motion Python SDK v2.3 documentation", *Developer.leapmotion.com*, 2016. [Online]. Available: https://developer.leapmotion.com/documentation/python/api/Leap.CircleGesture.html. [Accessed: 26- Nov- 2015].

[30]"Coordinate Systems — Leap Motion Python SDK v2.3 documentation", *Developer.leapmotion.com*, 2016. [Online]. Available: https://developer.leapmotion.com/documentation/python/devguide/Leap_Coordinate_Mapping.html. [Accessed: 05- Nov- 2015].

[31] J. Barnett, "Controlling DC Motors Using Python With a Raspberry Pi", *Computer Skills Envato Tuts+*, 2014. [Online]. Available: http://computers.tutsplus.com/tutorials/controlling-dc-motors-using-python-with-a-raspberry-pi--cms-20051. [Accessed: 17- Mar- 2016].

[32]"How To Autorun A Python Script On Boot Using systemd", *Raspberrypi-spy.co.uk*, 2015. [Online]. Available: http://www.raspberrypi-spy.co.uk/2015/10/how-to-autorun-a-python-script-on-boot-using-systemd/. [Accessed: 15- Mar- 2016].

6. Appendices

For the purpose of clearer arrangement, the following pages have been aligned in landscape view with narrow margins.

6.1 LeapMotion Code

```
import os, sys, inspect, math
from pubnub import Pubnub


src_dir=os.path.dirname(inspect.getfile(inspect.currentframe()))
arch_dir='../lib/x64' if sys.maxsize>2**32 else '../lib/x86'
sys.path.insert(0, os.path.abspath(os.path.join(src_dir, arch_dir)))


pubnub=Pubnub(publish_key='pub-c-xxxx',subscribe_key='sub-c-xxxx')
#xxxx refer to unique keys which can be acquired for individual users. For safety purposes, these keys
have been marked as xxxx


import Leap
from Leap import CircleGesture, SwipeGesture


def callback(message):
        print(message)


cl=0
councl=0
bck=0
fwd=0
```

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

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

    def on_connect(self, controller):
        print "Connected"
        controller.enable_gesture(Leap.Gesture.TYPE_CIRCLE);
        controller.enable_gesture(Leap.Gesture.TYPE_SWIPE);

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

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

    def on_frame(self, controller):
        # Frame available
        frame=controller.frame()
        global cl
```

```
global councl
global bck
global fwd

# Get gestures
for gesture in frame.gestures():
    if gesture.type==Leap.Gesture.TYPE_CIRCLE:
        circle=CircleGesture(gesture)

        # Determine clock direction using the angle between the pointable and the circle normal
        if circle.pointable.direction.angle_to(circle.normal)<=Leap.PI/2:
            clockwiseness="clockwise"
        else:
            clockwiseness="counterclockwise"

        if clockwiseness=="clockwise":
            cl=(cl+1)
            if cl==35:
                pubnub.publish('test', 'Turn Right', callback=callback, error=callback)
                cl=0
        else:
            councl=councl+1
            if councl==35:
                pubnub.publish('test', 'Turn Left', callback=callback, error=callback)
```

```
                        councl=0


            if gesture.type==Leap.Gesture.TYPE_SWIPE:
                swipe=SwipeGesture(gesture)
                swdir=swipe.direction
                if (swdir.z>0 and math.fabs(swdir.z)>math.fabs(swdir.y) and
math.fabs(swdir.z)>math.fabs(swdir.x)):
                    bck=(bck+1)
                    print "Move Backward"
                    if bck==25:
                        pubnub.publish('test', 'Move Backward', callback=callback, error=callback)
                        bck=0
                elif (swdir.z<0 and math.fabs(swdir.z)>math.fabs(swdir.y) and
math.fabs(swdir.z)>math.fabs(swdir.x)):
                    fwd=(fwd+1)
                    print "Move Forward"
                    if fwd==25:
                        pubnub.publish('test', 'Move Forward', callback=callback, error=callback)
                        fwd=0


        def state_string(self, state):
            if state==Leap.Gesture.STATE_START:
                return "STATE_START"
            if state==Leap.Gesture.STATE_UPDATE:
                return "STATE_UPDATE"
```

```
                if state==Leap.Gesture.STATE_STOP:
                    return "STATE_STOP"
                if state==Leap.Gesture.STATE_INVALID:
                    return "STATE_INVALID"

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
        pubnub.publish('test', 'Exit', callback=callback, error=callback)
        controller.remove_listener(listener)
        pubnub.unsubscribe(channel="test")
```

```
  if __name__=="__main__":
      main()
```

6.2 Raspberry Pi Python Script

```
import sys,os
from time import sleep
import RPi.GPIO as GPIO
from pubnub import Pubnub

pubnub=Pubnub(publish_key='pub-c-xxxx',subscribe_key='sub-c-xxxx',ssl_on=False)
#The keys here should match the keys from the LeapMotion controller Python code

GPIO.setmode(GPIO.BOARD)

dir1=16
mtr1=12
dir2=15
mtr2=11

GPIO.setup(dir1,GPIO.OUT)
GPIO.setup(mtr1,GPIO.OUT)
GPIO.setup(dir2,GPIO.OUT)
GPIO.setup(mtr2,GPIO.OUT)
```

```
def _callback(message, channel):
    print (message +" - Pi")
    if message=="Move Backward":
        print "Backward"
        GPIO.output(dir1,1)
        GPIO.output(dir2,1)
        GPIO.output(mtr2,1)
        GPIO.output(mtr1,1)
        sleep(0.5)
        GPIO.output(dir1,1)
        GPIO.output(dir2,1)
        GPIO.output(mtr2,0)
        GPIO.output(mtr1,0)
    elif message=="Move Forward":
        print "Forward"
        GPIO.output(dir1,0)
        GPIO.output(dir2,0)
        GPIO.output(mtr2,1)
        GPIO.output(mtr1,1)
        sleep(0.5)
        GPIO.output(dir1,0)
        GPIO.output(dir2,0)
        GPIO.output(mtr2,0)
        GPIO.output(mtr1,0)
```

```
elif message=="Turn Right":
    print "Turn Right"
    GPIO.output(dir1,0)
    GPIO.output(dir2,1)
    GPIO.output(mtr2,1)
    GPIO.output(mtr1,1)
    sleep(0.5)
    GPIO.output(dir1,0)
    GPIO.output(dir2,0)
    GPIO.output(mtr2,0)
    GPIO.output(mtr1,0)
elif message=="Turn Left":
    print "Turn Left"
    GPIO.output(dir1,1)
    GPIO.output(dir2,0)
    GPIO.output(mtr2,1)
    GPIO.output(mtr1,1)
    sleep(0.5)
    GPIO.output(dir1,0)
    GPIO.output(dir2,0)
    GPIO.output(mtr2,0)
    GPIO.output(mtr1,0)
elif message=="Exit":
    GPIO.cleanup()
```

```
        pubnub.unsubscribe(channel="test")
        os._exit(1)

def _error(message):
    print (message)

def _reconnect(message):
    print (message)

pubnub.subscribe(channels="test", callback=_callback, error=_error, reconnect=_reconnect)
try:
    while 1:
        pass
except KeyboardInterrupt:
    GPIO.cleanup()
    pubnub.unsubscribe(channel="test")
```

6.3 Google Drive Link

Further information including log book, articles and demonstration videos can be found in the following Google Drive folder:

https://drive.google.com/open?id=0B6OpccbHcILoam5jYXFwbHNDVXM