Summer 6-15-2019

RaspBot - Raspberry Pi-powered Robot controlled by a WebApp

Julio Lopez
CCT College Dublin

Eric Michel
CCT College Dublin

Jhon Loiaza
CCT College Dublin

Nancy Aguilera
CCT College Dublin

Vilmarys Salgado
CCT College Dublin

RaspBot

Raspberry Pi Powered Robot Controlled by a Web App

Final Year Project

May 2019

BSc in Information Technology

Julio Lopez - 2016113

Eric Michel - 2016426

Jhon Loaiza - 2016225

Nancy Aguilera - 2016102

Vilmarys Salgado - 2016236

Supervisor: Amilcar Aponte

"The Secret of my happiness lies in not striving for pleasure, but in finding pleasure in the effort".

Andre Gide

Declaration

We hereby declare that the work described in this dissertation is, except where otherwise stated, entirely our own work and has not been submitted as an exercise for a degree at this or any other university.

_____

Student Names

2019

Acknowledgement

Firstly, we would like to thank our supervisor, Amilcar Aponte, for being our tutor through the full development of this project, and for taking out time of his busy schedule to help explain certain technologies and concepts that were applied in our experiments.

We would also like to thank our CCT lecturer, Graham Granville, for his guidance and advice that he has provided us throughout this process.

And the lastly, we would also like to thank the college, CCT, our home of study, for providing us a memorable education experience in Information Technology, which empowered us to carry out our final project with the professionalism it deserves.

Abstract

This project aims at designing a robot prototype made from Lego, which contains a Raspberry Pi device, that will be remotely controlled through a web application to allow the robot to perform certain movements which would be sent from the client's interaction. The process began by researching several different technologies that we used as our starting point for designing both prototypes (hardware and software). Concurrently, we focused investigation on the IoT and were inspired to make a robot prototype that would implement the basic concepts and, at the same time, merge it with the previously learnt knowledge acquired during our 3 year course in Information Technology.

Table of Contents

Table of Figures

Table of Appendices

List of abbreviations

Abbreviated Form	Meaning
RPi	Raspberry Pi
GPIO	General Purpose Input Output
IoT	Internet of Things
LED	Light Emitting Diode
DOM	Document Object Model
UD	User Interface
HTML	Hypertext Mark-up Language
CSS	Cascading Style Sheets
UI	User Interface
W3C	World Wide Web Consortium
DNS	Domain Name Service
IP	Internet Protocol
RDP	Remote Desktop Protocol
RDS	Remote Desktop Service
RPM	Revolution Per Minute

Introduction

This project is composed by 8 chapters which are structured as follows:

Chapter 1: Proposal

This Chapter will cover the initial proposal of the project. This includes what the core objectives are and the justification of the project as to why Caffeine chose to do this project. It will also outline the basic goals and objectives that were set out initially for the project to be accomplished.

Chapter 2: Project Planning

This chapter introduces Caffeine and the group members as well as their initial planning of the project. It will display the activity plan which outlines which goals and steps were needed to be achieved and what timeframe was given to each task. It will also describe the main challenges and risks towards completing the project that the group faced during the development phases.

Chapter 3: Literature Review (Background Concepts)

This chapter explains the more abstract concepts and methodologies that this project incorporates. In a high level view. the goals and objectives that have been set out in the project are based through these concepts.

Chapter 4: Literature review (Technology)

This chapter introduces the different technologies that are used in this project. Each piece of technology is regard as a component during the explanations of what they are and what their purpose is. This chapter is designed to give the reader a basic understanding of each component and how it fits in the full picture.

Chapter 5: Design (mechanical and software)

This chapter explains the design of the project in 2 separate parts, the hardware and the software. It will give the reader an understanding of Caffeine's design choices and incorporation

of each technology used. It will also explain to the user how they could design and implement their own prototype from the examples from this project and other components that is used.

Chapter 6: Design and Implementation

This chapter will go through all the different design ideas and experiments that Caffeine undertook. It outlines each relevant experiment and their findings to better help the reader understand what kind of process Caffeine went through when trying to reach its final goals.

Chapter 7: Results

This chapter will explain what objectives of the project were met or weren't met as well as list out what was a success or not. It will outline whatever failings Caffeine came across and why a specific component was chosen over another one.

Chapter 8: Future development recommendations

This chapter looks back at the process of this project and explains what can be done differently or if any other components can be added to increase the features of the project. This section is dedicated to those that see an opportunity continuing with this project and want to go forward.

1 The Proposal

This Chapter will cover the initial proposal of the project. This includes what the core objectives are and the justification of the project as to why Caffeine chose to do this project. It will also outline the basic goals and objectives that were set out initially for the project to be accomplished.

1.1 Core Objective

The main focus of this project is to create a functional robot which is controlled through a web application. The web application will host a user interface that can control the robot's states, whether it would be moving, turning or stopped. It would also be able to adjust the speeds at which these actions can be done. The user would be able to access the UI from any location in the Republic of Ireland as long as RaspBot is connected to a mobile network service provider, in this case: Three Mobile, as they have ideal coverage around the country at a cheap price. The RaspBot's functionality is broken off into two different parts which can be considered its main goals for success.

- Basic Movement: This includes moving forwards, backwards, turning left and right and stopping.
- Availability: Having RaspBot's basic movement be controlled through an interface over a local network or over the internet by being connected to a mobile service provider.

Thanks to the Pi's extensive capabilities, future implementations and upgrades can be made. Upgrades include adding a camera that feeds into the UI as well as having sensors added to RaspBot that would enable it to sense objects around it. There are many different aspects of this project that need to be considered which include the purpose and justification of this project, what technologies will be required and how the group will set out to achieve its core objective.

The project itself will explore the concept of the Internet of Things (IoT) as well as basic robotics through circuitry and electronics and fuse them both. These concepts strengthen the reasoning for the core objective of controlling RaspBot from anywhere at any time. It is also aiming to broadcast this concept (controlling devices with a touch of a button from any location) to others who wish to learn it.

1.2 Justification

With the aim of completing the project, it will be possible to cover many other different interesting topics which include:

- Taking what we have learnt from our educational train and putting it in a practical use.
- Learning about and unifying different technologies to achieve the core objective of this project.
- Create a valid and disseminative document about the application of the electronic makeup, construction of the robot and the connection between user and the bot.

One of the most important benefits that will be acquired from the completion of this project is having the chance to explore the capabilities of the Raspberry Pi computing system which will allow us to clearly distinguish its potential. The Raspberry Pi is a low-cost, miniaturized computer that blossomed from the idea of improving the educational sector in a very short time in a cost-effective manner. It has become a primary exponent of cheap hardware and is the core foundation of a significant number of projects around the globe.

The project will also include, with what is known today as, the Internet of Things (IoT). IoT systems consists of different devices (which have sensors, functions, storage/ database) that "talk" between each other using a Wide Area Network (WAN) which is more commonly known as the ‘Cloud’.

Simultaneously, this project aims to be a fulfilling document for future students and thinkers who are interested and wish to explore the concept of the IoT, the Raspberry Pi, electronics or other technologies related to it.

1.3 Project Goals & Objectives

- Learn the functionality of and how the different electrical components required in this project work.
- Learn about the RPI Operating System ‘Raspbian’ and coding language Python.
- Create the required code and apply it to the RaspBot prototype that will allow it the mobility to move forward, backwards and turn in any direction.
- Create the Web Application to control RaspBot remotely.
- Apply a 4G connecting device to RaspBot to provide it a connection to the Internet.
- Assemble the required hardware components including the RPI, motors, wheels, chassis of RaspBot, internet connection device, etc. and apply a designed plan that will outline the construction of these materials to assemble RaspBot.

Chapter 2

2 Project Planning

This chapter introduces Caffeine and the group members as well as their initial planning of the project. It will display the activity plan which outlines which goals and steps were needed to be achieved and what timeframe was given to each task. It will also describe the main challenges and risks towards completing the project that the group faced during the development phases.

2.1 The team Members

Caffeine was founded in the 2nd year course when the final year project was first proposed to us. We first discussed the different ideas on what we could do and how to do it. Once we had figured out which idea was best suited for us, we gave birth to our first initial concept: 'A Raspberry Pi Powered Robot Controlled by a Web App'. Jhon and Eric were the two initial members of Caffeine who came up with the idea of working on a robot prototype that can be remotely controlled through a Raspberry P. However, this concept was first introduced to all of us from our lecturer Kyle Gossling, who told us about his experience using a Raspberry Pi for a personal project of his where he created a heating system in his home that was remotely controlled by his smart phone. From this moment, many of us were very curious about the RPi and took it upon ourselves to search for more information about it. We officially became a group and named ourselves "Caffeine", in honor to the many coffees we drank daily while studying at CCT. Once we finished our brainstorming, set ourselves goals and objectives we then assigned tasks to each member, giving them a specific role to be able to successfully complete the project in an efficient and timely manner.

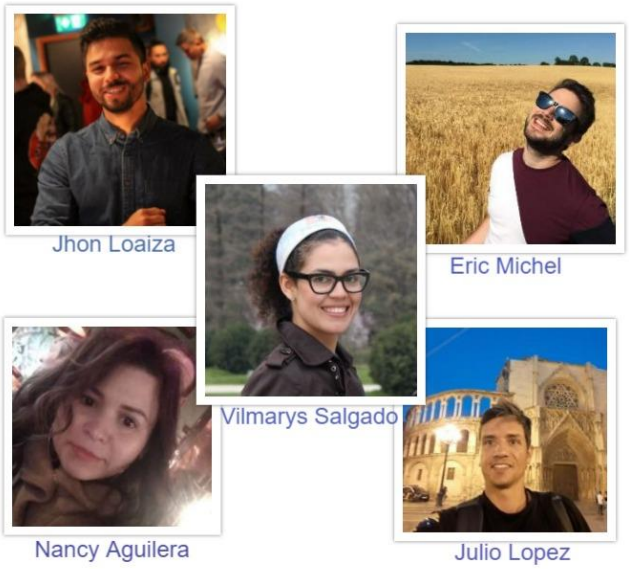


Figure 1: Caffeine's Members.

2.2 The Planning

Planning was the first step to be completed before we started creating our Prototypes for the UI and the RaspBot's body. We began by doing an analytical brainstorming session between all the team members to distinguish the different components and technologies that will be needed to build the UI and body. We defined what would be our main core objectives, functionalities, design, cost, organization, roles, meeting agendas and timeline to develop our project. We then proceeded to buy the hardware components needed to start our testing phase where we would analyze the results in order to plan our design. After the testing was concluded we carried out the first experiments between the RPi, DC motors and python scripts. You can refer to the Experiments in Chapters 6 to understand further. Throughout the whole process we were setting weekly targets for Caffeine to achieve, with the help and supervision of our tutor, Amilcar Aponte. Mr. Aponte provided reliable and friendly support during the development of the most crucial phases of this project, which can be summarized as follow:

- **Research**: All members researched projects, articles, threads, websites and even talked to other professionals in the field related to this project. This was done in such a way that the whole team was involved and contributed to the development of the project as a whole by gathering information and provide an understanding of how to implement or combine the technologies needed to materialize the prototype.
- **Prototypes Design**: This phase was divided into two parts (logical and physical). The logical design outlined architecture, coding languages, libraries, frameworks, etc. that was required to connect the physical prototype with the web application to achieve control of the movements of the RaspBot prototype. This involved developing backend and frontend code. The design of the hardware was more related to the combination of all the physical components (such as the DC motors, breadboard, RPi) components in a way that they could be efficiently structured and form a wanted shape that would best suit the movements.
- **Testing the Prototypes**: This phase was also divided into two parts (logical and physical) with which the logical testing being fundamental throughout the development process. As this project was our first encounter with the RPi, we made testing a top priority from the beginning to try and fully understand how it functions with the connected motors, breadboard, power supply and to test its connectivity as a whole system. After, we were able to check the speed and the movements (forward / backward) of the motors without having the RaspBot assembled.
- **Initial prototypes**: Once all the testing of the components is completed, we could merge the 2 different parts into one and create a prototype RaspBot. There would be little emphasis on the front end of the application as there would be further need for testing still. The main focus would be to get the interaction and communication channels set up between the user and the RaspBot. After the communication is set up, the different movements can be properly implemented to allow the user more control over the RaspBot.
- **Final Prototypes:** This phase of the development will produce the result of the combined components of the hardware and the software (backend and frontend). With the base of

the robot firmly set and the different electrical parts installed in the correct positions, the RaspBot is could now be fully functional and the user be able to interact with it in a responsive and efficient manner. As proof of concept we will recorded a screencast to navigate a user through our web application as well as show them an example of how the RaspBot is able to move.

- **Documentation**: In this phase we set a first template design to contain the entire documentation to them materialized there all our researching, experiments and findings, combined with the results.

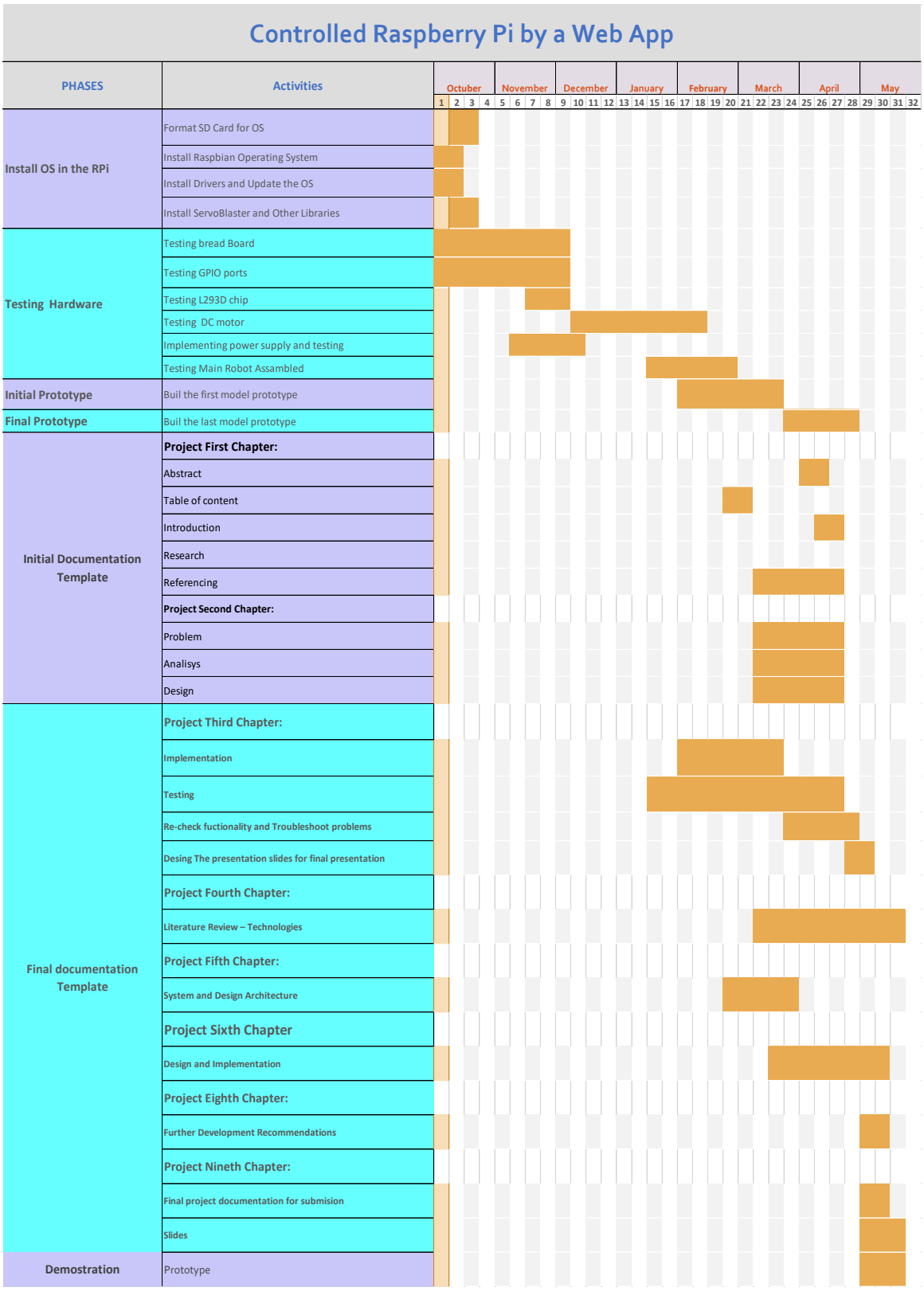


Figure 2: Project Activity Plan

2.3 The Risk Analysis and Main Challenges

Before and after we started building our robot prototype and the web application, we were able to identify multiple risks that could have hindered us in achieving our goals and objectives. In every process there is always something that could go wrong or there will be something that does not work in the way it is meant to be, and so for this reason we highlighted some of the risk that could be expected in the development process of this project. Some of these problems that we identified are:

- the lack of time to finish or improved the prototype.
- procurement of wrong materials.
- defected parts.
- wrong implementations.
- human error (wrong decisions or incidents).
- Disagreements between group members.

There were many challenges faced in the entire process of this project. When we were assembling the robot while we were developing the web application, there were many tests done with extensive troubleshooting. Sometimes motors were not running at all because of the wrong use of code, other times because they were not receiving enough electricity from the power supply or even because it was not assembled correctly with the other components of the RaspBot. Other problems included us not having a full understanding of the L293D H-Bridge Chip (please refer to Chapter 4.1.7) and the wrong installation of it. We experienced a number of different issues which kept us busy by trying to find a solution to fix it or improve our design. Each issue brought us closer in achieving our objectives as it helped us learn and enabled us to shape this project to the form that is in now.

Chapter 3

3 Literature Review: Background

This chapter explains the more abstract concepts and methodologies that this project incorporates. In a high level view. the goals and objectives that have been set out in the project are based through these concepts.

3.1 The Internet of Things (IoT)

The term IoT, which was first proposed by Kevin Ashton, a British technologist, in 1999 has the potential to impact everything from a new product opportunity to a shop floor optimization and even factory worker efficiency gains (power top-line and bottom-line gains). As Internet of Things has overcome the expectations of any statistics that were researched in the past years, is has become the common or the “must have” of almost any new device that is created. A simply definition for this concept will be that is the capability of objects (“things”), created with computational features to connect to the internet or to a network.

Morgan, J. (2014). “*IoT allows for virtually endless opportunities and connections to take place, many of which we can't even think of or fully understand the impact of today*”.

McClelland, C. (2019) *also define the IoT as “it means taking all the things in the world and connecting them to the internet*”.

3.2 The Principle of Least Effort

The principle of least effort is the theory that the "one single primary principle" in any human action, including verbal communication, is the expenditure of the least amount of effort to accomplish a task. Also known as Zipf's Law, Zipf's Principle of Least Effort, and the path of least resistance.

The principle of least effort (PLE) was proposed in 1949 by Harvard linguist George Kingsley Zipf in Human Behavior and the Principle of Least Effort .

This Principle says that if something can be done in different ways, the one that uses the least energy is always better. The process that apply the least amount of energy when performing a task and get the same results is consider to be more efficient. Animals and humans apply this behavior on most scenarios.

Humans tend to follow this behavior in order to safe resources, energy and time. Based on this way of behaving, computers were created implementing the same features. First computers were created to perform simple tasks involving mathematics, design patterns and processing of data. Nowadays computers are able to learn and apply this principle, performing tasks while spending the least amount of energy and finding the affordable way to achieve their assignments.

3.3 Artificial Intelligence

> *Artificial intelligence (AI) is the simulation of human intelligence processes by machines, especially computer systems. These processes include learning (the acquisition of information and rules for using the information), reasoning (using rules to reach approximate or definite conclusions) and self-correction. (Rouse, 2018).*

The perception of artificial intelligence has changed drastically over the past few years where before the common concept of AI was that a robot could think, respond, emotionally react and be creative much like a human being could. However, modern robotics and AI aren't as close to that perception as we would like to believe. Although AI hasn't reach that level yet, it has definitely progressed a lot in terms of limitations. For example, much of today's machines can replicate some specific elements of intellectual ability into other machines. Much like an AI system explaining to three separate air drones, with a net attached between each drone, to catch a ball that's falling in the air.

3.4 Mechanical Design Concepts

3.4.1 Moment

Chegg.com. (n.d.). "It refers to the propensity of the force to cause rotation in a body about any fixed point. The moment's magnitude can be obtained by multiplying force's magnitude with the perpendicular distance at which the force acts".

3.4.2 Flexion

Flexion is the mechanical deformation that can be produced on an element as result of a force applied over it when that force is higher that the resistance of the element based on its physical properties. Another simple and easy definition stated by Oliver Jones says (Jones, 2019) "*flexion refers to a movement that decreases the angle between two body parts*".

We can explain this deformation based on the moment definition that says moment is equal to the product of the force (weight) and the moment arm (distance from a supporting point to the force application point), M = F * d. The moment will generate an effort that try to modify the original condition of the element over which is the force applied. If the value of the moment is higher than the resistance of the material, it will flex until start initially deforming and finally breaking if the force is too high. On our case the moment was just enough to generate a slight deformation. A graphical explanation is given on the figure below. Also this weight affect the force required for the motors to move the RaspBot.

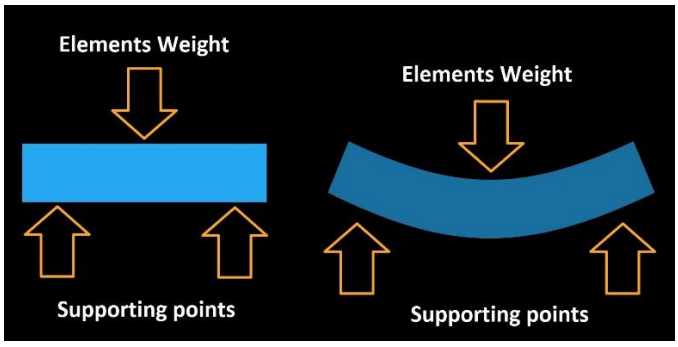


Figure 3: Graphical explanation of moment and its effects. On the left, Material is enough resistant. On the right, Material is not too strong for the moment applied.

Chapter 4

4 Literature Review: Technologies

This chapter introduces the different technologies that are used in this project. Each piece of technology is regard as a component during the explanations of what they are and what their purpose is. This chapter is designed to give the reader a basic understanding of each component and how it fits in the full picture.

4.1 Hardware

4.1.1 Raspberry Pi 3 Model B

First introduced in 2012, this small credit card sized computer runs on Linux operating system (Raspbian) and is powerful enough to be used either as a desktop PC or as a base for building smart devices. It has a compact design and versatile characteristics as well as a low cost, making it an excellent device for experiments.

Buckley, I. (2018) "The Raspberry Pi is the perfect computer for learning. The Linux-based Raspbian OS has Python built in, which makes it a great first system for beginner coders. Its General Purpose Input / Output (GPIO) pins make it easy for budding makers to experiment with DIY electronics projects. It's especially easy when you use code libraries that control these pins, and the popular RPi.GPIO Python library is an excellent example of such a library. The GPIO (General Purpose Input and Output) pins, an integrated technology that enables the RPI to communicate with other electrical components and circuits (such as the LEDs and relays used for this project in the experiment number 1 in chapter 6). The Raspberry Pi 3 (model B) is the latest product in the Raspberry Pi 3 range, boasting a 64-bit quad core processor running at 1.4GHz, dual-band 2.4GHz and 5GHz wireless LAN, Bluetooth 4.2/BLE, Fast Ethernet, and Power over Ethernet (PoE) capability. The dual-band wireless LAN comes with modular compliance

certification, allowing the board to be designed into end products with significantly reduced wireless LAN compliance testing, improving both cost and time to market.

When searching on the Internet, there are countless real-world practical applications using the RPi for implementing IoT devices such as enabling a smart home. RPIs also offers support for a large number of input and output peripherals, including a Network Interfaces Card (NIC) which will allow the RPI access for remote communication which is a key requirement for this project.

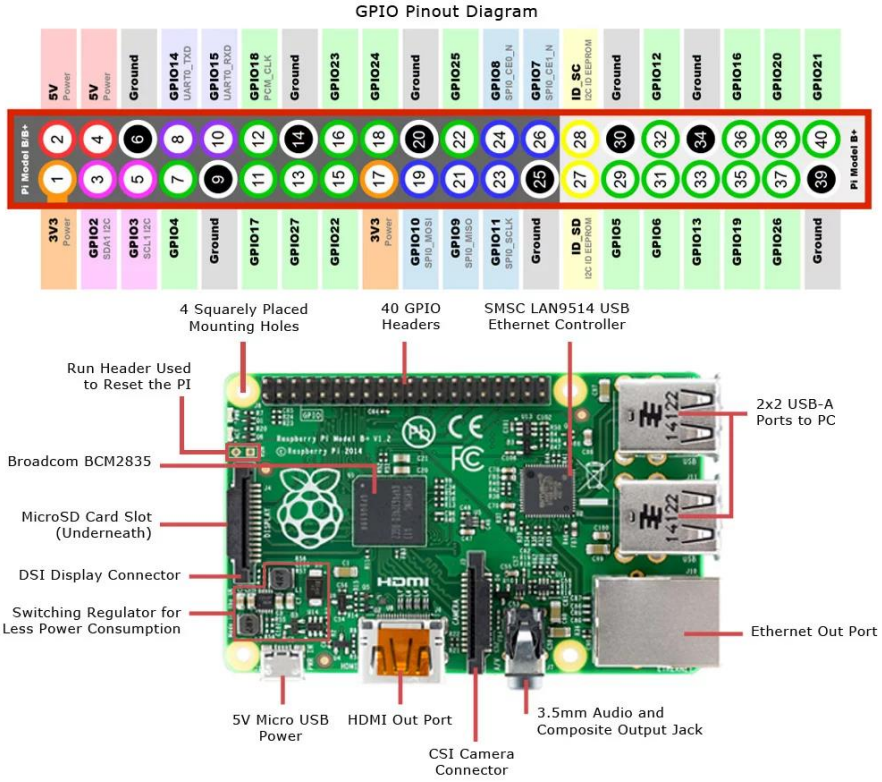


Figure 4: GPIO pinout Diagram (about). Raspberry Pi design architecture (Below).

4.1.2 GPIO Cables

The GPIO cable is a wire or group of wires used to control or manage elements (motors, sensors, resistance, led and others) that are connected to the RPI through electrical or electronic signals and interact with them. The cable or set of cables can be directly connected to the GPIO pins of the Raspberry Pi or to the breadboard for testing.

The GPIO cables can be found in two presentations:

• Ribbons

• Single Cables

A group or set of cables that are perfect for a connection add-on PCB boards to a Raspberry Pi. Usually there are two versions: 26 pins or 40 pins. This project will require these cables to transfer the Raspberry Pi pins to the breadboard which will allow testing the connections required during the experimental stage more comfortable.

Figure 5: Ribbons

There are individual cables used to connect the Raspberry Pi GPIO pins or bread board with the electronic and electrical elements that will be controlled. There are three different types of single GPIO cables:

- Male to Male
- Male to Female
- Female to Female

Male to Male Jumper Cables

Female to Male Jumper Cables

Female to Female Jumper Cables

Figure 6: Single Cables

4.1.3 Breadboard

A breadboard is a solderless device for temporary prototyping for electronics and test circuit designs. Most electronic components in electronic circuits can be interconnected by inserting their leads or terminals into the holes and then making connections through wires where appropriate. The top and bottom rows (usually red and blue rows) are usually the (+) and (-) power supply holes and these move horizontally across the breadboard, while the holes for the components move vertically Each hole is connected to the many metal strips that are running underneath that allows these vertical and horizontal (+) and (-) holes to share an electrical flow respectively. Each wire forms a node and a node is a point in a circuit where two components are connected. Connections between different components are formed by putting their legs in a common node. On the breadboard, a node is the row of holes that are connected by the strip of metal underneath. The breadboard is used as an extension for the GPIO ports from the RPI allowing it to connect and control more than one device through those ports. As a testing tool, it is very functional for the early stages of the project as it can be tested with all connections before permanently soldering components together.

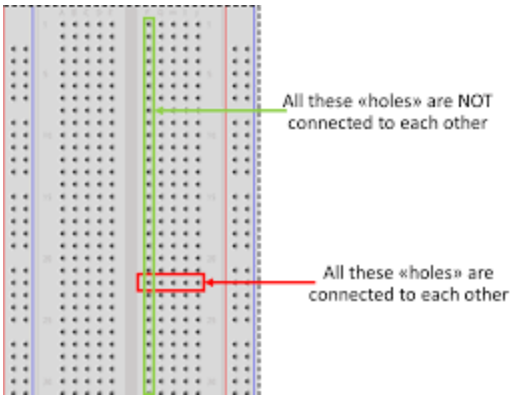


Figure 7:Breadboar Representation

4.1.4 LEGO

LEGO is generally used as a child's development toy to help develop their creativity and immigration. It is also a useful component in DIY projects as it offers versatility when constructing placeholders or objects. It has colorful pieces that are attractive to the eye as well as it being easy to make adjustments. LEGO proved to be a great component to use as it provided an easy way to make changes to the base of the RaspBot as the project developed further and evolved.

LEGO proved to be a great component to use as it provided an easy way to make changes to the base of the RaspBot as the project developed further and evolved.

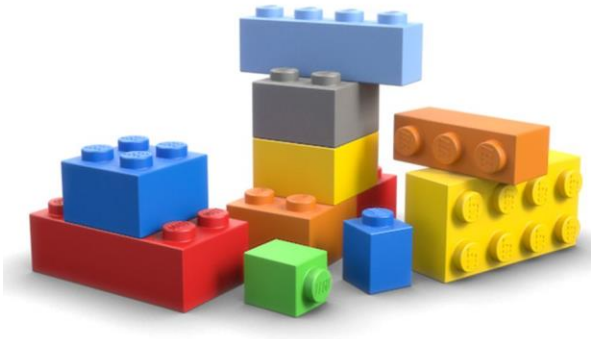


Figure 8: Legos Pieces Diagram

4.1.5 Power Bank

A power bank can be defined as a portable battery charger that has built in circuitry to control any flow of electrical power in and out of its cell. It can charge up its cells by using a USB cable that is connected to a USB port in either a wall socket or other electrical device (such as a PC). If the battery bank has sufficient charge in its cells it can then be used to charge other battery powered items like mobile phones and a host of other devices that would normally use a USB charger. For the purpose of this project, a lithium-ion battery power bank is needed to power the RPI (allowing the device to run independently from a wall outlet) and the DC motors. High capacity power banks allow for a longer supply of energy, specifically for the requirements of this project, a capacity of 10000mAh will be used for RaspBot. Power banks work perfectly with RPIs as the voltage they provide (5 Volts) is enough for the RPI to run at full capacity with a continuous supply of 2.1A. This, of course, is possible due to the internal voltage regulator and current converter that is built into the device.

4.1.6 GPIO Pins

A powerful feature of the Raspberry Pi is the row of GPIO (general-purpose input and output) pins along the top edge of the board. A 40-pin GPIO header is found on all current Raspberry Pi boards (unpopulated on Pi Zero and Pi Zero W). Prior to the Pi 1 Model B+ (2014), boards comprised a shorter 26-pin header. Any of the GPIO pins can be designated (in software) as an input or output pin and used for a wide range of purpose. It is important to define which pin numbering system will be set when creating code to control the GPIO pins. There are two ways of numbering the IO pins on a Raspberry Pi within the RPi.GPIO library. The first is using the BOARD numbering system. This refers to the pin numbers on the P1 header of the RPI board. The advantage of using this numbering system is that the hardware will always work, regardless of the board revision of the RPI. There will not be the need to rewire the connector or change the code. The second numbering system is the BCM (Broadcom SOC Channel) numbers. This is a lower level way of working - it refers to the channel numbers on the Broadcom SOC. You have to always work with a diagram of which channel number goes to which pin on the RPI board. The

challenging aspect of using BCM is that a script could break between revisions of Raspberry Pi boards.

It requires the same logic to set a channel (pin) that will be used as an input (IN) or an output (OUT) on the RPI. There are two different states for the Input and Output which are 0 and 1 i.e. 0 = GPIO.LOW (False) and 1 = GPIO.HIGH (True). For the testing purpose of this project the BOARD numbering system is used. It is most likely that the project will remain using the numbering system through the development of RaspBot.

4.1.7 L293D H-Bridge Chip

This will allow the RPI to control the two DC motors bi-directionally which is the basic movement of the robot. A great feature about L293D is that it comes with built in kick-back protection which prevents your Raspberry Pi from damage the circuit inside the chip or before the chip. The purpose of using L293D it allows to has controlled the prototype setting the GPIO outputs to an L293D motor controller IC and run two separate DC motors in two conditions, the direction of rotation is different for example to moves to left or right at any speed.

1. Enables and disables the motor whether it is on (HIGH) or off (LOW).
2. Input pin for 1st motor (input is either HIGH or LOW).
3. Output for 1st motor (can be either + or -).
4. Ground.
5. Ground.
6. Output for 1st motor (can be either + or -).
7. Input pin for 1st motor (input is either HIGH or LOW).
8. Actual voltage that needs to be carried.
9. Enables and disables the 2nd motor on (HIGH) or off (LOW).
10. Input pin for 2nd motor (input is either
HIGH or LOW).
11. Output for 2nd motor (can be either + or -).
12. Ground.
13. Ground.

14. Output for 2nd motor (can be either + or -).
15. Input pin for 2nd motor (input is either HIGH or LOW).
16. Logic voltage supply 5v

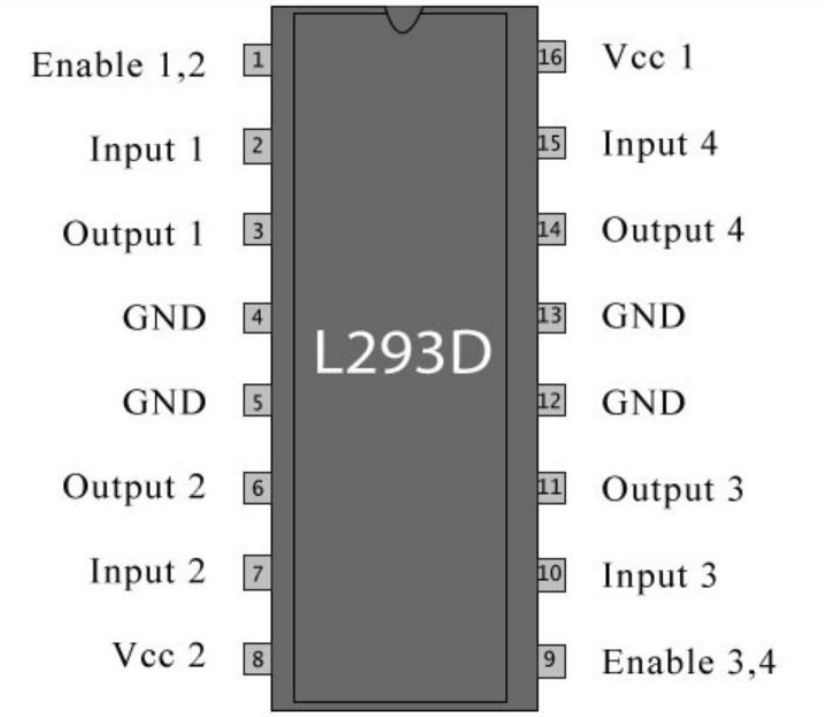


Figure 9: PIN Diagram

Pin No	Function	Name
1	Enable pin for Motor 1; active high	Enable 1,2
2	Input 1 for Motor 1	Input 1
3	Output 1 for Motor 1	Output 1
4	Ground (0V)	Ground
5	Ground (0V)	Ground
6	Output 2 for Motor 1	Output 2
7	Input 2 for Motor 1	Input 2
8	Supply voltage for Motors; 9-12V (up to 36V)	Vcc_2
9	Enable pin for Motor 2; active high	Enable 3,4
10	Input 1 for Motor 1	Input 3
11	Output 1 for Motor 1	Output 3
12	Ground (0V)	Ground
13	Ground (0V)	Ground
14	Output 2 for Motor 1	Output 4
15	Input2 for Motor 1	Input 4
16	Supply voltage; 5V (up to 36V)	Vcc_1

Figure 10: PIN Description

4.1.8 DC Motors

DC motors are motion components that take electrical power in the form of direct current (or some manipulated form of direct current) and converts it into mechanical rotation. Internally the motor passes on the energy through copper cables wired in series to allow the energy to flow from the north to the south of the field coil, generating a magnetic field that turns the armature. For this project, two DC motors (model MM18 medium torque 3V) will be used to move RaspBot forward and backward. This specific device operates between a voltage of 1.5V to 4.5V and can turn at a typical speed of 5000 RPM (Revolutions Per Minute). To supply the energy that is required to run the motors, they are connected to a high capacity power bank that will also ensure continuous energy for hours. During the early stages of the project, these are connected through two GPIO cables each (positive and negative) attached with crocodile clips (small metal clips) to a breadboard (which is connected to the RPI and the power bank). In the final stages of the project the DC motors will be directly connected and soldered to the external energy supply and the RPI with the cables (set at the right length) to allow a compact management of components when it is being assembled.

4.1.9 Angle Geared Hobby Motors

A gear motor is a type of gear that reducer stem from an AC or DC electrical motor. The gear and the motor are integrated in one element. Gear motors are designs include right angle gear motors ranging in wide diversity sizes. The motors speeds are around the 90 RPM (approx.). These motors so match perfectly with the wheels they were shipped with, leading us more towards being able to build our prototype.

These motors offer a number of performance benefits such as certain combinations being efficient and cost-effective than others extremely high torque transmission, low deflection with large applied loads, low backlash, quiet operation.

4.1.10 Light-emitting Diode (LED)

Light-emitting diode (2019)" Is a two-lead semiconductor light source. It is a p–n junction diode, which emits light when activated. When a suitable voltage is applied to the leads, electrons can recombine with electron holes within the device, releasing energy in the form of photons."

It is important to notice that LED has a specific positioning (due to its polarization) to allow the current to pass through it, even you could notice it has a leg shorter than other, that shorter leg represents the cathode (the negative pole) while the longer leg represents the anode (positive pole), for a better understanding see the figure below.

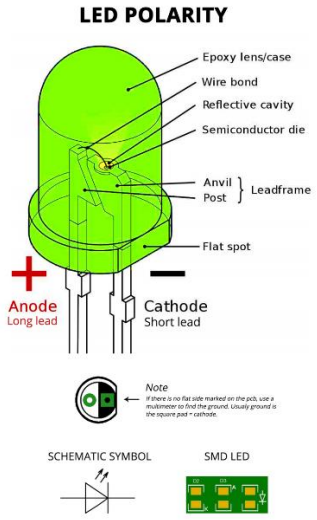


Figure 11: LED description

4.1.11 Resistor

Resistor. (2019, May 07) "*A resistor is a passive two-terminal electrical component that implements electrical resistance as a circuit element to maintain a constant relation between current flow and voltage. Resistors act to reduce current flow, and, at the same time, act to lower voltage levels within circuits. In electronic circuits, resistors are used to limit current flow, to adjust signal levels, bias active elements, and terminate transmission lines among other uses.*"

4.2 Software

4.2.1 Raspbian Operating System

Raspbian is a free Operating System based on Debian (Linux platform). It has been optimized for Raspberry Pi hardware and contains a basic set of programs and utilities to help the Raspberry Pi function. According to the Raspbian website, it hosts over 35 000 packages to provide best performance of the Pi and even though this optimization was completed in 2012, it is still under active development. Just like any other Linux system, Raspbian has those features extended to it meaning that whatever packages or commands that are run in a different Linux machine, chances are good that it would work in the Raspberry Pi.

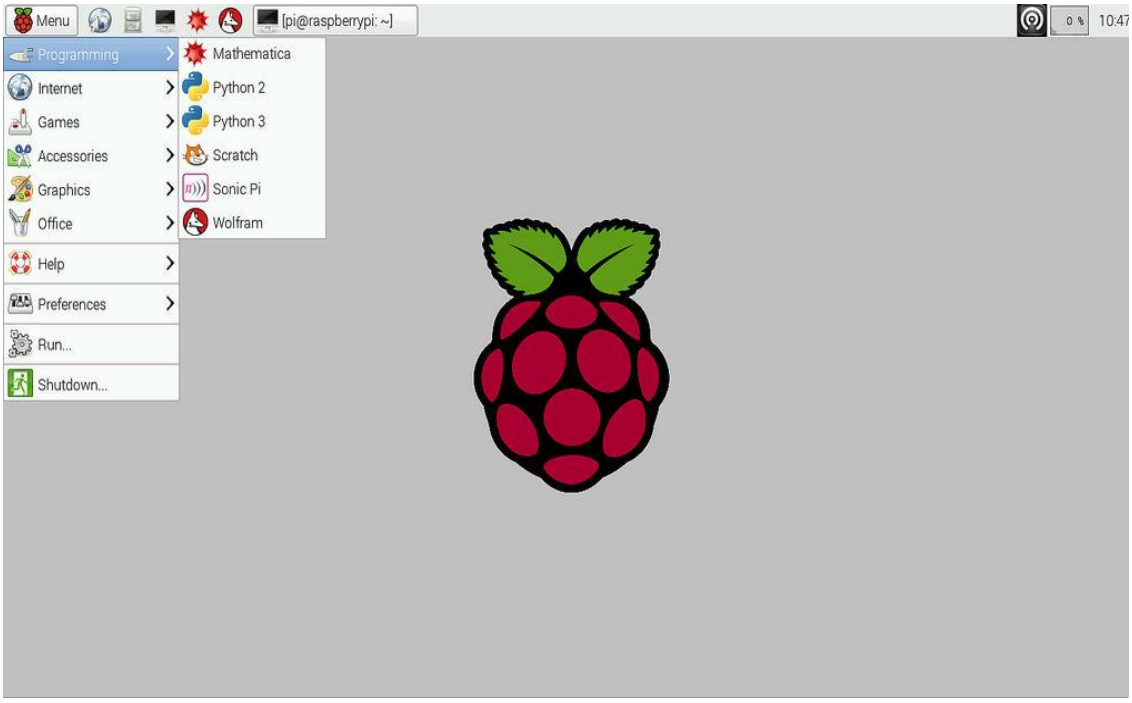


Figure 12: Raspbian Desktop.

4.2.2 Remote Desktop Connection (RDC)

Remote Desktop is a feature provided by an Operating System to allow another device to remotely connect it from an outside network or another location. The interaction seems as

though that the machine that is being accessed is done so locally. There are many different reasons that a user may want to use RDP such as:

- Troubleshoot and fix network issues from a different location.
- Perform administrative tasks.
- Control or setup another PC environment.

In the case of this project, we used RDS from a Windows machine which used RDP to access the Raspberry Pi. From there we were able to control the Pi's terminal and launch the node.Js server on port 8080. In order to successfully connect to the Raspberry Pi, the user has to enable and allow the RDP connection to be established to the RPi. This was done by accessing the Raspberry Pi's BIOS. To access the BIOS, you can enter '**sudo raspi-config**' in the terminal. This will restart the RPi and load the BIOS. An image of the BIOS settings can be seen below:

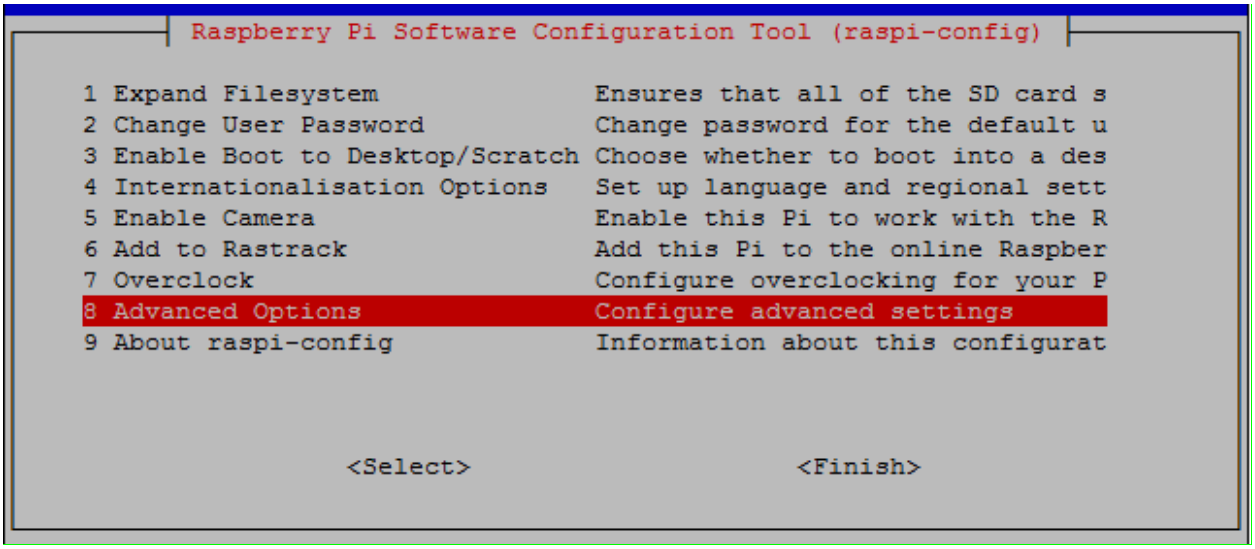


Figure 13: Raspberry Pi's BIOS Setting.

4.2.3 Lucidchart

Lucidchart is an online tool that allows the creation of different diagrams and charts. This was a really useful tool to use to help visualize how the project structure looks as well as to build multiple diagrams to explain the technologies used. It is a web application that is backed by cloud-based technologies allowing a user to save their workspaces and different diagrams they have

created. They also offer training and courses to help explain how their tools can be used to effectively create a stimulating and easy to understand diagram.

Below is an example of how Caffeine used Lucidchart to create their diagrams.

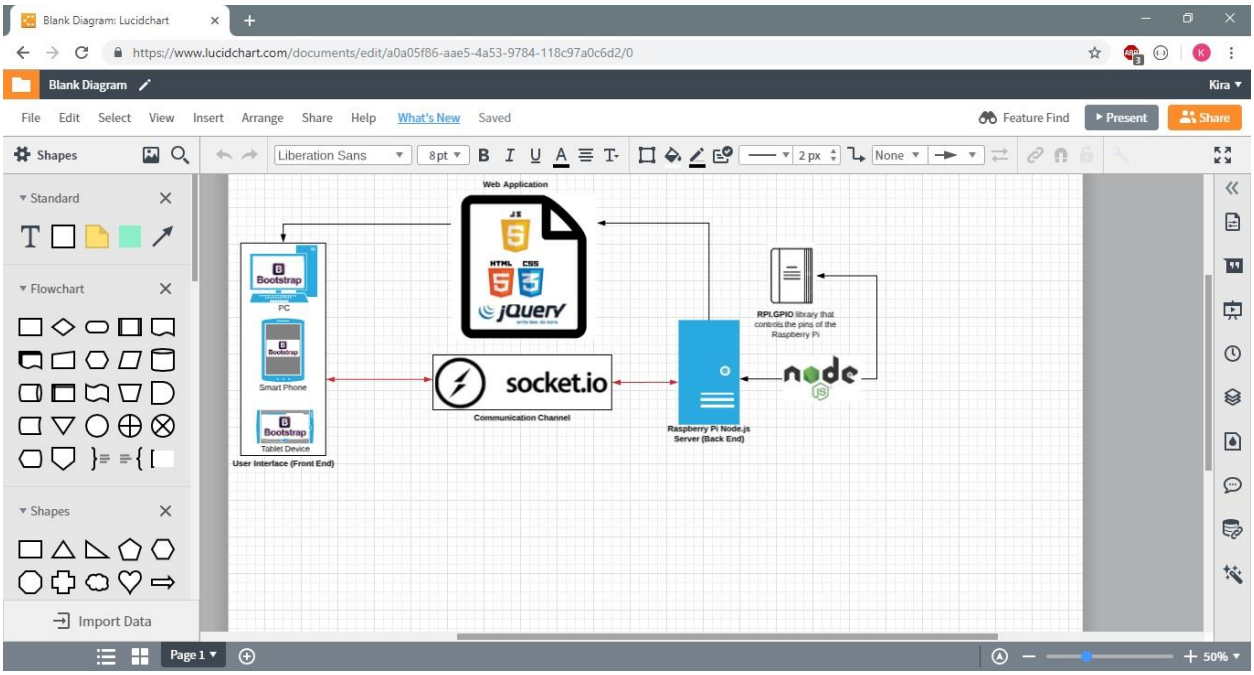


Figure 14: The Use of Lucidchart.

4.3 Programming Languages

4.3.1 Python

Python is an interpreted, object-oriented, high-level programming language, focused on offering simplicity, versatility and speed in the development of both applications and web pages. It is considered an interpreted language because there is no need to compile the source code to be able to execute it, which offers advantages such as the speed of development and disadvantages as a minor speed. Its popularity is mainly because it belongs to an open source development community. In addition, it contains a number of libraries, functions and data incorporated in its own language that helps perform many common tasks without the need to code them from scratch. The project uses Python 3 to create files dedicated to each available

movement of the robot and is applied through the web application (which will be created later on). It has been very helpful in the basic testing of the components that will be connected to build the final device. Two versions of Python are available: Python 2 and Python 3. Python 2 development ended with 2.7, which was released in 2010. Python 3 was first released in 2008 and is now the recommended versions to use for development, however, Python 2 is available for legacy applications which does not support Python 3 applications yet. Python 2 and Python 3 come pre-installed on Raspbian operating system, but to install Python on another Linux OS or to update it, a simple commands need to be run at the command prompt. To access the Python REPL (where can be entered Python commands through a command line) enter python or python3 depending on which version will be used. A read-eval-print loop (REPL), also known as an interactive top-level or language shell: Is a simple, interactive computer programming environment that takes single user inputs (i.e. single expressions), evaluates them, and returns the result to the user; a program written in a REPL environment is executed piecewise.

4.3.2 Python Libraries (RPi.GPIO)

The GPIO of the RPi is what makes this device so unique. The GPIO allows a moderately powerful microprocessor to talk directly to the circuits and interact with them by supplying power or switching their states (on /off). Handling I/O pins is often a very difficult task but there are Python libraries available that make it a less complex task for novice enthusiast. Some of the GPIO Python libraries implemented in the earlies stages of this project are:

RPi.GPIO: This package provides a class to control the GPIO on a Raspberry Pi. The RPi.GPIO library allows easy configuration of the input/output pins on the RPI's GPIO header within a Python script. The RPi.GPIO has been used during the process of building this project by allowing Python programs to access the GPIO pins and turning them on which turns on the DC motors.

GPIOZero: This library includes interfaces to many simple everyday components, as well as some more complex things like sensors, analogue-to-digital converters, full color LEDs, robotics kits and more. GPIO Zero is installed by default in the Raspbian desktop image, available from raspberrypi.org. Using the GPIO Zero library makes it easy to get started with controlling

GPIO devices with Python. This library was initially used in the project for making small tests such as turning on / off an LED light.

4.4 Web Application

4.4.1 Hypertext Markup Language (HTML)

HTML is short for Hyper Text Markup Language and is used to create electronic documents for the internet. These documents are defined and formatted by elements called tags and these tags determine how the content of the document is to be displayed to a user.

The creation of HTML belongs to Tim Berners-Lee during the 1990s and is now currently on its 5th official version, HTML5. According to Jeffery Veen of wired.com, Tim's creation was made with the idea of a user being able to easily navigate and view online documents. He created a hypertext language that was able to allow clients, through the hypertext format, to communicate with servers over the internet.

Today, HTML is the bones of many online documents or webpages a user interacts with over the internet. It is the official web standard and is maintained by the World Wide Web Consortium (W3C). The appearance and overlays can be adjusted with style sheets using CSS as well as giving it dynamic functionality with JavaScript.

4.4.2 Cascading Style Sheets (CSS):

Cascading Style Sheets is a one of the big three world wide web technologies. It is used to define visual appearances and formatting of HTML documents. It can be used to adjust text and background colors as well as set the widths, heights, and margins dimensions of content boxes.

The initial concept was proposed by Håkon Wium Lie in 1994 with the idea to separate the structure of a websites code to the visual design of it. CSS has since gone through different variance of itself with CSS 3 being split into different modules. Because of this, there will not able a specific CSS 4 as other modules could just be added or updated in CSS 3.

CSS allows an entire visual appearance of a web page to be updated and changes quickly and prioritize what is displayed on the screen. The cascading nature of it allows styles to inherit and overwrite other styles that have been assigned.

4.4.3 JavaScript

JavaScript is a programing language that is widely utilized for web development. It can dynamically change and update HTML and CSS files that are served to a user when they access a web page or web application. It is a useful language to implement as it can calculate, manipulate and validate data from a user.

According to Gediminas at Hostinger.com, JavaScript was created by Brandon Eich who worked for Netscape back in September 1995. The initial name was Mocha and it went through various changes before it was named LiveScript and then finally to JavaScript. The language has grown thanks to the community of developers who continue to work with it.

JavaScript is what brings a web page to life. It compiles and runs on the client's device and responds to their interaction such as button clicks and mouse movements. This also allows less network traffic to be processed on the server side making it a very lightweight programming language.

Below is an example of how JavaScript, HTML and CSS interact with each other:

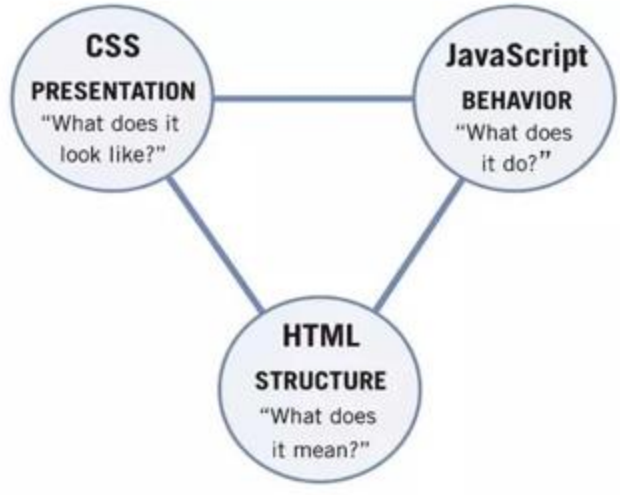


Figure 15: Interaction Between the 3 Main Web Application Technologies.

4.4.4 JQuery

JQuery is a cross-platform JavaScript library that allows easy transversals through the DOM tree model and provides developers a simple way to implement event handling, manipulation and animation of web pages.

According to Mary Parker of Coderwall.com, JQuery is offered as an open-source library and was initially developed in January of 2006 by John Resig who was inspired to create a simple way to keep HTML tags separated from JavaScript code.

With JQuery being easy to learn as it uses the same coding style and syntax as normal JavaScript, as well as having some key features such as HTML manipulation and DOM manipulation, it is used in over 66 million different websites across the world.

```
<!-- HTML -->
<button id="test">Click Me!</button>

<!-- JQuery -->
<script type="text/javascript">
    $("#test").click(function () {
        alert("I was clicked!");
    });
</script>

```

Figure 16: Action listener setup on a button using JQuery

4.4.5 Bootstrap

Rascia, T. (2016) describes Bootstrap as the most popular HTML, CSS and JS front-end framework for developing responsive, mobile first projects on the web. In other words, it takes HTML and CSS and scales it for which ever device is being used to create a better viewing experience to the client.

It was developed by 2 former Twitter employees (Mark Otto and Jacob Thornton) with a grid style layout that starts with 1 column and scales down to 12 in which developers can insert their content into whichever grid they'd like.

A very important aspect of Bootstrap is that it's open-source and just like many other open source projects, it has a large community of developers and designers continuously adding to it as well as creating colorful and responsive templates for others to use.

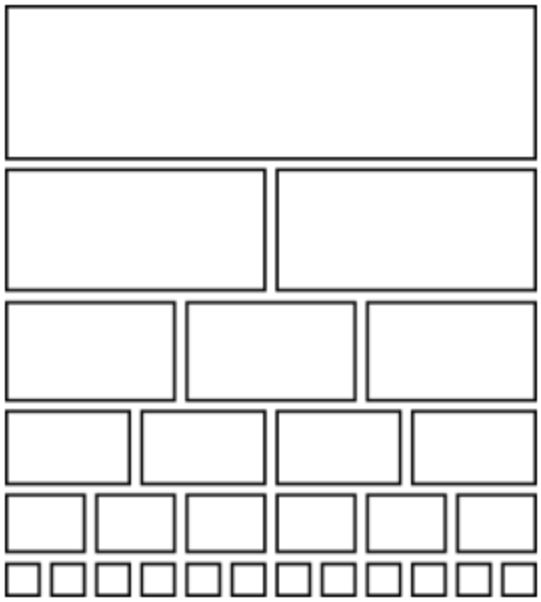


Figure 17: Basic Overview of the Bootstrap Gri.d

4.4.6 Node JS

Known an open source server-side platform that is used to easily build fast and scalable network applications. Those applications are JavaScript or Typescript and provides various libraries of JavaScript modules that simplify the development of web applications.

After seeing a progress bar of a file being uploaded to a server, Ryan Dahl was inspired to create Node.js. He was not satisfied with the way that the Apache HTTP servers were not able to handle concurrent requests and how some code that was being created would either block an entire process or create multiple executions stacks (excessive network traffic).

Node.js does not follow the typical request and response model of a web server. Instead it uses a single thread with an event loop to process requests by using callback functions to the systems operations such as the File System, Network, Computation etc. Once the callback functions are complete, a call will be made to Node.js and if there is any idle time, node.js will handle those calls. This is all part of the Node.js LibUv library.

Below is an example of how Node.js interacts with Chromes V8 Engine, the client's OS and LibUv library:

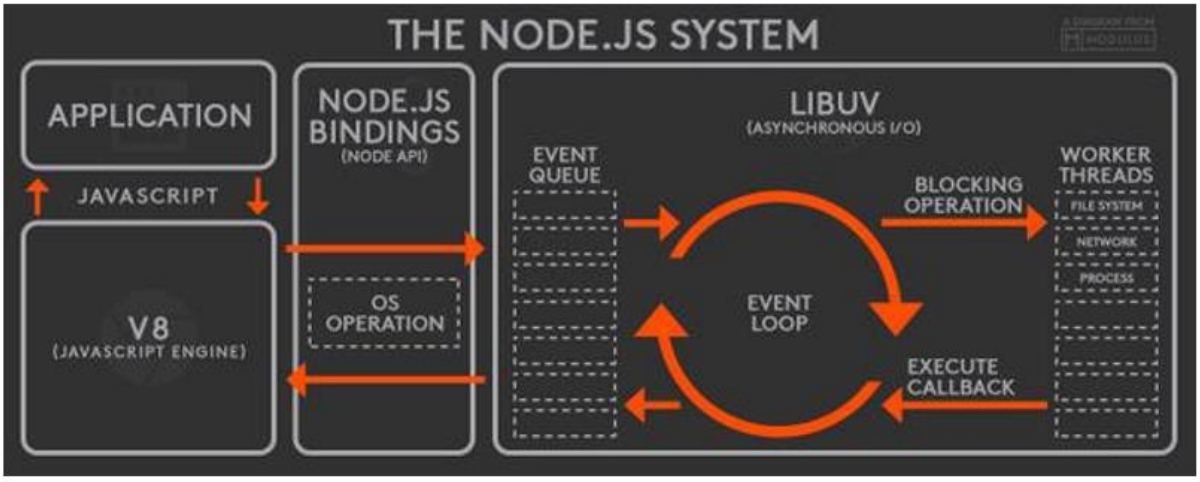


Figure 18: Basic Overview of How a Node.js System Works.

4.4.7 Socket.IO

SocketIO is a JavaScript library that incorporates the use of Web Sockets to enable real-time, synchronous communication between the client and a server. It is widely used for online games, real-time web applications as well as instant messaging applications. Web Socket is a communication protocol that uses full-duplex communication over a single TCP connection.

As Jacek Splawski explains, initially, an HTTP connection is made between the server and the client but if the clients web browser supports the Web Sockets protocol, it overwrites the HTTP connection header to keep the connection open. SocketIO is a library that implements this concept by having the connection open on the Server side and client side.

Below is a basic illustration of how a Server which has SocketIO setup communicates with clients who also have SocketIO setup.

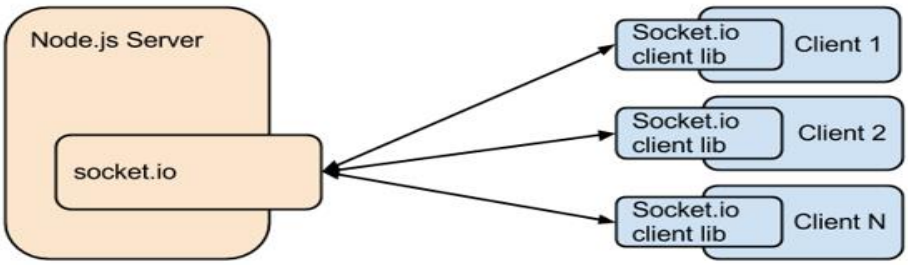


Figure 19: Basic overview of Socked.IO Communication Channel.

4.5 Networking

4.5.1 Internet Protocol (IP)

There are billions of devices that have been created, and more to be created that will require a way to connect from one network to another using the internet. IP addresses help to identify networks and the clients connected to it. It serves two main functions:

- for host or network interface identification
- location addressing.

IPv4, principal version used nowadays for every network in the world (until all the addresses run out). Each address is 32-bits long; this allows for a maximum of 4,294,967,296 (2^32) unique addresses.

One experiment done during this project required the IP address of the Raspberry Pi to be able to connect to it and allow it to be controlled remotely by a Windows OS. By doing so, we could run a script code from a laptop computer and (with the device build at this point) test the movement of the robot independently from cables attached to it.

4.5.2 Subnet Mask

Subnet mask set the limits of IP addresses available on a network, it allows the network designer to build a network according to the client specifications or needs. Systems within the same subnet can communicate directly with each other, while systems on different subnets can only communicate through a router.

4.5.3 Domain Name Service (DNS)

As the number of devices connected to the internet, wanting to connect to servers on webpages, the number of IP addresses increased exponentially and singular users could not simply remember an IP address on were to connect to a web site. For this reason, a system that

could relate an IP address to the name of the page (something easy for the user to remember) was developed. That system is DNS.

The Internet's DNS system works much like a phone book by managing the mapping between names and numbers. DNS servers translate requests for names into IP addresses, controlling which server an end user will reach when they type a domain name into their web browser.

4.5.4 TCP/ IP Model

The TCP/IP model was designed and developed by the Department of Defense (DoD) in the United States in 1960s and is based on standard protocols. It stands for Transmission Control Protocol/Internet Protocol. It contains four layers (unlike seven layers in the OSI model) which are the process and application layer, transport layer, internet layer and the network access/link layer.

4.5.5 Secure Shell (SSH)

Secure shell allows a user to remotely get access to a hosting account, it provides a user to control and navigate the terminal of a computer as if it was in front them. Other network protocols such as Telnet are often used for this purposes, but information transmitted using this protocol can be viewed without too much work as it works with a clear text transfer method. SSH works with encryption and allows the user to navigate securely as any message that is sent will be passed with an encryption.

4.6 Integrated Development Environment (IDE)

It is important to make sure that the developer is able to produce, store, compile, run, debug and test different code for their projects that they are working on. For this, an IDE is the best tool to use as it offers a way for the developers to oversee those aspects. According to

Veracode (2019) “The overall goal and main benefit of an integrated development environment is improved developer productivity. IDEs boost productivity by reducing setup time, increasing the speed of development tasks, keeping developers up to date and standardizing the development process”.

The RaspBot project was developed across multiple platforms with different Operating Systems and IDEs. The most part of the developing was done on a Windows 10 machine using Visual Studios IDE while testing and debugging was done on the Raspbian OS using Geany.

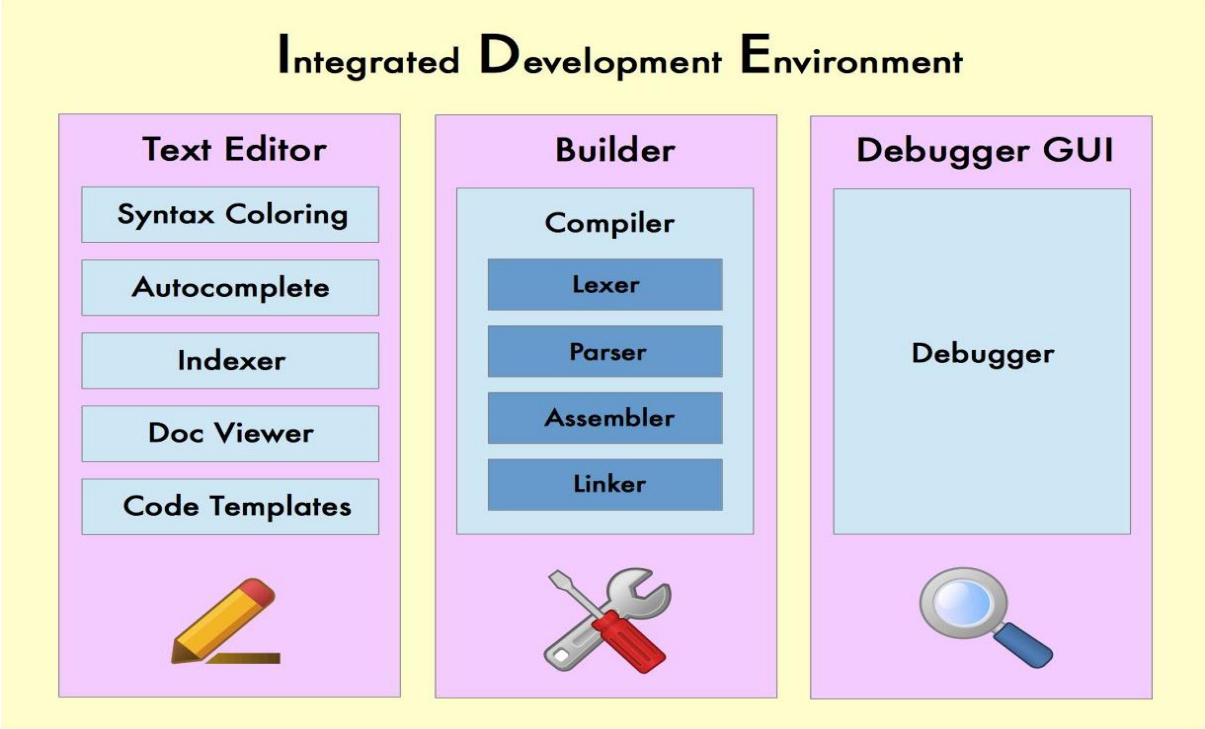


Figure 20: Basic Overview of an IDE.

4.6.1 Microsoft Visual Studio

Microsoft Visual Studio is an IDE from Microsoft. It has a wide range of functionality to help developers create computer programs, websites, web applications and services as well as even mobile applications. It contains a library of all the mainstream programming languages (C, C#, C++, Java, JavaScript, Perl, Python, Ruby etc.).

This project has made the use of Visual Studio to help develop the front and the back end of RaspBot. By easily importing and implementing different libraries and initializing the project's dependencies when it was migrated it to the Raspberry Pi made it very easy to deploy the project.

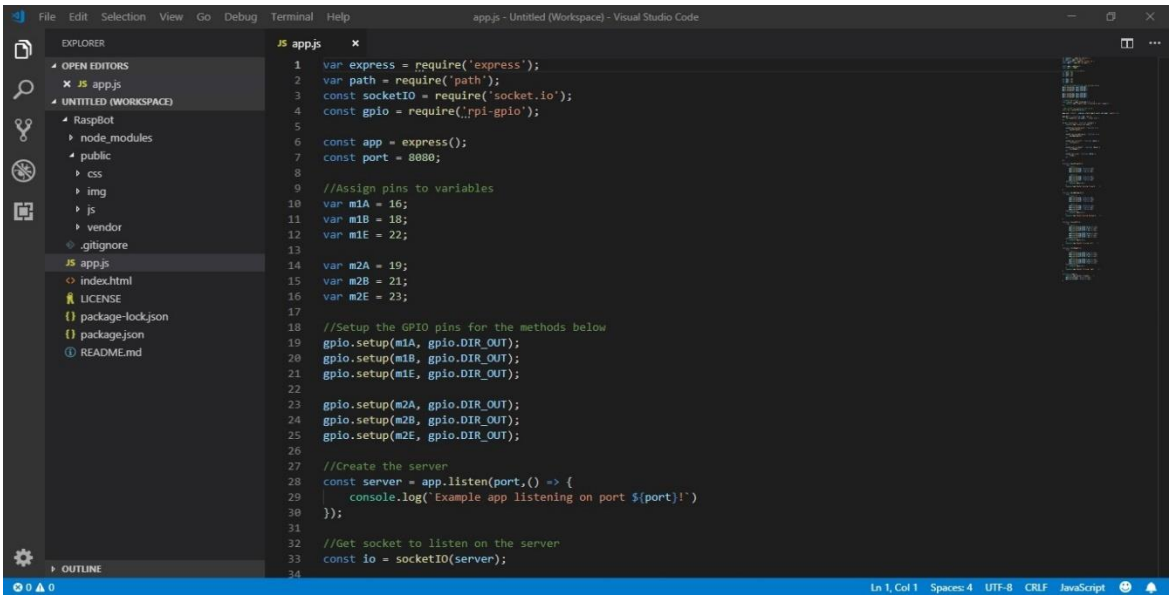


Figure 21: Microsoft Visual Studio GUI

4.6.2 Geany Text Editor

Geany is text editor that can be downloaded on the Raspberry Pi, it has basic features of an IDE such as syntax highlighting, symbol and auto completion as well as build systems to compile and execute your code. It also has a few dependencies from other packages making it a great lightweight code editor.

Geany was used as a simple code editor to make small adjustments to the code once the project files were migrated to the Raspberry Pi from the Windows 10 development environment. This code included changing the Socket.IO channels to listen on the Raspberry Pi's IP address. You can refer to Chapter 5.2.4.4.

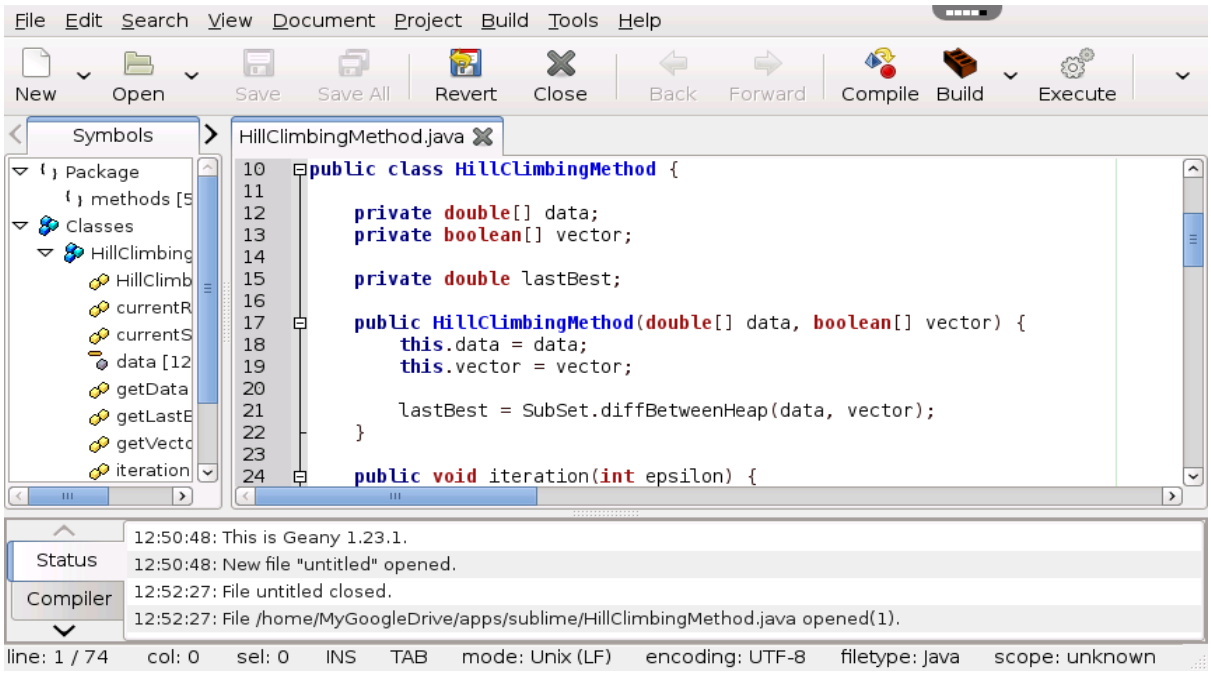


Figure 22: The use of Geany Text Editor.

Chapter 5

5 Design: Mechanical and Software

This chapter explains the design of the project in 2 separate parts, the hardware and the software. It will give the reader an understanding of Caffeine's design choices and incorporation of each technology used. It will also explain to the user how they could design and implement their own prototype from the examples from this project and other components that is used.

5.1 Mechanical Design

RaspBot as any other device that is designed for motion using multiple wheels, will need a structure in which it will contain the different components that make it a live device. Once an idea of how to build the body for the RaspBot was finale, the next step was deciding the shape that we wanted it to be in. This required us to do a bit of research to gather enough knowledge that allowed us to design and build something that was sturdy and reliable. You can see some of the elements required by RaspBot in the below figures.

Figure 23: Left top, Swivel wheel, motors, copper clamps and wheels. Right top, Raspberry Pi. Left Bottom, Breadboard with chip L293D. Right Bottom: Battery bank 10000mA.

After we conducted enough research, we chose a build that is similar to a buggy car with a platform, wheels and motors that can provide a strong but light structure using LEGO blocks. In the following paragraph, there will be an explanation on how we came up with the design features, the mechanical design itself, its evolution, changes, constraints and different reasons that drove us to the final mechanical design achieved.

5.1.1 Central Processing Module

When we started the project we had to decide between an Arduino and a Raspberry Pi to use as Central Processing Module. We made this decision for the RPi due to its ability to run multiple programs at the same time providing a variety of functionalities to RaspBot, using a RPi we leave a door open to continuously add more utilities and features (such as the use of sensors, autonomous Wi-Fi, camera, GPS, mapping software, etc.). This allows RaspBot to reach another level of functionalities that can be mixed with IoT and which allows many different variety of uses.

5.1.2 Platform

We decided to build the platform using LEGO blocks as this element allows us to build something which is not too heavy and has the ability to accommodate all other of the elements required. The platform is shaped by two areas which is the base and the chassis.

We choose blue as the color of our prototype. Blue is the colour of the sky and sea that usually is related with a cool and calming colour that also represent creativity and intelligence, reason that finally made us to take de decision of use it.

Figure 24: Tools Used to Cut and Adjust Elements.

5.1.3 Base

The base represents the ground of the RaspBot where some of the components such as the RPi, battery bank and breadboard can be located.

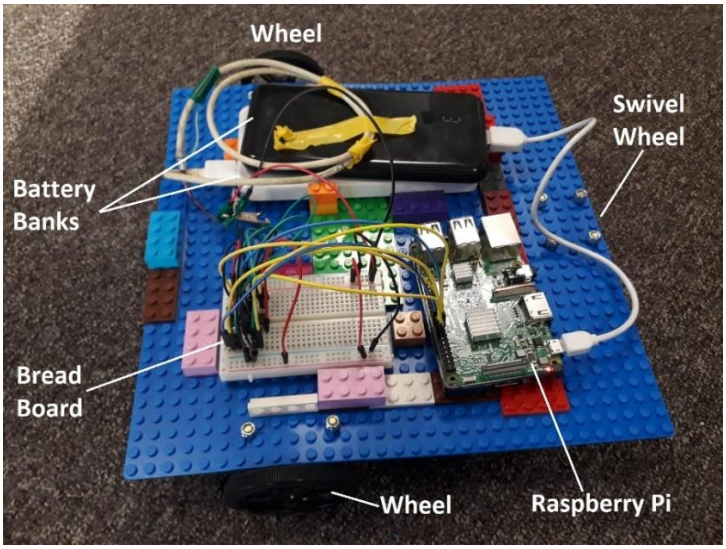


Figure 25: RaspBot Initial Prototype

Initially we choose to use a LEGO base plate with dimensions of 38cm x 38cm, the base plate was very thin and as result, was not able to hold all the elements without flexing. Beside that the distance between support points was an important factor to the flexion presented on it.

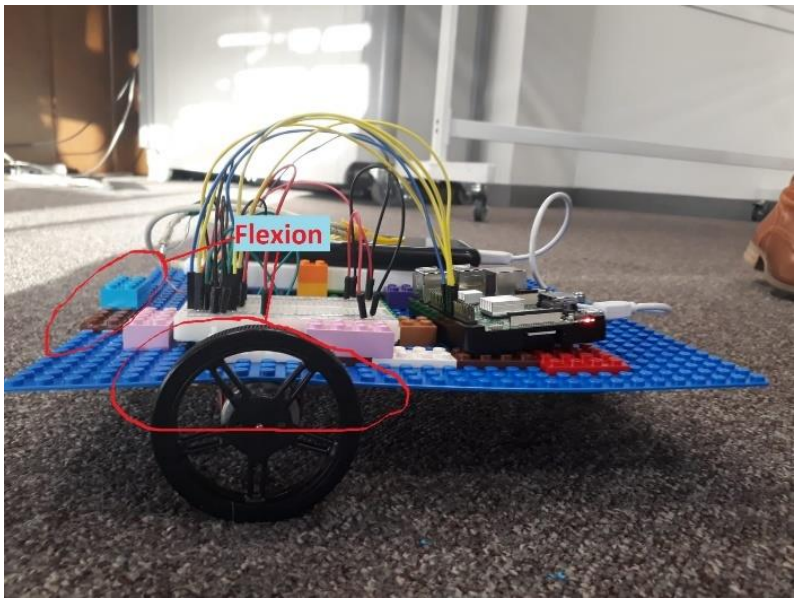


Figure 26: Platform Flexion.

The issues mentioned above brought us to modify the dimensions of the platform by reducing it. From this we saw that the flexion issue was still present but in less proportion.

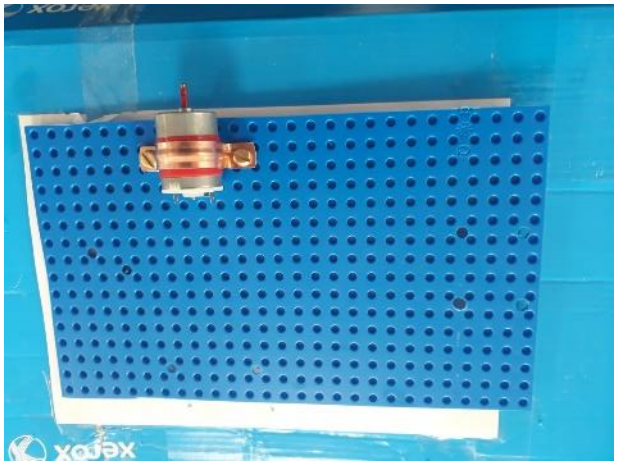


Figure 27:Base With Dimension Reduced.

The dimension change not only improved the platform resistance but also improved the esthetic of the mechanical design as well. However, we still needed to enhance it because flexion was still present. For that reason, after we analyzed and thought about how to improve it, we got the idea of create a chassis and install it on the platform to give the strength required to reduce or eliminate the flexion. The chassis structure was made based from the previous experience of one of our members combined with the Trucks platform chassis.

5.1.4 Chassis

As was mentioned above, the idea of building a chassis and installing it on the base was to create a strong platform that allowed the RPi to contain all the elements required without generating a problem for the RaspBot while in motion.

This structure was made using the leftover pieces from the LEGO base plate after we reduced the base dimensions. A LEGO door frame and super glue was used to stick all the pieces together and finally stick it to the base. Below is an image of the Chassis after it was assembled.

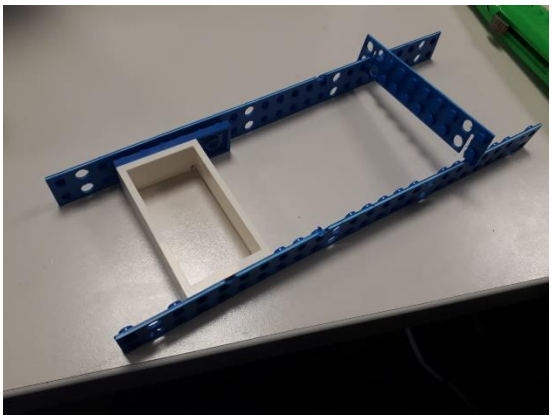


Figure 28: Chassis Structure.

Finally, after assembling the chassis we installed it on the base getting the result of the platform that you can see in the below figure. This final structure gave enough resistance to the platform that was needed, making it able to hold all the elements without any sign of flexion.

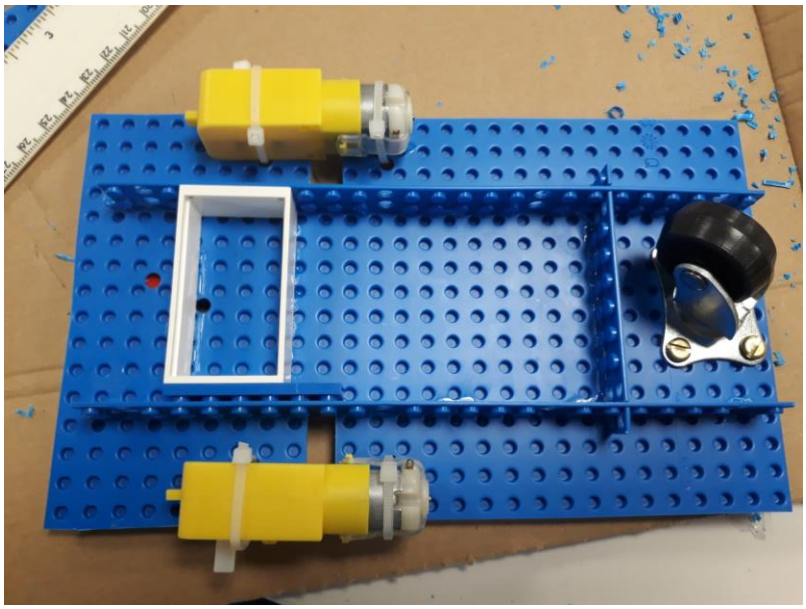


Figure 29: Platform. Base and Chassis Assembled with Motors.

5.1.5 Second Level Platform

Based on the necessity to keep all the elements tidy and organized we decided to create a new level platform to hold the RPi and Breadboard, while the lower level holds the battery bank.

We created this new platform and gave it strength by adding a structure based on the three beams that were being used (front, middle and back).

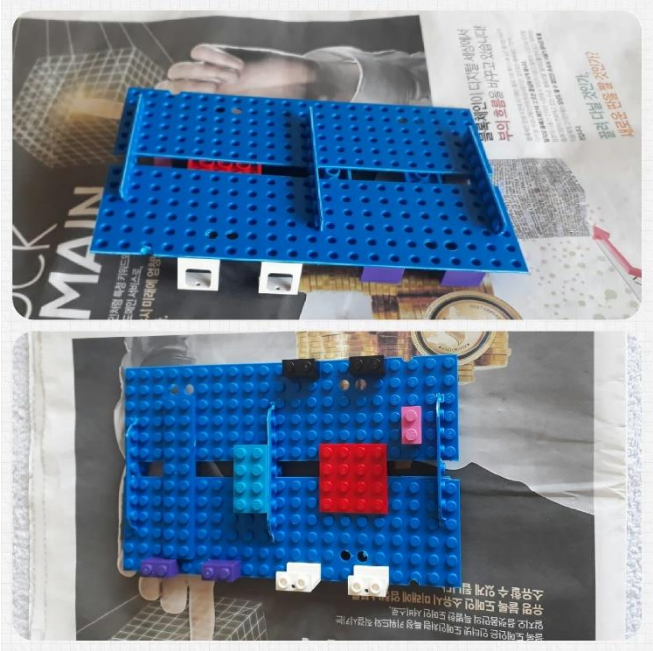


Figure 30: Base Where is Located the RPi and Breadboard. Top: Down Side and Bottom: Up Side.

5.1.6 Motion System

At the beginning of the project we were thinking about creating this device with two wheels and four motors (2 in the front and other two at the back). However, after some analysis into the scripts that executed the movement for the RaspBot we decided to implement only two wheels and motors at the back and have one swivel wheel at front. This design gave us the freedom of movement without the need to synchronize too many elements.

The motion system is shaped by: motors, wheels, a swivel wheel where each motor is independent from the other. The logic that make this work is based on the way that we want to make the RaspBot move. For it to:

- Move forward: Turn on both motors spinning in a clockwise direction
- Move backward: Turn on both motors spinning in an anti-clockwise direction
- Turn left: Turn off the left motor and turn on the right motor spinning in a clockwise direction
- Turn right: Turn off the right motor and turn on the left motor spinning in a clockwise direction

The initial prototype took a pair of medium torque DC motors that works at 5000 rpm consuming 3v, then we realized that this kind of motor was not capable to provide enough torque to beat the RaspBot's inertia and generate motion. This was due to the medium torque DC motors only being able to give motion to light elements, our motor in specific was able to move 20 g when RaspBot require a motor able to move at least 300 g.

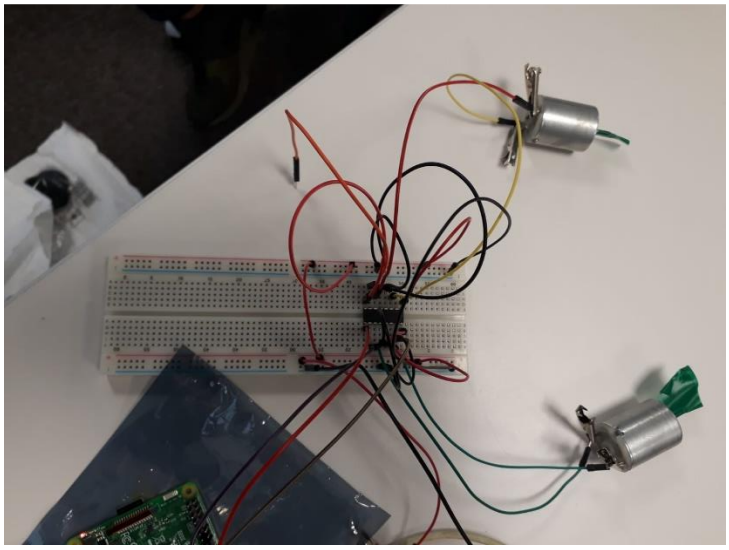


Figure 31: Medium Torque DC Motors.

Finally, after some research we found out a motor able to beat the RaspBot's inertia and provide the motion required, these motors are right angle geared DC motors, you can refer to Chapter 4.1.9 to review the information on it. It works at 90 RPM consuming 3v.

Figure 32: Right Angle Gear DC Motor.

5.1.7 Wheel

Choosing the wheels required for RaspBot was not a difficult decision to make as this wheel was the only based for this motor that we have no implemented. Initially we bought a pair of wheels that matched with the medium torque DC motor that we had before but after the we changed the motors we had to change the wheels as there were not ergonomic enough to fit onto the motors as well as to carry the weight of the RaspBot. The wheel is shown in the next picture.

Figure 33: Right Angle Gear DC Motor and its Wheel

5.1.8 Swivel Wheel

After few discussion and many brainstorming sessions about how RaspBot's turning mechanisms will be, we decided to establish a structure conformed by two regular wheels assembled with a motor on each one of them. They would be positioned at the back with one swivel wheel at the very front that will not offer too much opposition to the change of direction and facilitate the RaspBot's turning movement.

Our decision was made based on the idea that RaspBot will stop one motor and turn the other to make a turn, this in conjunction with the swivel wheel will allow a smooth change of direction. The picture below shows a swivel wheel.

Figure 34: Swivel Wheel.

5.1.9 Power Supply

In relation to the power supply we decided to work with two battery banks, one to supply energy to the RPi and another to the motors due to in one of our initials experiments we thought that we need one battery for the RPi and another for the motors.

Later we realized that only one battery is enough, using the chip L293D generating as consequence a lighter and compact prototype.

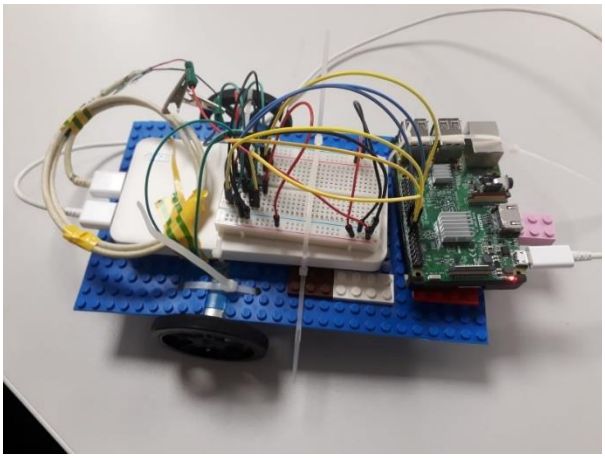


Figure 35: New Size, More Compact and Lighter.

5.1.10 Final Mechanical Design

Once we defined all the elements and positioned them in the correct place we drilled some holes to allow the passing of the jumpers, added some LEGO blocks to give an esthetical design, reinforced the elements through the use of super glue and added some minor LEGO pieces to achieve the final result of the RaspBot Mechanical Design.

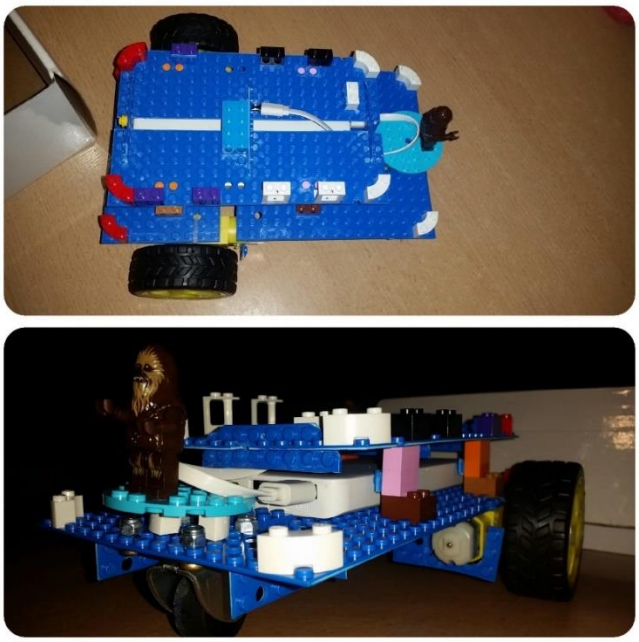


Figure 36: RaspBot With Extra Level for RPi and Breadboard.

5.2 Software Design

This chapter will outline the proposed software system design for the RaspBot and its functionality. It will explain how a user would be able to connect to the RaspBot and control its movements from a control panel which is hosted on a web application. The technical specifications listed inside are all found through research done by Caffeine members to achieve the core objectives of this project.

5.2.1 The purpose

We will describe the purpose and implementations of the web application technologies (see chapter 4 – technologies) and how these technologies are used to:

- Set up the Raspberry Pi as a server.
- Set up the functionality for the pins (motors).
- Create a connection between the user and the Raspberry Pi.
- Allow a user to control the pins with a press of a button.

This chapter will also explain the architecture of the web application and will act as a guide to create a replica of this project.

5.2.2 The System Overview

The system of this project includes a basic interface (Web Application) that a user can interact with to control a Raspberry Pi. This concept stems from the IoT and mimics many projects system design that are based around it. An example of this would be a home heating system that a user can adjust from his smart phone or computer. Instead of manually going to his heating system and adjusting it by hand, he is able to access and control the heating system through his home network (local network) from a touch of a button. The concept can be depicted below:

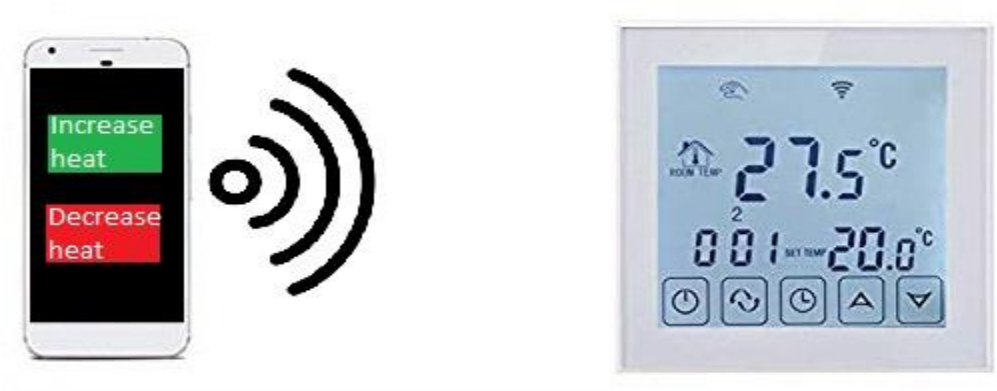


Figure 37: Design Style of IoT Project for Home Heating.

With regards to RaspBot, it follows a very similar concept and design structure. A user will press a button on their screen and the robot will move. As long as the button is being pressed, the robot will continue to move. The main difference between a design like the one above and the design of the RaspBot is that there is no data being passed between the user and the RaspBot. The heating system design would need to send, read, adjust and monitor the data to control the heat setting but there are no variables like that for this project. However, it should be stated that this project does have data values set up in its code in the case of if data is ever needed to passed. They have been left blank and can be incorporated at any time.

We aimed to not over complicate the design of this software architecture as there were many technical specifications and factors to consider. We wanted to keep the design as simple as possible so that other people may implement something similar for themselves and use this document as a reference to cut out a lot of developing time. This has become a benefit of this project design as its high functionality with its simple structure makes it easier to follow. It also serves as a base for more experienced learners to mirror this project and add their own implementations and other features they wish, such as adding a motion sensor.

The web application will provide the following capabilities to the RaspBot:

- Move forward
- Move backward
- Turn left
- Turn right

For this, the project architecture is broken down into 2 separate parts. The front end which shows the user a control system and the back end which enables the Raspberry Pi to act as a server and control hub for the RaspBot. The overview of both sides can be seen below:

Figure 38:Front end (Left) and Back end (Right).

The Raspberry Pi on the back end has all the necessary project files stored that are needed to run a Node.js (refer to chapter 4 – technologies) server on a local network as well as display the web application and control panel to the user.

It is the server that receives messages and the actual robot that respond to those messages by moving. An overview of the folder layout with descriptions what file does can be found below.

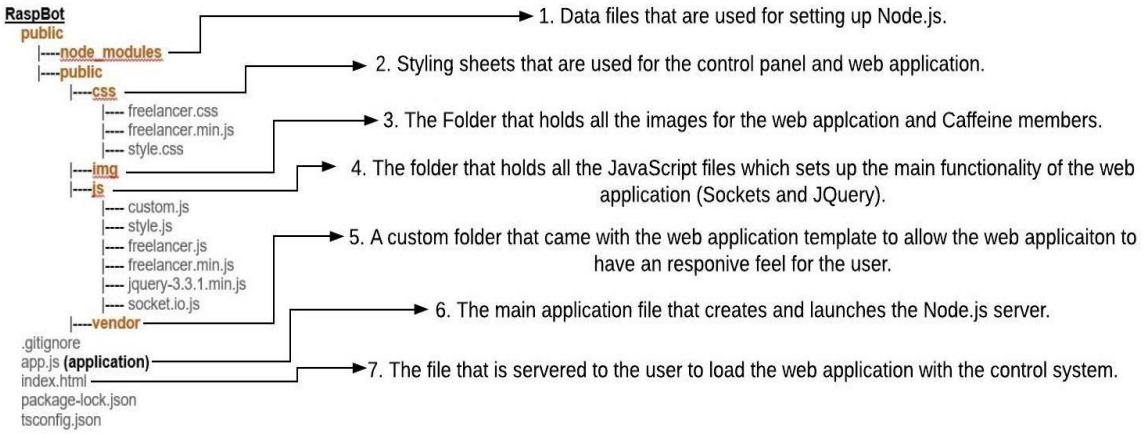


Figure 39: Final Project Folder Structure

5.2.3 The System Architecture

The components that make up this project have all been listed in the previous chapter of web application technologies, for a better explanation you can refer back to them. The way that these technologies have been incorporated in this project are as follows:

5.2.3.1 HTML & CSS

HTML is used to construct the control panel as well as the speed control systems. Each button in the control panel has a unique ID that will allow the server to process the request and execute the correct action listener. The rest of the web page is set up through a bootstrap template with added style sheets which are both from the template as well as custom made.

5.2.3.2 JavaScript

JavaScript is used in this project design to incorporate the core functionality (controller) of the Raspbot. Its use is extended from:

- setting up the web server for the localhost.
- serving a static interactive single page web application to a user.
- opening a real-time communication channel between a user and the RaspBot.
- adding action listeners to process interactions from the user to the Raspberry Pi's GPIO pins.

5.2.3.3 JQuery

With each action listener assigned to each button on the control panel, JQuery is set up to handle the request through the Socket.IO channel. The requests happen in real-time and will executes as soon as the user interacts with it. Some of the other features that are used for JQuery have been implemented in the template for the web application. IE for the dynamic flow and interaction for the user.

5.2.3.4 Bootstrap

As there are many different smart devices for users to use these days, we wanted to ensure that if a user connected to the RaspBot they would still have a responsive, scaled interface despite the device they use. We searched for a Bootstrap template that resonated with our project goals and system design. The template was changed by having the control system (that was created by Caffeine) inserted.

5.2.3.5 Node.js

Node.js is the framework that the web application is running off. It is used to host the server and allows users to make requests through HTTP or Web Sockets (Socket.IO). The web pages and other styling sheets are served to the user when they connect and any requests or events are handled through Node.js

5.2.3.6 Socket.IO

Socket.IO could be considered the blood flow of this project. Once a client has connected to the server, a fast, responsive real-time movement from the robot is achieved when a user presses a button.

5.2.3.7 RPI GPIO

This library was implemented with Socket.IO to enable the Raspbot to move after the user interacts with the Raspberry Pi from a click of the button.

- The button on the control system in the web application would send a Socket signal to the server.
- The server would evaluate which signal was ‘emitted’.
- The server would then call the appropriate function based on that signal’s name through the action listeners.
- The function would be used to turn on the correct pins using the RPI GPIO library.

An overview of the system design and architecture as well as how these components linked with each other can be seen below:

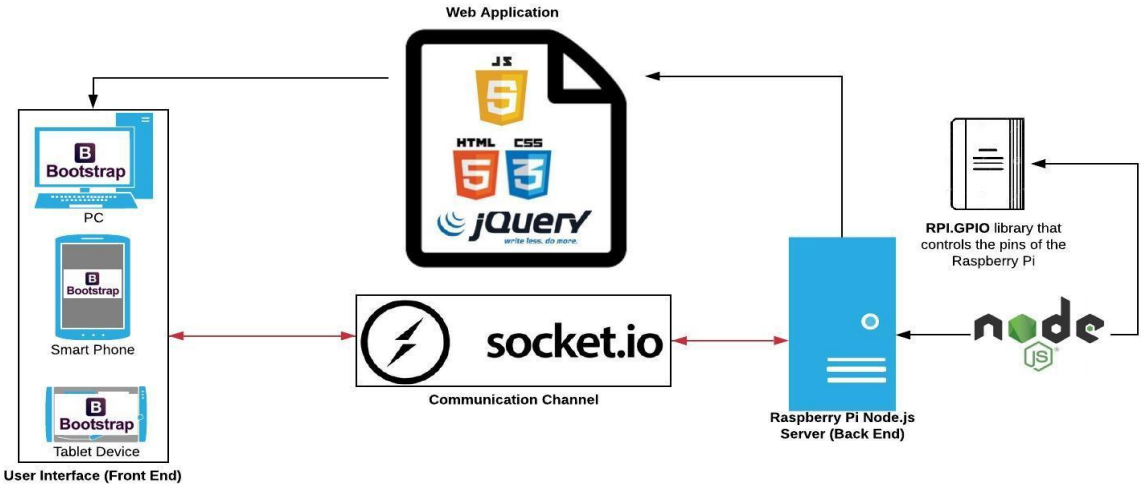


Figure 40: Overview of the System Design

5.2.4 Creating a Prototype

In this part of the chapter we will explain:

- What is needed to start the project.
- How to get the correct components in the correct place.
- How to make a small connection between the Raspberry Pi (IoT device) and a user.

We assume that the user has already configured the Raspberry Pi and enabled the Wi-Fi adapter, you can refer to experiment 1 if you need a refreshed explanation.

5.2.4.1 Setting up the Project Environment

The first thing to do would be to access the Raspberry Pi and create a folder anywhere on the Desktop. This folder will house the project files needed to run a Node.js server which uses Express to run the application.

Once the project folder has been created you will need to initialize it and add the correct modules. Open up the terminal and select the directory of your project folder. Once you are in the correct directory you will need type the following:

- `npm innit –y`

To install node.js you need to type the following:

- `apt-get install -y nodejs`

The other modules you will need to install would be:

- `npm install express`
- `npm install rpi.gpio (Only for Raspbian OS)`
- `npm install socket.io`
- `npm install socket.io-client`

To check if node and the other dependeicies were insatlled correctly you can type “node –v” in the terminal and view the response. After you can pen your package.json file to see if the dependecies are all listed there:

```
{
  "name": "raspbot",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.16.4",
    "rpi-gpio": "^2.1.3",
    "socket.io": "^2.2.0",
    "socket.io-client": "^2.2.0"
  }
}
```

Figure 41: The Dependencies that the Project Uses.

5.2.4.2 Adding the Backend

After all the decencies have been installed you can create a new file and name it ‘**app.js**’. This file will be the main application file that node.js will run from.

You can add the following code to the file to get started:

```
//Create the variables from the installed dependencies
```

```
const express = require('express');
const socketIO = require('socket.io');
const path = require('path');
const gpio = require('rpi-gpio');

//Create the express application and the port number to listen on
const app = express();
const port = 8080;

//Assign a pin from the Raspberry Pi's GPIO to a variable
const pin = 16;

//Setup the pin for GPIO functionality
gpio.setup(pin, gpio.DIR_OUT);

//Create the server
const server = app.listen(port,() => {
    console.log(`Example app listening on port ${port}!`)
});

//Add a static path for node.js to send the correct files for the user to utilize
Socket.IO and JQuery
app.use('/static', express.static(path.join(__dirname, 'public')));

//Send the control system to the user when they connect to the server
app.get('/', function (req, res) {
    res.sendFile(__dirname+'/index.html');
});
```

```
//Get the socket to establish a channel to the server
const io = socketIO(server);


//Set the socket to listen for when a button (turnON) is pressed
io.on('connection', function (socket) {
    console.log("socket connection made");


    //A function to turn on the pin
    socket.on('turnON', function () {
        try{
            gpio.write(pin, true,()=>{});
            console.log("Pin 16 has been turned on");
```

```
        }catch(error){
            console.log(error);
        }
    });
});
```

5.2.4.3 Adding the Frontend

The next part would be to set up a basic frontend interface for the user to press a button. Create a new file called '**index.html**'. Please ensure that you have a reference link to your JavaScript files (**socket.io.js**, **custom.js**, **jquery-3.1.1.min.js**).

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <!-- My Custom JS-->
    <script src="./static/js/socket.io.js"></script>
    <script src="./static/js/jquery-3.3.1.min.js"></script>
    <script src="./static/js/custom.js"></script>

</head>

<body>
    <button type="button" id = "turnON">
        Turn On
    </button>
</body>
</html>
```

5.2.4.4 Adding Interaction Between the Frontend and Backend

Now that we have the front and the backend set up, we need to make sure that when a user from the frontend pushes a button that the server is able to process that action and respond in the right way.

Create a new folder called '**public**' and in that folder you can create another sub folder called '**js**'. Inside this folder you can create a new file called '**custom.js**'.

In this file ensure that the variable for socket is made and is referring to the localhost address with the port number (or any address and port number that the server is running from).

```
const socket = io.connect('http://localhost:8080');

$('document').ready(function(){
    console.log("client is connected to server");
```

```
    //Pin on
    $('[id^=turnON]').on("click", function(){
        console.log("Button is turned on...");
        socket.emit('tunrON', { my: 'data' });
    });


});
```

5.2.4.5 Connecting the Frontend, Backend and Importing JQuery

Now that the frontend, backend, and the interaction between them has been set up, all that's left to do is establish the Socket.IO connection.

Create a file in the '**js**' folder and call it '**socket.io.js**'. Take the meta-code from the following website and paste it in this file.

https://raw.githubusercontent.com/socketio/socket.io-client/master/dist/socket.io.js

After that is done you can create a new file in the same folder and call it '**jquery-3.3.1.min.js**'.

Take the meta-code from the following website and paste it in this file.

https://code.jquery.com/jquery-3.3.1.min.js

And that's it! This has completed the basic architecture set up between the user and the Raspberry Pi. Any other features such as Bootstrap implementation can be done on any else's accord. The basic folder structure should look like this:

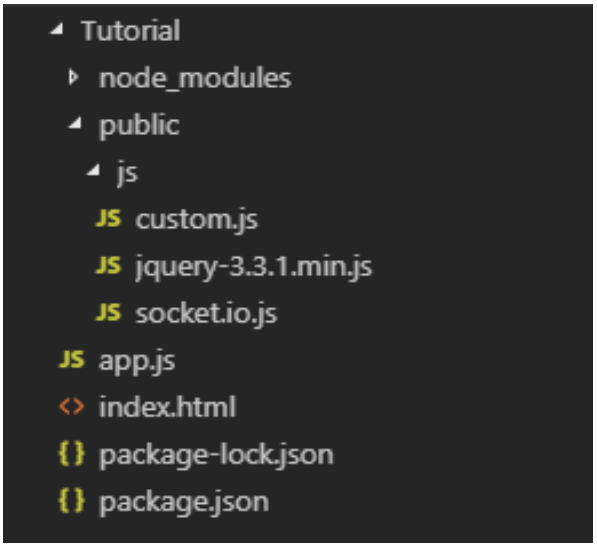


Figure 42: The Prototype Folder Structure.

5.2.5 Design Constrains

During the development phases of the project, Caffeine encountered a few constraints that have impacted the end result of the project. Some of these constraints were beyond the scope of the project outline while others were due to members not having the right tools or knowledge to complete the objectives.

These constraints are listed as:

- Not being able to purchase the correct modules to allow the Raspberry Pi to connect to a 4G mobile network set the project back in reaching one of its core objectives. The 4G HAT module that fits itself on top of the Raspberry Pi would allow a micro SIM card to be inserted and using one of the Network Service Providers 4G channels, allow a user to connect to and control the Raspberry Pi.
- Lack of knowledge for security implementation means that there are no security features set in place. This could pose a risk for the RaspBot as anyone who is connected to the local network would be able to access the RaspBot and 'hijack' the controls. If this project was another IoT concept (such as the heating system) that passes data, it could potentially be a high level security risk for the users and their private network.

- Not having enough time meant that we weren't able to implement the speed control of the RaspBot. Although the interface for the speed control system is included in the user interface with the three different speed settings, it was not connected to any back end code. This left the RaspBot with the same speed setting but room for allowing other developers the opportunity to experiment with it.

Chapter 6

6 Design and Implementation

This chapter will go through all the different design ideas and experiments that Caffeine undertook. It outlines each relevant experiment and their findings to better help the reader understand what kind of process Caffeine went through when trying to reach its final goals.

6.1 Experiment 1: First contact with a Raspberry Pi.

The RaspBot's main component is the Raspberry Pi, which means the first element that we need to understand and control was the Raspberry Pi. The first experiment consisted of the following components:

- Raspberry Pi 3B.
- Bread board
- GPIO wires.
- Red LED
- 330-ohm Resistor
- Mouse (USB Connection)
- Key board (USB Connection)
- Monitor (HDMI)

6.1.1 Exploiting the Raspberry Pi uses and functionalities

In this experiment we powered on the RPi for the very first time. We download and configured the OS (Raspbian - based on Debian). To achieve this, it was necessary to use a keyboard, mouse and monitor. Using NOOBS, we installed Raspbian OS, then configured the time and location (check reference for video "Getting started with the Raspberry").

6.1.2 First experience with a Pi

Once the RPi was installed and configured, we used Python 3. This scripting language was already integrated in the device. The next step was to create a script that allows the turning on and off of a red LED by enabling or disabling a GPIO port (which was connected to the red LED). During this phase we used the RPi.GPIO library and GPIO.Board mode instead of GPIO.BCM because with the GPIO.BCM, the ports will be different if a different Raspberry Pi model is used. We didn't want this project or the experiments be solely connected to the RPi model 3B.

We actually burnt out one of the LED diodes due to the connection being in the wrong position as well as not using a resistor. The resistor helps to avoid the LED bursting (overheating) due to the heat generated by the circuit.

After identifying the ports to be used as well as the red LED's polarization, we created a python script where we imported the RPi.GPIO library, set the GPIO mode and setup the port to be in use as input or output, in this case as output because that port would provide the power and could be identified with .HIGH (open port) or .LOW (closed port). We then connected the electronic circuits as can be seen in figure XX and figure XX, where the LED's anode has to be connected to the output port assigned in the python script and the LED's cathode to the resistor and this last element to the ground port of the breadboard or Raspberry Pi.

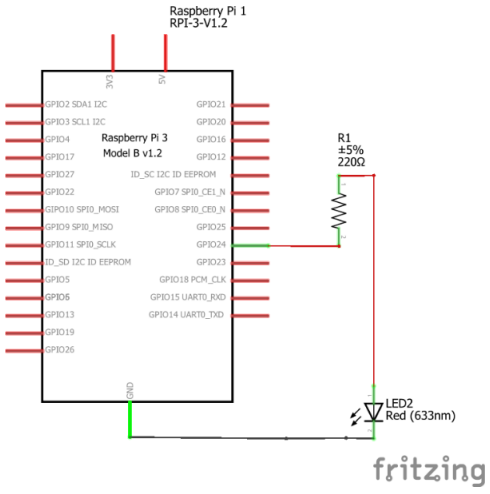


Figure 43: Circuit Used to Turn on the LED.

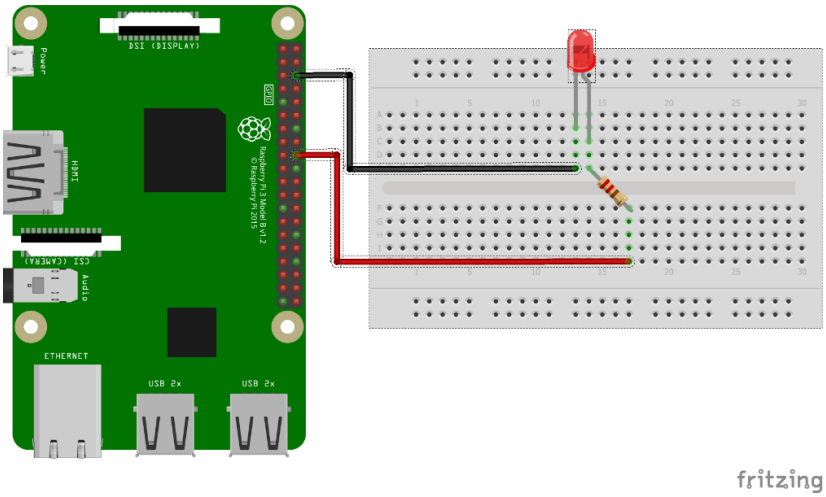


Figure 44: Connection Between RPI, LED and Resistor, to Complete the Experiment.

Finally, we ran the python script getting the result of the LED turning on and off based on the python script used.

6.1.3 Findings and troubleshooting:

- The NOOBS installer and Wi-Fi connection (or any internet connection) is required to install the Raspbian OS. A keyboard, mouse and a monitor are necessary to get initial access to the Raspberry, after the configuration and using a constant network you can get access through Remote Desktop from a Windows OS (development environment) using the IP address that was assigned to the RPi.
- The resistor can be connected to the positive or negative pole of the LED; it won't affect the LED.
- The LED polarization and resistor is really important to avoid burning out.
- Python has different libraries for GPIO, the most common ones are GPIO zero and RPi, which were was used in our project. GPIO zero may be easier to use due to it being more user friendly.
- The GPIO.RPi has two modes to identify the Raspberry Pi pins, one is the BCM which its configuration depends more on the RPi model and board in which the pins numeration is

always the same independently of the RPi model. We chose Board mode because in case of any device change we will not have to change the code base on the pins numeration.

- The command time.sleep(i), this python command suspends the execution of the script for an amount of time, where "i" is the time in seconds that the script will be suspended.

6.2 Experiment 2: Interacting with DC Motors

6.2.1 Phase I

To start to be familiarized with the components of the robot, a first interaction with the DC motor allowed us to understand how it can work with the RPi. The elements used for this first experiment were:

- Raspberry Pi 3B.
- Bread board.
- GPIO wires.
- 1 DC motor (Medium torque)
- 1 Mouse (USB Connection)
- 1 Key board (USB Connection)
- 1 Monitor (HDMI)

6.2.1.1 Starting with DC Motor (Medium Torque)

According to the science web page "www.MiniScience.com" the typical current or power necessary to start a medium torque DC motor is 350 milliamperes (mA). For this experiment we are using the pin #2 or 5 volts, which is connected directly to the Pi's power input, providing the current left after powering the Pi's, proximally 1.5 Ampere (1500mA).

6.2.1.2 Running the DC Motor from the Raspberry Pi

Having the Raspberry Pi already configured from the previous experiment to run the python scripts and the necessary libraries installed, we focused on setting the connection of the DC motor with the GPIO ports choose from the computer so that the engine can start to be tested.

Online writer author, Avayan, on 2009 outline two important facts about powering DC motors:

- "DC Motor rotation has nothing to do with the voltage magnitude or the current magnitude flowing through the motor."
- "DC Motor rotation does have to do with the voltage polarity and the direction of the current flow."

The figure below represents how the rotation can be changed by switching the current flow on the Dc motor:

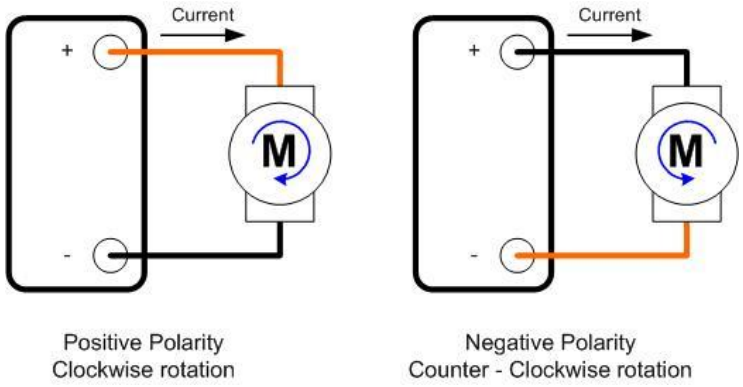


Figure 45: Current Polarization, DC Motor Rotation.

When running the first test on the motors we realized it was turning way too fast for the moving purposes of the robot, this was happening due to the stall current from the motor and the power from the RPi was given directly to it. A medium torque DC motor has a stall current of between 1.1 to 1.5 A and the RPi (as was mentioned earlier in Chapter 4.1), after powering itself is able to give 1.5 A through the 5 Volts port number 2. This was causing the motor to turn at its full capacity, almost 5000 rpm (revolutions per minute.

The connection used for this test can be observed on figures 46 and 47.

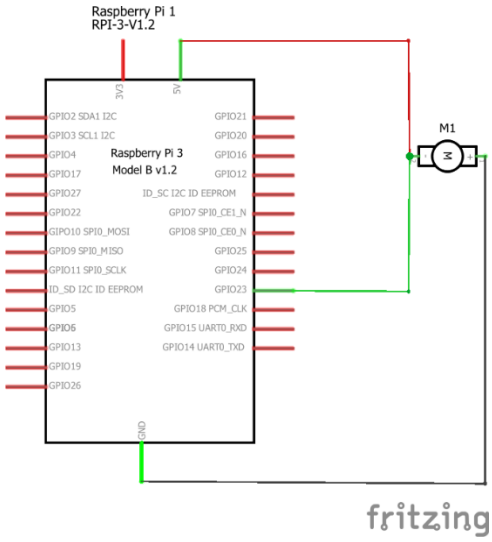


Figure 46: Schema of the Circuit Used to Turn on the DC Motor.

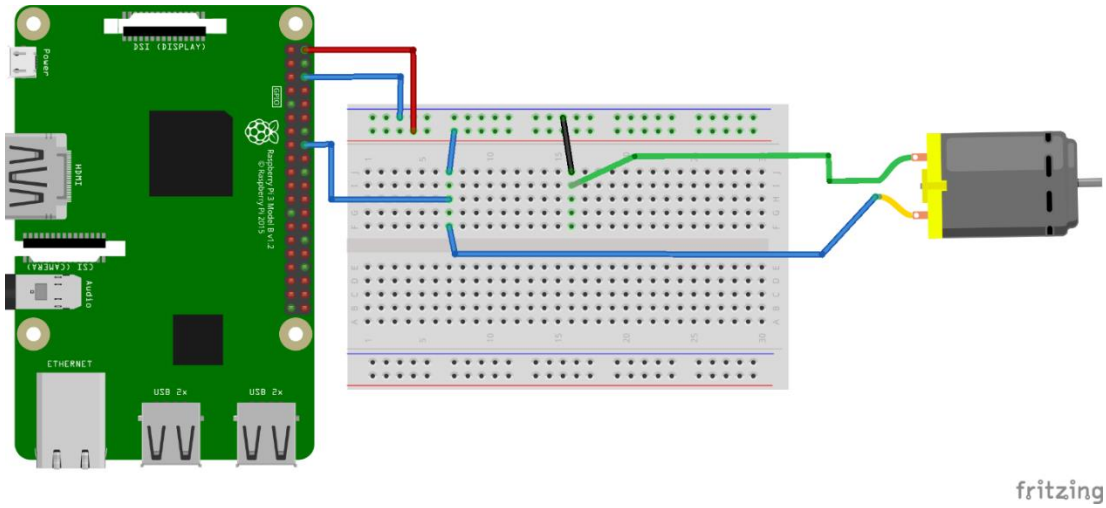


Figure 47: Represent of the Connection Used from the Raspberry Pi with the PC Motor.

6.2.2 Findings and Troubleshooting

- Connecting the DC motor directly to the RPi ports will cause the device to reboot due to the sudden surge of energy it needs to give the motor to start itself. After this results we decided to research further other examples with DC motor experiment to prevent this

problem from happening. We came across the L293D chip, which is further explained in the next experiment.

- The high velocity that the motor required is thought to be a problem in the future as we will have to reduce the speed when testing the fully assembled robot.
- We were unable to turn the rotation of the motor without disconnecting and connecting the jumper cables.

6.2.3 Phase II

After the first experiment we could understand how the DC motor work and how we could move forward to the next phase for the development of the Raspberry PI controlled robot. The elements used for this next experiment were:

- Raspberry PI 3B.
- Bread Board.
- Power Bank battery (10000mAh).
- GPIO wires
- 2 DC motors (medium torque).
- L293D chip.
- Crocodile clips.

Interaction with the L293D H-bridge chip

As the L293D chip is made to be a motor driver, it allows us to control the motors on a much accurate way, and to be able to change the rotation of the motors through code, without touching the GPIO cables (or changing manually the cables connection).

Power bank small test

With the power bank we had a doubt about whether one single power supply could or not power the raspberry PI and the motors, the first experiment we did was to leave on the

raspberry PI connected to the battery until the battery finished, after 4 hours we had to stop the experiment as we ran out of time, but the device was still running properly and the battery was still fill with more than half of the energy capacity.

Experiment with two DC motors

Having done several tests and getting familiarized on working with one DC motor, we could then start testing with two, but the connection as we started placing everything together had to be very different from the last experiment. We connected the motors, the Raspberry PI and the external power supply, all through the L293D chip which allows to enable and disable the current flow of the motors by controlling it with code from the computer. The connection used for this test can be observed on figures 48 and 49.

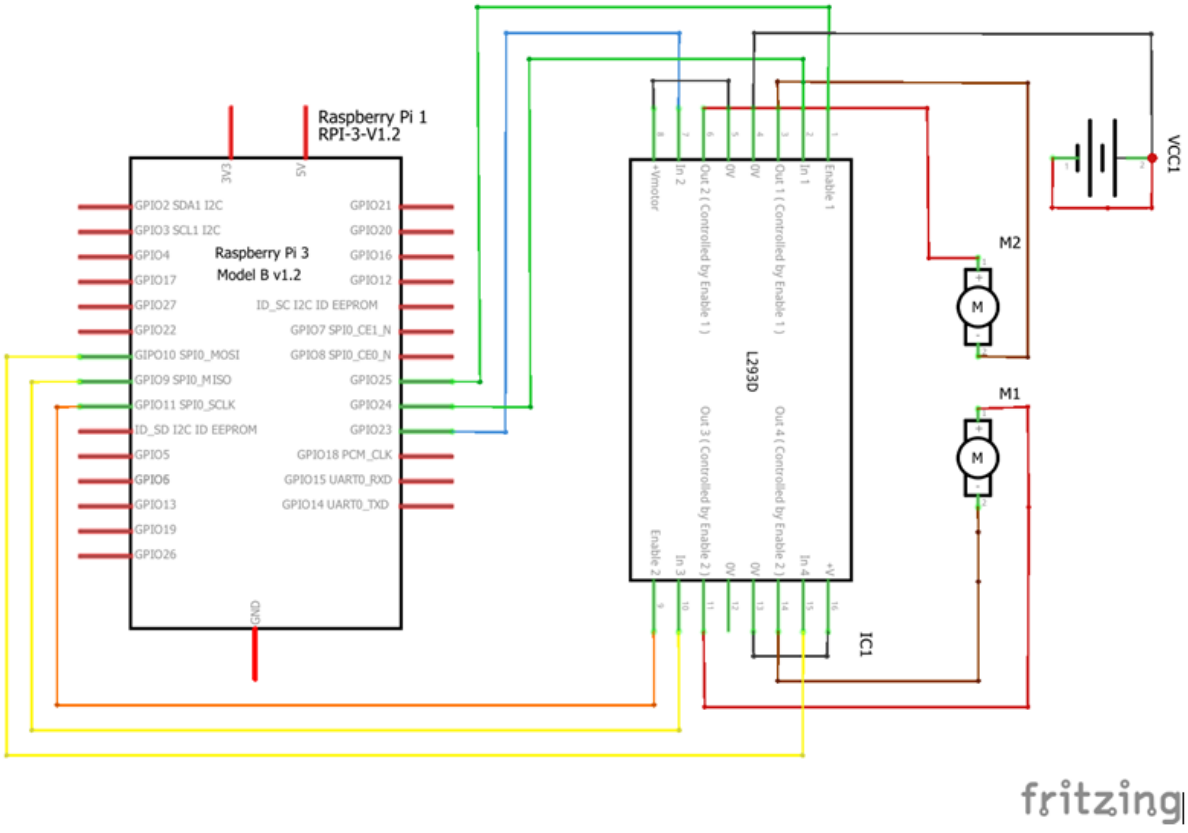


Figure 48: Schema of the Circuit Used, Shown all Devices Connected Through the L293D Chip.

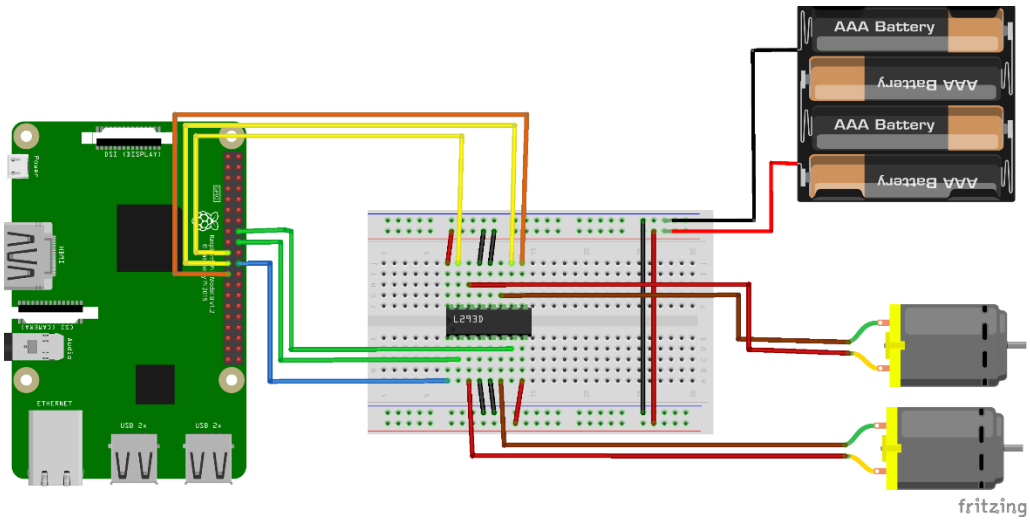


Figure 49: Representation of the Connection Using Two DC Motors.

6.2.4 Findings and Troubleshooting

While researching we came across with: the power bank used for the experiment could hold 10000mAh, the Raspberry PI needs 700mA to 1000mA to function properly, and the DC motors need 350mA to run. While doing the small experiment with the power bank we got the equation to calculate the battery life of the power bank:

- (1000[Raspberry PI] + 350[motor 1] + 350[motor 2]) mA=1700mA [required to run the computer plus the motors].
- 10000mAh [power supply capacity] / 1700mA = 5.9 hours, approximation of the battery life from the power bank.

When running the first try, nothing could work, while running the code the motors will not run at any direction, we try the motors directly why the battery and they run, which make us think about that the chip could be broken (from delivery mismanage), lots of time was focused on troubleshoot this problem, in the end was the chip, which was place wrongly , had one of the pins bended and was not making contact with the breadboard, simply we straight the pin a connect correctly, afterwards the code and connections was working as planned.

L293D chip worked as imagine, when enabling the sides and running the code the motors could turn according to the code settings, when disabling the “enable pins” the motors stopped

and when running the code with different setting (to change the rotation of the motors), the motors could stop (if running) and turn backwards.

6.3 Experiment 3: RaspBot Initial Prototype

Now that we have configured the Raspberry Pi, know how it functions and have tested the motors we decided to put something together. During this stage we focused our efforts on building a prototype and testing the RaspBot for its very first time based on our first mechanical design. As this stage is just about testing if we can get any movement from the RaspBot, we decided to use the python scripts that would be executed through Remote Desktop protocol. The elements used during this stage were:

- 1 LEGO 10692 Classic
- 1 LEGO 10701 Classic Blue Baseplate
- 2 Medium torque DC motors
- Tie cables
- 1Swivel wheel
- 2 Wheels
- 12 5mm bolt
- 12 5mm nut
- GPIO jumpers
- 2 copper clips

6.3.1 RaspBot Motor Testing: Medium Torque DC Motors

Before testing could commence, the application that gave us the possibility of controlling RaspBot remotely was tested. This initial prototype was set up to ensure that all the physical components were working correctly and in a case that we came into any issue, we would know that it was due to a problem in the application instead of troubleshooting where the problem was.

To develop this experiment, we implemented a python script that would execute the RaspBot's forward movement for a few seconds. Before we put the RaspBot on a surface, we

tested it while it was suspended in the air to check that everything was running, the test was successful.

Once we established everything was functioning correctly, we put RaspBot on a carpeted surface and ran the script. We observed the motors functioning but the RaspBot was not moving. We made a double check and tried again but nothing happened. We then did the following modifications in the python code as:

- Increased the speed to the motors as we thought the speed was too slow (wrong theory).
- Lowered the speed to try get a higher torque and beat the RaspBot's inertia. We were on the right track with this theory because after some researches we realized that the torque generated for our motors is capable to move 20g when we needed to move at least 300g. Beside that we did not realize that the only way to modify the torque is with the use of gears, modification that was not possible to implement with the elements that we had.

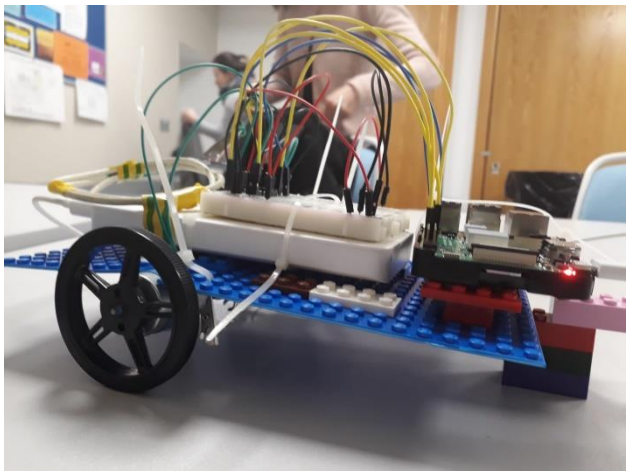


Figure 50: RaspBot During Motion Testing.

After this unexpected result and some analysis, including the mechanical design, we did some structural changes and we decided to buy a new pair of motors with a higher torque of

about 90 RPM. We also changed the wheels that now have power to move at least 300gm compared to the old ones that can move a maximum of 20gm.

6.3.2 RaspBot Motor Testing: Right Angle Gear Motor Testing

Before moving on with the development with these two new motors we decided to test it as we did with the medium torque motors on Experiment 1. Firstly, by connecting them directly to power and then connecting the other end to the breadboard and running one of the python scripts used to test the motors.

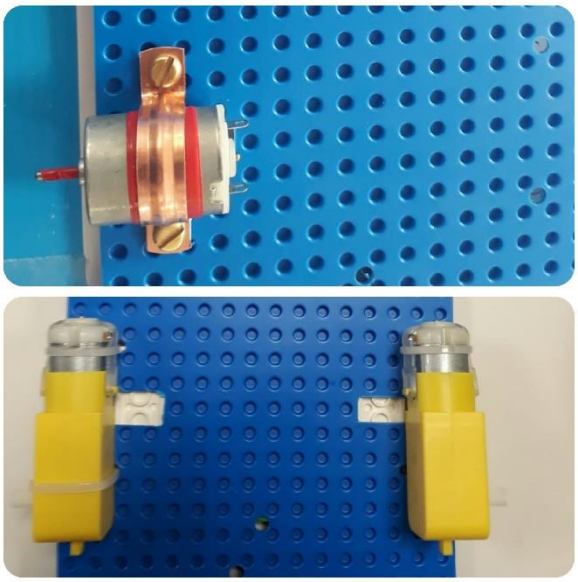


Figure 51: At the top, Medium Torque Motor, at the Bottom, Right Gear Motor.

Once we complete the isolated tests using the RPi with L293D chip, we proceeded to check if they were able to move the RaspBot. We figured out a place to install the new motors and we ran the same python script that was used with the medium torque motors.

This test generated a great result because after the previous experience that helped us to improve our initial prototype as well as other factors that we did not consider at the beginning. The initial script used had predetermined movements, moving the RaspBot forward for a certain amount of time and then turning left and then finally to stop. In this test we observed that we needed to physically adjust the motors to keep the wheels aligned.

We produced a successful test and modified the initial script by adding more movements per number of seconds. The movements done by RaspBot on this second script were:

- Move forward
- Spin 180 degrees to the left
- Move forward
- Move backward
- Spin 400 degrees to the right
- Move forward
- Stop

A great feeling overwhelmed all the members of Caffeine after this experiment because it meant that we were much closer in reaching our main objective for this fascinating project.

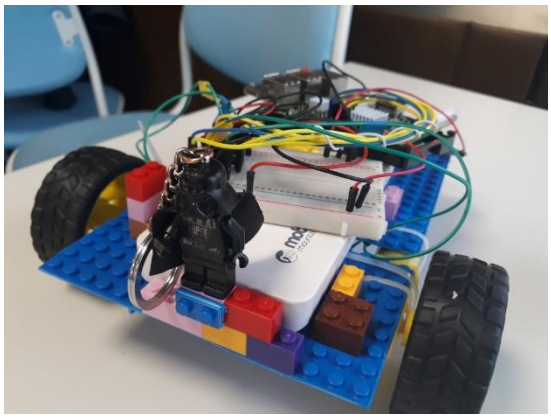


Figure 52: RaspBot Prototype With Tie to Fix the Motors.

6.3.3 Findings and Troubleshooting

As result of this experiment we learnt the following:

- Flexion is present due to long dimensions between supporting points of the RaspBot platform and its weight. We solved it out by reducing its size, taking out some elements and applying a chassis structure that provided more strength.
- Motors and RPi can work using only one battery bank, we did not realize this before because in our initial test with the motors, we did not test it with the chip. While running the motors we could find out that it was possible to allow it to work with one battery bank without generating problems to the RPi
- We labelled the cables to avoid confusion with the connection of the cables to the motors because depending on this a factor, the RPi could interpret a different function from what the script is actually telling it to do.
- We discovered that the motors that we were using originally are used for speed and are not capable of providing enough torque to move elements with more than 20 g of weight.
- It is important take under consideration weight and function before choosing some elements that carry out important functionalities (such as the motors in our case).

6.4 Experiment 4: Establishing Communication Between the Web Application and Raspberry Pi

This experiment will explain how the initial movement of the Raspbot was developed as well as how the server would process the user's interaction with the control system.

Through the beginning phases of testing the breadboard with the raspberry pi we used LED diodes and a python library called RPi.GPIO. We created basic scripts that would turn on the diodes for a short amount of time and then turn them back off (explained in the previous experiments). Once this basic connectivity was established we then replaced the LED diodes with DC motors.

We had written a script (as can be partially seen below) that would:

- start the motors and turn them in one direction for a few seconds
- stop the motors
- start the motors and turn them in the opposite direction for a few seconds
- stop the motors

After the testing of the motors were successful we decided to break that code into different parts based on the basic movements of moving forward, backward, turning left and right. As there are only 4 movements for the RaspBot, we created 4 different python scripts that would turn on the correct pins. The movement scripts were kept in the same project folder as the server.js (application) so that the server would be able to execute the scripts.

Figure 53: Left, Original Project Folder Structure. Right, MoveF.py.

The server.js was set up using express that would run it on a local host on port 8080. When a user connected to it from the local network it would display a basic control system in the index.html page. The index.html had 4 different buttons for the movement of the RaspBot, as can be seen below.

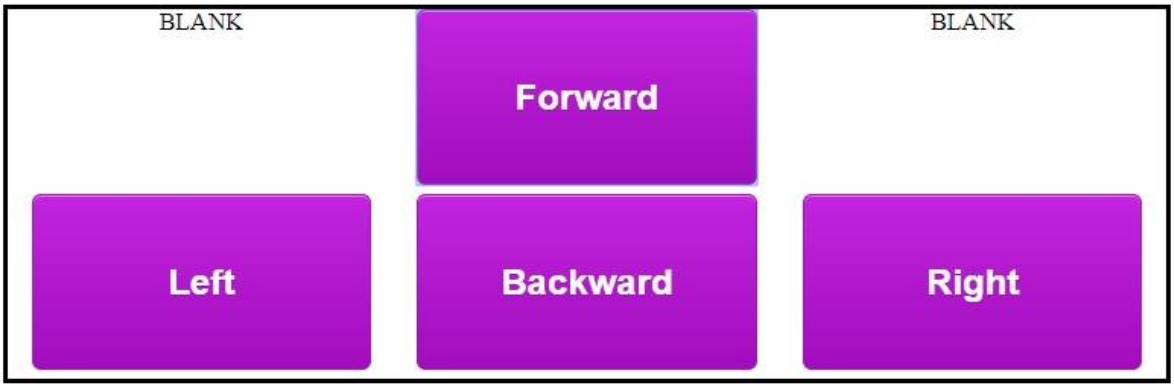


Figure 54: Basic Control Pane in the index.html Page.

Each button was wrapped in a <form> tag to allow the user to request the movement of the RaspBot. The action of the request would either be "/f", "/l", "b", "/r" depending on the button that was pressed.

```
<form method="GET" action="/f">
  <div class = "buttonArea">
      <button id = "forward" class = "buttonStyle">Forward</button>
  </div>
</form>
```

Figure 55: Initial <form> Tag to Handle Send Requests.

After the button was clicked and the request was sent to the server, the application would have endpoints set up to handle the request and perform a method which used PythonShell library to execute scripts accordingly and then send back the index.html file to reload the control panel for the user.

```
app.get("/f", function (req, res) {
    try{
        PythonShell.run('./src/moveF.py', undefined, function(err){
            if(err)throw err;
            console.log('moved forward...');
        });
    }catch(error){
        res.sendFile(__dirname+'/public/index.html');
    }
    res.sendFile(__dirname + '/public/index.html');
});
```

Figure 56: Endpoint Used to Handle "/f" Request.

The way that the scripts were written and the way the request were handled by the Raspberry Pi made the interaction between the user and the RaspBot much slower than we wanted. The RaspBot would move for a few seconds and then the webpage would load again making the whole process more intensive on computing resource than was actually needed.

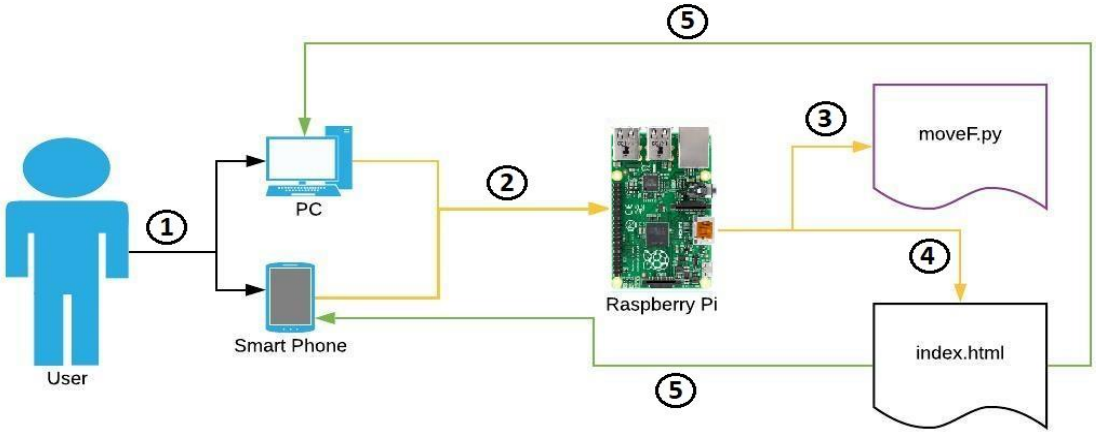


Figure 57: High Level View of First Network Architecture.

1. *A user presses the "Forward" button on their end device.*
2. *The button (wrapped in a <Form> tag) sends out a GET request to the Raspberry Pi (Server).*
3. *The server processes the request and executes the "moveF.py" file. This moves the RaspBot forward for 2 seconds and then stops.*
4. *The server sends back the index.html page to the user so they may press another button.*
5. *The index.html is displayed to the user's device awaiting for the next button to be pressed.*

This experiment was a success in getting the RaspBot to actually move when a button was pressed and opened up a way for the project to evolve. The success came in the form of the core functionality of the project being met. The downside to this experiment was that it didn't actually provide an optimal way between commination and control as well as the concept that this project incorporates (IoT) isn't achieved through this architecture.

6.5 Experiment 5: Optimizing the Communication

We were aiming at allowing the user to have a more instant responsive experience by letting them choose how long the movement should be when holding in a button. This was a follow up from the previous experiment as the movement that, was the result of that experiment, was not what we wanted to have implemented. Through research and troubleshooting we searched for a better user experience.

Our troubleshooting and research included:

- using different libraries that would allow the interaction of express and python code.
- learning about server requests and responses.
- opening different communication protocols in node.js.
- talking to experts tin the field of web application developments.

We then looked at SocketIO for a solution (based on the expert's opinions and researching into how server handles request and responses) to create a real-time connection between the client and server.

SocketIO was used to open a channel of communication between the client and the server. We imported the SocketIO library and set up some JQuery code to handle the events of when a button was pressed. This channel would remain open and listening for any communication messages that would be sent between devices. This meant that if a button was pressed, it would immediately be received at the server. Each signal that was sent from the application to the server had an identifier. A brief example can be explained in the diagram below.

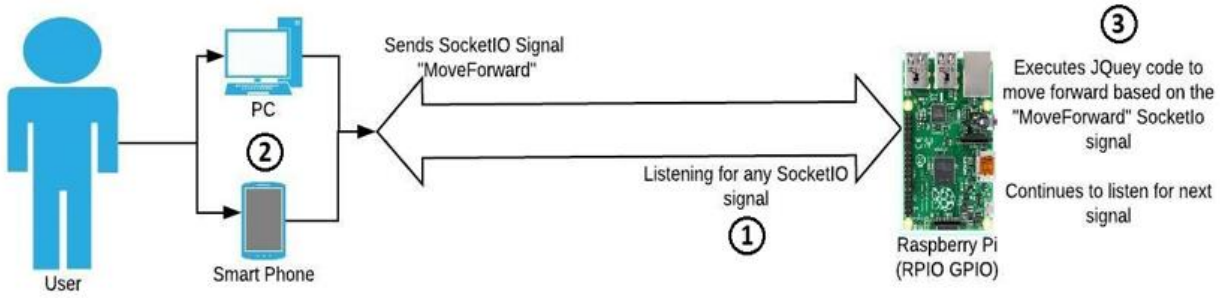


Figure 58: The Use of Socket.IO, RPI GPIO and JQuery.

1. *When the server starts, SocketIO immediately starts listening for signals.*
2. *The user is served an index.html page with a control panel and has pushed the forward button.*

   The button sends off the "MoveForward" signal to the server.
3. *The server receives the signal and executes the MoveForward() method using a JQuery function and then continues to listen to for the next signal.*

The RPI GPIO library was used to execute python code instead of executing an external python file removed another redundant step in the communication process.

This experiment resulted in a single-page web application that would processes a user's requests instantaneously without having to reload, resend any other webpages or tell the Raspberry Pi to execute a script (less computing resources). This can be seen in the image below.

6.5.1 Findings and Troubleshooting

A problem that we faced was the set up the SocketIO connection between the devices. We had successfully downloaded the library and installed the files in the correct folder location but there was no signal getting passed between the server and an end device. We went through different website and tutorials trying to figure out how this problem could be solved. We stumbled across a stack overflow article that seemed to have the exact same problem as us.

It turned out that we had created the SocketIO object correctly but we did not have the client side connected to the socket channel. This was fixed by adding in the following line about the custom.js file.

```
var socket = io.connect('http://localhost:8080');
```

6.6 Experiment 6: Final Prototype

We have achieved the main core functionality of the RaspBot using python scripts and we have built the web app able to control the RaspBot remotely, this experiment was done to define the final prototype and build it to start doing tests using the web application. This experiment is based on provide a feasible final prototype. For this experiment we required:

- 1 RaspBot buggy frame
- Raspberry Pi 3B
- 2 Right angle gear motor and wheel
- GPIO jumpers
- 4 5mm bolt and nut
- 1 Swivel wheel
- Tie cables
- Pieces of LEGO 10692 classic and 10701 classic
- Laptop
- Smart Mobile Phone
- Wi-Fi

6.6.1 Implementing Final Structure and Positioning

Now that we have defined all the elements required for RaspBot and were able to give motion too, a final establishment of the structure was needed. This structure would assign a position for every element that is part of this device. We decided to implement a double platform where the power supply (battery bank) is on the ground level, the RPi and bread board on the first level. Another improvement done was fix the motors through the use LEGO pieces tied to the Chassis to keep the alignment of the wheels. On the next picture we have the final Mechanical Design.

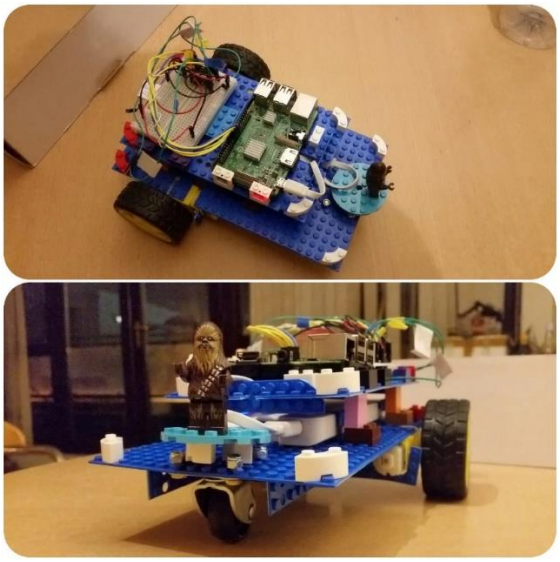


Figure 59: Raspbot With all The Elements on Position

6.6.1.1 Final Web Application

The final web application composes of a control panel that has four different buttons to control the movement of the RaspBot depending on how long the user holds in the button. There is also a speed control system, however that has not been fully implemented yet. The rest of the web application is made of sections that explain a little about the project itself as well as the members of caffeine.

6.6.1.2 Final Prototype Testing

The final prototype testing was done before the screencast was made. The RaspBot connected to a network from one of the member's phone and the user also connected to that network to be able to access the RaspBot web application through the local server. The correct adjustments were made to the project files to set up the Socket.IO channels and a track was set up to give the user a challenge in manoeuvring the RaspBot. The user would then drive the RaspBot from its parked position and then move it around the track and then park the RaspBot in its original position. During this final test, the controls proved to be very responsive and easy to use. A laptop was used for the first beginning part of the track to control the RaspBot and a mobile phone was used for the second part of the track. Both worked very well but there were some noticeable differences between the two as can be seen in the Final Results Chapter.

6.6.1.3 Finding and Troubleshooting

During the development of this last experiment we found out:

- The final mechanical design provides more organization and a better distribution of the elements adding value to the functionality and esthetical design.

7 Results

This chapter will explain what objectives of the project were met or weren't met as well as list out what was a success or not. It will outline whatever failings Caffeine came across and why a specific component was chosen over another one as well as what was learnt.

7.1 Hardware Component Findings

The RaspBot development gave us a great experience starting from gathering technological knowledge, going through team work experiences and finishing with adrenaline to meet deadlines. There were many challenges presented on the way to reach the core functionalities and it was a rollercoaster of emotion.

After we developed all the experiments we discovered some findings and problems. Its trouble shootings and detours on our way trying to reach all the objectives proposed one year ago. The following lines will highlight the most important facts that we went through to reach each objective.

7.1.1 Learning About the Electrical Components

RaspBot is built from different electrical components which we have not been in contact with before and for that reason we had to do some research about them. We had to learn about:

- LEDs to be able to understand more about the RPi and its functionalities, we learnt about the LEDs polarization, needs of use resistors, how to connect them and as consequence we learnt the use of the RPi GPIO.
- L293D motor driver Chip, we learnt that we can use it to be able to provide the current required by a motor and control it using its pins. Also we learnt it can provide an extra security layer using its enable pins that will allow the pass of current without manage it or we can disable it and just enable when we need to do it.

- Medium torque DC Motors, they are not suitable to provide movement to heavy elements.
- Right angle gear motors, suitable to move elements due to its high torque.

7.1.2 Raspbian and Python

The Raspberry Pi has Raspbian as its default operating System, because it's not as common as Windows other Linux Oss, we had to learn how to use it. Its configuration is based from Linux. We learnt how to set it up which was mandatory.

In relation to Python, we had to learn the syntax and how to create and run scripts. At the end we did not use Python as communicational language for RaspBot but it allows us to test the device on its initial stages.

7.1.3 Code Implementation

This objective was achieved successfully. We were able to create code using Python from scratch to test the electrical elements in an isolated environment as well as an assembled product (RaspBot). We found out details related to RPi as libraries, L293D interaction, GPIO nomenclature (BCM or board), GPIO uses and establishing the circuitry.

We wanted a smooth interaction between the user and the RaspBot. The initial PythonScript library was used in a complicated fashion and only executed a single script at a time. RPi.GPIO allowed us to directly control the current flow on the Raspberry Pi from the web application. We learnt that when PythonShell was used, it would only execute a single python file at a time which used more processing power instead of just having the Pi control the pins directly

7.1.4 Assembly of Hardware Components Through the Design Plan

The RaspBot was successfully assembled with an impressive final design based on LEGOs, where all the components were placed in a specific, tidy position. Through the development of the project we were improving and getting new ideas and solutions day by day. As any project the times established on the designed plan at the beginning suffered some changes due to unexpected issues and modifications that provided a better final RaspBot.

7.1.5 RaspBot Web Application

The RaspBot application was created with four simple buttons: The language used was JavaScript (instead of TypeScript) and SocketIO to establish connection with RaspBot. This choice was made early in the development process to choose JavaScript over TypeScript. Initially we tried to create the project using TypeScript and React as it was what we were learning in one of the modules in our final year and it could've worked well in conjunction with this project. However, we discovered through trial and error as well as research that the libraries and interaction with the Raspberry Pi was easier to manage though JavaScript.

This objective was reached successfully, creating a functional web Application that allow us to control remotely the RaspBot as long as RaspBot and its user are on the same network.

7.1.6 Why Bootstrap was chosen:

As we wanted something that could scale well to a user's device, bootstrap offered that solution. This scaled the web application in a sleek and responsive manner to the user. We tested the different devices to see which had the better response from the RaspBot.

We concluded that the web application runs better on a mobile device as its more user friendly in pressing the buttons. The layout is easier to navigate the control system. We did not assign any keys to the buttons of the web application so a keyboard would not work. We also noticed that when the application was run from a laptop there would sometimes be a moment where the Raspbot would get stuck in the response and continue moving forward for an extra second.

7.1.7 RaspBot 4G module

The idea of implementing a 4G module was to provide autonomous internet using a mobile SIM card and establish connection with RaspBot from other devices connected to a different network.

Unfortunately, we were not able to reach this objective due to lack of time. We know that we must acquire a RPi 4G module, a SIM card from any mobile service provider and install a driver that allow the RPi to recognize the 4G module and its futures.

8 Further Development Recommendations

This chapter looks back at the process of this project and explains what can be done differently or if any other components can be added to increase the features of the project. This section is dedicated to those that see an opportunity continuing with this project and want to go forward.

Some of the ideas can be seen in the lines below:

- Provide autonomous internet using a RPi hat that allow it to get internet by itself through use of a Mobile phone SIM Card.
- Web application on the cloud, this could be helpful providing access to the RaspBot from anywhere in the world using the cloud.
- Use of a sensor to add functionalities as Artificial Intelligence, Mapping, collision avoidance and others.
- Use of GPS using a GPS hat or GPS by SIM Card that allow RaspBot being a tracking and mapping device.
- Camera that allows controlling remotely, mapping, discovering and others.

Conclusion

A lot was learnt during the process of this project and creating the proposal. The team work was most impressive from each member as they managed their time effectively to enable them to be available for meetings every Wednesday of each week. These meeting would usually span over long periods of hours with each minute being as productive as the next. We, Caffeine, adapted our project in a way that all members' ideas were taken into consideration and implemented making this project a truly remarkable team working experience. During the process we shared tasks weekly to speed up and perform the entire research and development process in relation to the time allocated to complete and present the project. Each member took it upon themselves to take responsibility in delivering the task that were assigned to. United through the teamwork that we established, we also found the right opportunity to apply the technologies previously learned during these three years as Information Technology students and to merge our IT knowledge with the comprehensive research previously done to complete the projects necessities.

We enjoyed the whole journey taken to complete out prototypes with the utmost satisfaction. We did learn that time management was crucial factor and task delegation was a great strategy for minimizing workloads and completing our objectives.

We highly recommend to take the time to first research, then set a core objective and goals. Make sure to design an activity diagram or an activity plan to be able to see the progress and all work that's left to be done. Create minutes for each meeting where each task that is assigned to each member can be documented as well as to document what was done in each meeting, and what will be achieved in the next. Do a weekly scrum meeting where each member will present their findings and completion of tasks as well as discuss what they will be trying to work on or achieve for the week or current meeting.

And finally, keep in constant contact with the supervisor who is there to guide and encourage you to make the what seems impossible to come true and who will lead you to improve your performance through constructive criticism.

References

(n.d.). Retrieved April 10, 2019, from https://www.chegg.com/homework-help/definitions/moment-5

(n.d.). Retrieved March 01, 2019, from https://raw.githubusercontent.com/socketio/socket.io-client/master/dist/socket.io.js

(n.d.). Retrieved from https://www.bing.com/videos/search?q=how to run python script from command line&&view=detail&mid=121391D938CE5F3FF266121391D938CE5F3FF266&rvsmid=8E6FE9344CF55F5965858E6FE9344CF55F596585&FORM=VDQVAP.

(n.d.). Retrieved April 8, 2019, from https://trends.builtwith.com/javascript/jQuery

Balkhi, S. (n.d.). What is CSS? How to use CSS in WordPress? Retrieved March 10, 2019, from https://www.wpbeginner.com/glossary/css/

Barrag, H. (n.d.). Breadboard Wiring. Retrieved December 7, 2018, from http://wiring.org.co/learning/tutorials/breadboard/

Buckley, I. (2018, October 01). Why GPIO Zero Is Better Than RPi.GPIO for Raspberry Pi Projects. Retrieved November 01, 2018, from https://www.makeuseof.com/tag/gpio-zero-raspberry-pi/

Budimir, M. (2017, April 28). FAQ: How does the performance of parallel and right-angle gearmotors compare? Retrieved April 15, 2019, from https://www.motioncontroltips.com/faq-performance-parallel-right-angle-gearmotors-compare/

Chandler, N. (2012, March 13). How 4G Works. Retrieved December 9, 2018, from https://electronics.howstuffworks.com/4g8.htm

Calcular el torque de los motores para un robot velocista o de sumo. (n.d.). Retrieved April 11, 2019, from https://www.sistemasorp.es/2013/03/20/calcular-el-torque-de-los-motores-para-un-robot-velocista-o-de-sumo/

Codes, I. (1970, January 11). Harnessing the power of Bootstrap in React with Reactstrap. Retrieved March 10, 2019, from https://dev.to/ilonacodes/harnessing-the-power-of-bootstrap-in-react-with-reactstrap-40ma

Coleman, A. (2016, January 18). Build a Mini Web App - Creating a webpage with HTML & CSS. Retrieved December 10, 2018, from https://selftaughtcoders.com/simple-webpage-html-css

Convertir de Amperios a Watts con esta calculadora en linea – online. (n.d). Retrieved April 9, 2019, from https://www.calculatorsconversion.com/es/convertir-amp-a-watts-esta-calculadora/

Cómo calcular el par en un motor servo. (n.d.). Retrieved April 9, 2019, from https://www.cusiritati.com/Y3b99kZBM/

Eitel, L. (2011, October 04). What are dc motors and how do definitions vary? Summary for engineers. Retrieved December 4, 2018, from https://www.motioncontroltips.com/dc-motors/

Electronica Industrial y mas. (June 20, 2016). Como calcular el torque de un motor electrico. Retrieved April 16, 2019, from https://www.bing.com/videos/search?q=como calcular potencia de un motor&&view=detail&mid=471D030D803D1CA9603E471D030D803D1CA9603E&FORM=VDRVRV

Element14. (2016, October 05). Getting Started With The Raspberry Pi 3. Retrieved March 10, 2019, from https://www.youtube.com/watch?v=gbJB3387xUw

Gewarren. (2019). Overview of Visual Studio. Retrieved March 27, 2019, from https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019

GPIO libraries on python. (n.d.). Retrieved March 01, 2019, from https://search.yahoo.com/search?fr=mcafee&type=E211US105G91208&p=GPIO libraries on python

Gpiozero¶. (n.d.). Retrieved December 7, 2018, from https://gpiozero.readthedocs.io/en/stable/

HUAWEI E3372 Dongles Specs and Features | HUAWEI Global. (2018, December 07). Retrieved December 6, 2018, from https://consumer.huawei.com/en/mobile-broadband/e3372/specs/

How Basecamp works. (n.d.). Retrieved from https://basecamp.com/how-it-works

Instructables. (2017, November 11). Breadboards for Beginners. Retrieved December 8, 2018, from https://www.instructables.com/id/Breadboards-for-Beginners

Instructables. (2017, September 21). DC Motor Control with Raspberry Pi and L293D. Retrieved December 5, 2018, from https://www.instructables.com/id/DC-Motor-Control-With-Raspberry-Pi-and-L293D/

Introducing the Raspberry Pi 3. (2016, February 29). Retrieved December 6, 2018, from https://hackaday.com/2016/02/28/introducing-the-raspberry-pi-3

JavaScript. (n.d.). Retrieved December 11, 2018, from https://developer.mozilla.org/bm/docs/Web/JavaScript

Jones, O. (2018). Anatomical Terms of Movement. Retrieved April 22, 2019, from https://teachmeanatomy.info/the-basics/anatomical-terminology/terms-of-movement/

Kleback, M. (2014, February 28). Raspberry Pi GPIO Pins and Python | Make. Retrieved October 01, 2018, from https://makezine.com/projects/tutorial-raspberry-pi-gpio-pins-and-python/

LED polarization. (n.d.). Retrieved February 14, 2019, from https://images.search.yahoo.com/search/images?p=LED polarisation&fr=mcafee&imgurl=http://owlcircuits.com/tutorials/led_example.png#id=10&iurl=https://www.tips.modularparts.net/wp-content/uploads/2017/12/led-polarity-montage-modularparts.jpg&action=close

Miramsmirams 2. (n.d.). What is the difference between BOARD and BCM for GPIO pin numbering? Retrieved October 03, 2018, from https://raspberrypi.stackexchange.com/questions/12966/what-is-the-difference-between-board-and-bcm-for-gpio-pin-numbering

Morgan, J. (2017, April 20). A Simple Explanation Of 'The Internet Of Things'. Retrieved April 20, 2019, from https://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/

Notes, E. (n.d.). What is a Power Bank - portable charger. Retrieved December 11, 2018, from https://www.electronics-notes.com/articles/equipment-items-gadgets/powerbank/what-is-a-powerbank-battery-store.php

Pythonprogramming.net. (n.d.). User Control Remote Control Raspberry Pi Car. Retrieved March 13, 2019, from https://pythonprogramming.net/user-control-raspberry-pi-car/

PyTutorials. (2017, August 26). Simulate Key Presses in Python. Retrieved April 22, 2019, from https://www.youtube.com/watch?v=DTnz8wA6wpw

Quick Links. (n.d.). Retrieved November 28, 2018, from https://www.raspbian.org/

RakeshRon. (2018, April 20). L293D Motor Driver IC. Retrieved December 4, 2018, from https://www.rakeshmondal.info/L293D-Motor-Driver

RaspberryPiFoundantion. (n.d.). Build a robot buggy. Retrieved April 11, 2019, from https://projects.raspberrypi.org/en/projects/build-a-buggy

Right Angle Geared Hobby Motor. (n.d.). Retrieved April 15, 2019, from https://irishelectronics.ie/epages/950018241.sf/en_IE/?ObjectID=3438866

RPi.GPIO. (n.d.). Retrieved April 8, 2019, from https://pypi.org/project/RPi.GPIO/

Rpi-gpio. (n.d.). Retrieved April 13, 2019, from https://www.npmjs.com/package/rpi-gpio

Rouse, M. (n.d). AI (artificial intelligence). Retrieve from https://searchenterpriseai.techtarget.com/definition/AI-Artificial-Intelligence

Taniarascia. (n.d.). What is Bootstrap and How Do I Use It? Retrieved April 09, 2019, from https://www.taniarascia.com/what-is-bootstrap-and-how-do-i-use-it/

Turning on an LED with your Raspberry Pi's GPIO Pins. (n.d.). Retrieved February 02, 2018, from https://thepihut.com/blogs/raspberry-pi-tutorials/27968772-turning-on-an-led-with-your-raspberry-pis-gpio-pins

Tutorialspoint.com. (n.d.). Python time sleep Method. Retrieved November 8, 2018, from https://www.tutorialspoint.com/python/time_sleep.htm.

Veen, J. (1997, April 28). A Brief History of HTML. Retrieved April 8, 2019, from https://www.wired.com/1997/04/a-brief-history-of-html/

Tutorialspoint.com. (n.d.). Node.js Introduction. Retrieved April 6, 2019, from https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm

Unknown. (2017, February 25). What is Socket IO ? -Introduction. Retrieved April 8, 2019, from http://socketio.blogspot.com/2017/02/what-is-socket-io.html

What is a Universal Serial Bus (USB)? - Definition from Techopedia. (n.d.). Retrieved April 16, 2019, from https://www.techopedia.com/definition/2320/universal-serial-bus-usb

What is AI (artificial intelligence)? - Definition from WhatIs.com. (n.d.). Retrieved May 08, 2019, from https://searchenterpriseai.techtarget.com/definition/AI-Artificial-Intelligence

What is Bootstrap: A Beginners Guide. (n.d.). Retrieved April 09, 2019, from https://careerfoundry.com/en/blog/web-development/what-is-bootstrap-a-beginners-guide/

What is JavaScript? (n.d.). Retrieved March 10, 2019, from https://www.w3schools.com/whatis/whatis_js.asp

What Is IoT? - A Simple Explanation of the Internet of Things. (2019, March 13). Retrieved April 15, 2019, from https://www.iotforall.com/what-is-iot-simple-explanation/

Quackit. (2019). What is Bootstrap? Retrieved April 8, 2019, from https://www.quackit.com/bootstrap/bootstrap_4/tutorial/what_is_bootstrap.cfm

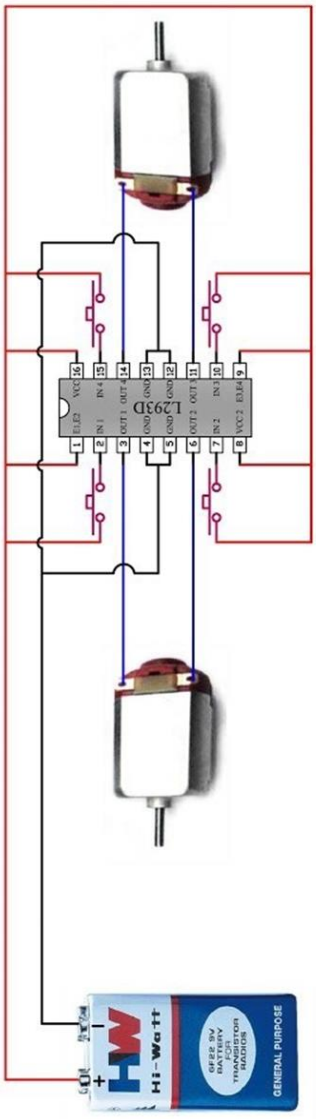
What is Linux? (n.d.). Retrieved March 28, 2019, from https://www.linux.com/what-is-linux

What is Python? Executive Summary. (n.d.). Retrieved December 5, 2018, from https://www.python.org/doc/essays/blurb/

Wiring Connection of Direct Current (DC) Motor. (n.d.). Retrieved December 6, 2018, from http://ijyam.blogspot.com/2013/11/wiring-connection-of-direct-current-dc.html

Appendices

Appendix A. Chip L293D Diagram



Appendix B. Caffeine Minutes

Meeting Minutes - [Caffeine]

Location: 9th Degree Café

Date: 10 October 2018

Time: 12:27 – 15:45

Scribe: Eric Michel

Attendance:

- Jhon Loaiza
- Vilmarys Salgado
- Eric Michel
- Nancy Aguilera
- Julio Lopez

Agenda Topics (Goals/ Tasks):

Explain the project (importance/ core objectives)

Presenter: Eric Michel **Time:** 2 weeks

We will set out to explain why we have chosen to do this particular project. This will include a deep dive on smart homes, remote access technologies, raspberry pi's and how to assemble the mechanical parts needed. We aim to explain the core fundamentals of the project and its objectives.

Research Raspberry Pi Drone Projects:

Presenter: Eric Michel **Time:** 2 hours

In this task we will observe, as a team, the specifications of the Raspberry Pi 3 as well as its previous versions. We will also observe other Raspberry Pi powered drones and document any findings on the basic process of building the robot and setting up the Raspberry Pi. This includes the formatting and loading the Raspberry Pi OS, getting an introduction of Python and Linux, construction of the bot, power requirements and controlling the bot.

Conclusion:

Explain the project (importance/ core objectives)

Not completed

Research Raspberry Pi Drone Projects:

In this task we set out to achieve an in-depth knowledge of Raspberry Pi's and how they are able to function and power a DIY built robot. We can break the outcomes of this task into The Raspberry Pi itself is a fantastic device in that its form factor is the size of a credit card yet it has enough processing power which is on par to some very basic desktops. Get specifications of a basic desktop Pc and compare it to the specifications of the Raspberry Pi3. The power consumption is easily managed by attaching some sort of external power supply, allowing the Pi to be extremely mobile. This expands its capabilities and use as it is now a powerful, portable and functional device. Many Raspberry Pi enthusiast use it in many different projects that have some core functionality, as its

information (private pilots) and even enabling smart homes by controlling heating, lighting, security and media settings.

Just like all new out-of-the-box PCs, the Raspberry Pi needs an Operating System in order to run any native applications or programs that a user would like to install. In order to do this one needs to format a storage device (SD) card and load an Operating System (OS) onto it so that the Raspberry Pi can boot up. Show steps of formatting an SD card and install NOOBS/ Raspbian. Once the Pi is formatted and ready to be booted to an OS (List all the different OS that can run off a Pi), functionality of the robot can be written, stored and executed.

Goals (Tasks) to Complete:

Robot Tasks:

- Weigh the equipment and document it.
- Inspect the robot's parts.
- Turn on and off the motor with the Raspberry Pi.
- Calculate the power consumption of the robot.
- Set up a remote connection to the Raspberry Pi.
- Create the necessary electronic diagrams.
- Make the motor turn forwards and backwards.
- Attach Raspberry Pi to the robot.
- Attach PiAnywhere to the Raspberry Pi.
- Set up a wireless connection to the Raspberry Pi.
- Control the robot from the wireless connection to the Raspberry Pi.
- Turn on the robot with the Raspberry Pi.
- Enable turning the robot.

Coding Tasks:

- Code the robot's drive functions.
- Code the robot's reverse functions.
- Code the robot's power on/ off functions.
- Code the robot's turning functions.
- Code the interface of the controls.
- Link the interface to the core code of the robot.
- Assign the Robot a physical address.

Research Tasks:

- Research ways to construct /assemble the robot.
- Decide how the project will be beneficial.
- Get information from previous Raspberry Pi projects.
- Learn about the interface technologies (JavaScript) needed to develop a web application.
- Learn how to code in Python.
- List differences between the Raspberry P versions (version 1, 2, 3, 0) and specs.
- Find out which technologies are the best for our project.
- Explain the technologies used.

- Explain the project (importance / core objective).

Documentation Tasks:

- Create use case models
- Review the goals
- Create a schedule
- Collect the previous minutes to write documentation
- Figure out the budget
- Decide about cameras, GPS and censors
- Create a blue print (frame, pi attachment)
- Make a list of equipment/ tools we need to build the robot
- Create a blue print (frame, pi attachment)

Goals (Tasks) Completed:

- Research Raspberry Pi Drones Projects.

Other Notes:

The task 'Explain the project (importance/ core objectives)' will extend into the next meeting as each member will perform the research on their own time.

A discussion of focus, specifically in the tasks that we set out in each meeting, is needed amongst the group so that we can move forward in this project together and stay on the same page with each other.

Meeting Minutes - [Caffeine]

Location: CCT College

Date: 28 November 2018

Time: 13.32

Scribe: Eric Michel

Attendance:

- Jhon Loaiza
- Vilmarys Salgado
- Eric Michel
- Nancy Aguilera
- Julio Lopez

Agenda (Goals/ Tasks):

Goal	Completed
Control the motors with the Raspberry Pi (on/ off)	✓
Separate Raspberry Pi and motor's power supply	✓
Test the LC93D chip's functionality	✓

Conclusion:

Control the motors with the Raspberry Pi (on/ off)

Our goal for this week was to try and get the motors to run independently of the Pi from a separate power source and then have the Pi control the motors function with an on/ off switch through a simple python program. We found some very helpful diagrams and explanations online that explained the process on how to do this with the help of the L293D chip, the breadboard and a separate power supply. We followed the diagrams as per their instructions, however we could not get the motors to even turn on. We then started troubleshooting different possibilities as to why it was not working. We figured it could either be:

- The power supply not supplying enough power: voltage and current
- The code not working properly with the python program
- The configuration of all the components in the breadboard

We began testing different ways of connecting the motor to the Pi but without success. Our troubleshooting list was coming to an end which highlighted one other possibility. The L293D chip. We then set a new goal of testing to see if the L293D chip was functional or not in hopes to achieve our initial goal.

Test the L293D chip's functionality

We followed a number of diagrams online that would test to see of the L293D chip was functional or not. However, none of them seemed to work. We were begging to feel defeated as the end of the day was approaching and still had not figured out why the connection wasn't working. Out of a stroke of luck, one of us removed the chip out from the breadboard and in doing so, found that the chip's pins had been bent. We figured it was bent from the initial plugging-in of the chip when we began testing the connection. The was most likely because when we received the chip it came out the box with the pins in an outward position, we had to forcibly bend them to a more straight-forward direction for to fit in the breadboard. It wasn't the easiest of fits for the breadboard even after the pins had been straightened and needed to be forced in a certain amount. Upon that forced insertion, the pin that supplied the 5v power current as well as the pin that enabled the switching on/ off of the motors was bent on the chip. After correcting the pins and reinserting the chip, we followed instructions to see if the chip was functional. The test was done with just a single power supply and without the Pi's controlling functionality and was a success! We then quickly added the Raspberry Pi to the breadboard to see if we would be able to control the switching of the motors on and off. The second test was also a success which completed our initial goal for this week.

Separate Raspberry Pi and motor's power supply

We needed to separate the power supply between the motors and the Raspberry Pi in order to not disrupt the flow of electrical current to the Pi. This disruption would force a shutdown due to a spike or drop of electrical current when the motors were turned on. We've also researched online that the motors shouldn't be connected directly to the Raspberry Pi as it might damage the motor and, even more importantly, the Pi itself, because the Pi only can provide a maximum of 20mA and the currente required for a motor is at least 400mA. We initially tried adding as external battery one AA battery but after some research we realized that the L293D require at least 5v (one AA battery = 1.5v). Then We tried connecting 3 AA batteries to our initial tests but we were not getting any results (due to the pins being bent in the chip and voltage was not enough). We figured it was a problem of voltage and then decided to try a different solution. That solution was to add another power bank as a power supply to the motors. We figured we could achieve this by stripping down a micro-USB cable to expose the 2 voltage wires (current and ground) and then connect the end of those wires to each side of a motor. The original USB port would then be plugged in to the power bank to supply power to the motors directly. The solution worked when we tested the motors being directly connected to the power banks, however when we tried implementing this to our breadboard it didn't work (again due to the bent pins of the chip). We kept this solution though as we decided this would be the most convenient way to power the motors for this project.

Other Notes:

- Explanation of the L293D chip is needed for documentation
- Explanation of how the L293D chip was tested independently and how it was tested while connected to the Raspberry Pi and separate power supply is needed.
- Goals for next week include:
    - Get 2 of the motors to work with the Raspberry Pi (on/ off)
    - Get the motors to turn independently
    - Get the motors to turn in reverse
    - Get the motors to turn in a specific speed

Meeting Minutes - [Caffeine]

Location: CCT College Dublin

Date: 12 December 2018

Time: 1700 hrs.

Scribe: Julio Cesar Lopez

Attendance:

- Jhon Loaiza
- Vilmarys Salgado
- Eric Michel
- Nancy Aguilera
- Julio Lopez

Agenda (Goals/ Tasks):

Goal	Completed
Work on Group Deliverable A & B from the Group Project Handbook	✓
Discuss basic design for the Robot	
Learn functionality L293D chip	✓
Get the motors to turn in different speed	✓
Detailed understanding of the diagram Connections to turn on the motors	✓
Understanding of the code used to turn on the motors and modify speed	✓
Decide goals for next meeting	✓

Conclusion:

WORK ON GROUP DELIVERABLE A & B FROM THE GROUP PROJECT HANDBOOK

--- Completed

DISCUSS BASIC DESIGN FOR THE ROBOT

--- Not Completed

LEARN FUNCTIONALITY OF L293D CHIP

Conclusion info

The L293D is a chip used to control motors, today we learnt the function of each pin that this chip has. During this meeting we tested the different pins of it and we realized that the Enable pin is ON by default and can be used as security switch to control when switch ON or OFF the inputs and

outputs of the section. The Chip has four sections and two Enable pins that act over two sections each.

GET THE MOTORS TO TURN IN DIFFERENT SPEED

The motors were capable to turn in different speed throw the application of Pulse Width Modulation (PWM) that allow us to decide the frequency of work of the motor through activating and switching off the motor to control the speed of spinning. After this we realized that one our future goals is find out the different speed to be controlled in the robot due to the weight of the prototype.

DETAILED UNDERSTANDING OF THE DIAGRAM CONNECTIONS TO TURN ON THE MOTORS

Once that we got the motors working and understood how the chip works we decided to analyze all the connections done to make it work and test a few changes, letting know everybody in the group how and why the connections are on that way. Understand the different pins that a raspberry pi has as ground pins, 5v pins, 3v pins and GPIO. Also the use of a bread board.

UNDERSTANDING OF THE CODE USED TO TURN ON THE MOTORS AND MODIFY SPEED

We analyzed the code and explain how it works to those teammates that have not worked to much with it. Differences on the numeration due to the use of BOARD or BCM library to identify the pins. Use of functions as sleep, GPIO.OUT, GPIO.IN and others.

DECIDE GOALS FOR NEXT MEETING

--- Not Completed

- Continuous discussion about Robot case.
- Digitalize the connections diagram.
- Research about autonomous Internet.
- Use of button to turn on and of the robot.

Other Notes:

- Other notes here
- Other notes here
- Other notes here

Meeting Minutes - [Caffeine]

Location: CCT - Westmoreland Room

Date: 20 February 2019

Time: 12.30

Scribe: Julio Lopez

Attendance:

- Jhon Loaiza
- Vilmarys Salgado
- Eric Michel
- Nancy Aguilera
- Julio Lopez

Agenda (Goals/ Tasks):

Goal	Completed
Establish meeting structure and responsibilities to reach core functionality	✓
Define current issues, responsibilities and specific objectives	✓
Web App Objective: Creating UML	✓
Web App Objective: Designing initial structure	✓
Documentation Objective: Reviewing of Documentation	
Documentation Objective: Defining point of improvement	
Robot Build Objective: Building the prototype	
Robot Build Objective: Initial prototype test	✓
Robot Build Objective: Gather final components	

Conclusion:

Create Main Objectives:

Goals include:

- Create a template for the main project
- Build the Robot
- Build the Web Application
- Connect the Web Application to the Robot
- Enable the 4G network for the Robot

Establish meeting structure and responsibilities to reach the core functionality

will explain what was done the previous week, issues and what is the goal for this week, in case of need a hand from somebody there is the moment to request it. Through this methodology every one will be involved with the responsibility of each other and how is running the project.

Define current issues, responsibilities and specific objectives

During the meeting we found out the following issues and decided responsible:

- Documentation and Planning. Nancy and Vilmarys.
- Web Application Front-End, Back-End and communication between them. Eric.
- Prototype Design, RaspBot primary motion based on hardware design. Jhon and Julio.

The specific objectives decided were:

- Review of Documentation and improve it.
- Decide structure for Plan and pending activities to reach the core functionality.
- Decide structure to be implemented for the Web App.
- Decide web app languages.
- Assembly design of the prototype.
- Test motion of current prototype.

Web App Objective: Creating UML

With the idea of being focused about what we want to reach, was decided to create the UML for the project and develop all the necessities discovered during the development of the UML

Web App Objective: Designing initial structure

This part of the project is about take one structure that we think could be more convenient and start developing it. The name of initial structure is due to our lack of experience at this point we are making focuse on create something that allow us to control the RaspBot remotely after that we will try to do it through the Web application that could be using a web browser (main objective for this segment) or just connecting directly to the Raspberry being in the same network (practical and easier). At this point we have chosen:

- Programming language: Python.
- Web Application language: TypeScript, HTML, CSS, NodeJS.

Documentation Objective: Reviewing of Documentation

After the feedback related to our proposal was required to review some information that we did not provide there, review our plan and add it because we did not put it on the proposal. Also since we did the proposal until now we have done some modifications that must to document and to avoid last minutes problems now is a good moment to start with it and give a really good documentation.

Documentation Objective: Defining point of improvement

We started reviewing some weak points on the proposal to improve it, add the plan of the project with deadlines that make us able to reach the core functionality and enhance our RaspBot.

Robot Build Objective: Building the prototype

The prototype was made physically for the first time on the project, we used Legos, Two medium torque motors, two wheels, one swivel caster wheel, GPio cables, Two Battery banks, one chip to control the motors. At this point we observed the base buckled a bit due to the weight of the elements and the distance between the support points.

Robot Build Objective: Initial prototype test

The initial test was made using the computer as remote control, for the test we used teamviewer to connect the RaspBot through its IP address and run the python scripts saved under the Raspbian OS. We connect all the elements and organized on RaspBot base, we were using two battery banks, long breadboard, two motors, two wheels, one swivel caster wheel and the raspberry over the Lego base. The Device did not show motion. We tried changing the speed, surfaces and testing the motors isolated and with the RaspBot on the air. We got the conclusion that the case is too heavy and decided make some structural modifications for next week.

Robot Build Objective: Gather Final Components

Continuous requirement of material due to changes on the design and unexpected issues.

Other Notes:

- More research is needed as to how connect back-end and front-end.
- More materials are needed to construct the full body of the robot as only the base was constructed and adapted to hold the elements
- Another major goal of “Coding the functionality of the robot” is to be added to the main goals.
- A timeline is still being developed by Nancy and Vilmarys to help with time management of the different goals.
- Test RaspBot to see if the current design is able to get motion.

Meeting Minutes - [Caffeine]

Location: CCT - Westmoreland Room

Date: 20 March 2019

Time: 12.30

Scribe: Julio Lopez

Attendance:

- Jhon Loaiza
- Vilmarys Salgado
- Eric Michel
- Nancy Aguilera
- Julio Lopez

Agenda (Goals/ Tasks):

Goal	Completed
Scrum meeting	✓
Web App Objective: Research of How to connect Front-end with Back-end	
Web App Objective: Designing Back-end	
Documentation Objective: Action Plan. Activities and Deadlines	
Robot Build Objective: Install new motors and Wheels to the prototype	✓
Robot Build Objective: Reinforce Lego structure	✓
Robot Build Objective: Test current prototype	✓

Conclusion:

Create Main Objectives:

Goals include:

- Build the Robot
- Build the Web Application
- Connect the Web Application to the Robot
- Enable the 4G network for the Robot

Scrum meeting

Last Week:

- Web App: Front-end is done (initial design).
- Documentation: Developed Project plan until 50%.
- Robot Build: Redesign and test of motion.

Today:

- Web App: Continuous designing and searching structure and solutions for connection Front-End and Back-end.
- Documentation: Continuous with the Project Plan, defining activities and its deadlines.
- Robot Build: Apply modifications to RaspBot and Test its motion.

Web App Objective: Research of How to connect Front-end with Back-end

Searching information and coding to be able of create an architecture that allow us to connect the Front-end with Back-end to control remotely the RaspBot. This research will allow us to define if we use a local server, web browser or socket to make the connection.

Web App Objective: Designing Back-end

Design a back-end that allows control the RaspBot remotely, at this point we are creating just functions in python that will be called by the back-end once the user command an action. We are trying to figure it out how to keep the function running while a directional button is kept pressed.

Documentation Objective: Action Plan. Activities and Deadlines

Continuous with the organization of the Plan Activities and its deadlines. Deciding the activities required to reach the goals and giving realistic deadlines. At this point Nancy and Vilmarys have developed the 75% of the plan

Robot Build Objective: Install new motors and Wheels to the prototype

After last week when we got the conclusion that the RaspBot was too heavy, we decided to change motors and wheels due to the medium torque motors did not have enough power to move that weight. Also, we bought another kind of wheels that suit better for these motors. We installed and then it looks more robust, but we can still observe a little buckle on the Lego Base that make the wheels lose alignment.

We decided to redesign the Lego base making it stronger, we added a structure to the base, even though the base area where the motors are located still bend a little bit, we used a cable ties to sort out that issue.

Robot Build Objective: Test new prototype

After one week waiting for the new motors and wheels, we installed and test the new prototype being a total success, we were able to give motion and control it remotely through the computer using Team Viewer to stablish the connection. This motion was programmed, we will be working on the instant remote-control next week and try to do it through the web app.

Other Notes:

- More research is needed as to how connect back-end and front-end.
- Research to live control
- General review of documentation to see how is our position in relation to the final document.

Meeting Minutes - [Caffeine]

Location: CCT - Westmoreland Room

Date: 10 April 2019

Time: 12.00 – 6.00

Scribe: Vilmarys Salgado

Attendance:

- Nancy Aguilera
- Eric Michel
- Jhon Loaiza
- Julio Lopez
- Vilmarys Salgado

Agenda (Goals/ Tasks):

Goal	Completed
Scrum meeting	✓
Web App Objective: Adapting back-end for mobile phone.	
Web App Objectives: Starting with front-end	
Documentation Objective: Schema of the documentation	✓
Documentation Objective: Graphs Design of the robot prototype	
Documentation Objective: Writing down the experiments description	
Documentation Objective: Adding more technologies that were used in the new experiment.	

Conclusion:

Create Main Objectives:

Goals include:

- Build the Robot
- Build the Web Application
- Connect the Web Application to the Robot
- Enable the 4G network for the Robot

Scrum meeting

Last Week:

- Documentation Objective: Was presented a project template for the final documentation, as well chapters' descriptions and content.
- Robot prototype: Were done more adjustment to the robot prototype, as well as tighten up its components.

- Web App: Was tested the web app with the robot prototype for further improvements

Today:

- Web App Objective: Adapting back-end for mobile phone.
- Documentation Objective: Schema of the documentation presented to the whole group members.
- Documentation Objective: Graphs Design of the robot prototype to deep down the explanation.
- Documentation Objective: Writing down the experiments description, to amplify the instructions.
- Documentation Objective: Adding more technologies that were used in the new experiment to the documentation.

Web App Objective: Adapting back-end for mobile phone.

Adapting back-end for mobile phone, and also testing.
Starting with front-end of the web app. This was showed to our supervisor Amilcar Aponte in place.

Documentation Objective: Schema of the documentation

We presented the final schema of the documentation to the group members, as well we finalized with a result, also was showed to our supervisor Amilcar Aponte and, and we meet 30 min with Mr. Graham Glanville who gave us some advices related with documentation.

Documentation Objective: Graphs Design of the robot prototype

Creating and designing the graphs for the experiments descriptions in the documentation

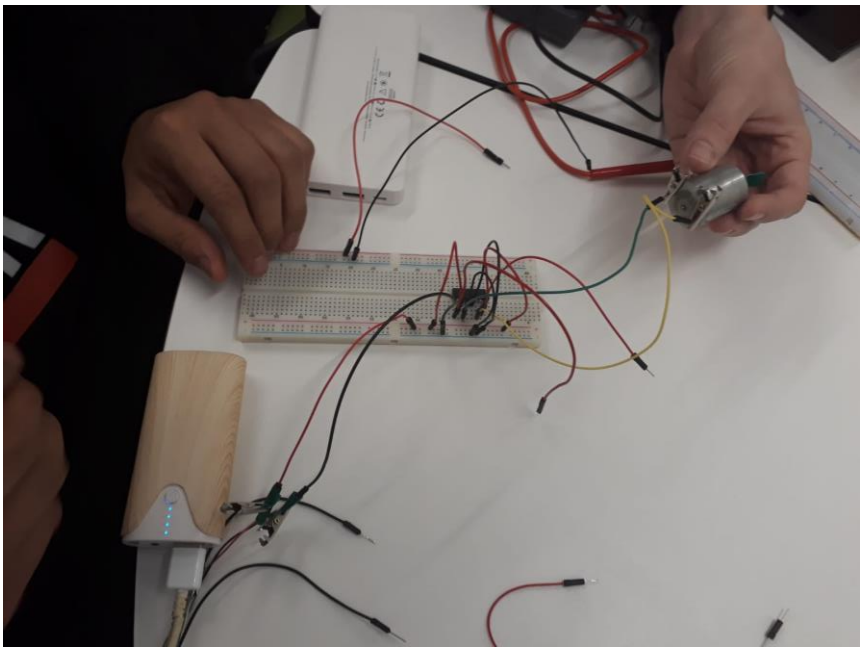
Documentation Objective: Writing down the experiments descriptions

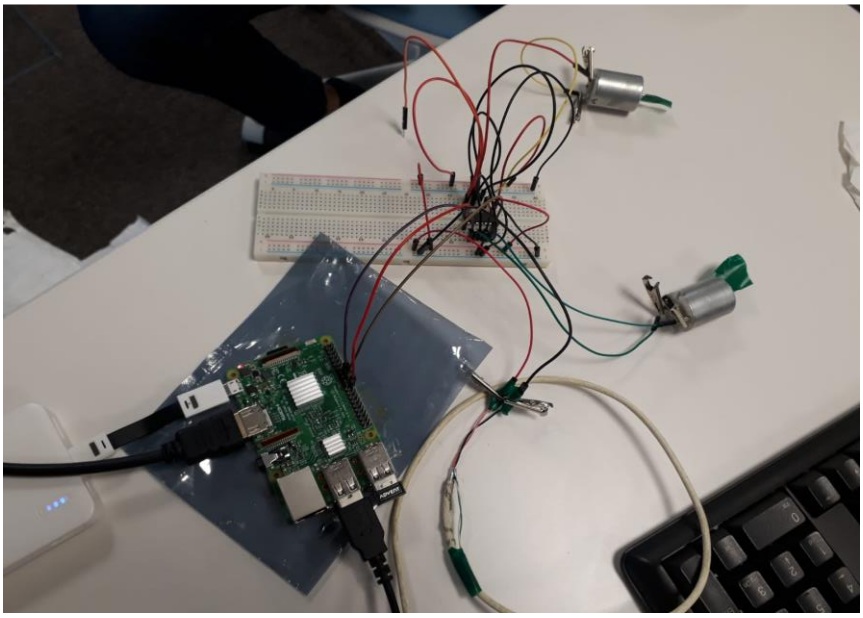
We were defining a way to explain in a proper and understandable manner how the experiments were carried out and the results obtained.

Other Notes:

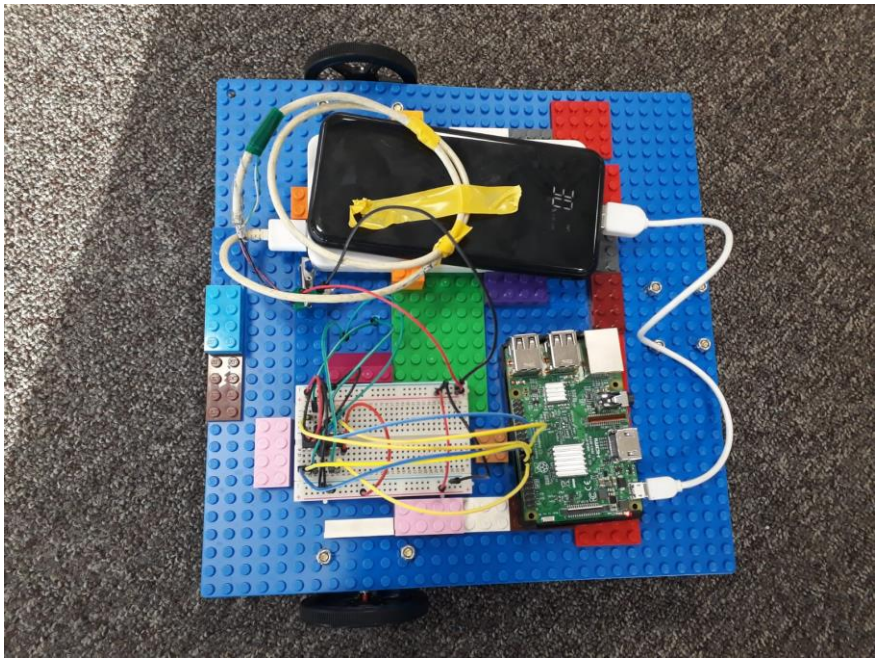
- Keep working on documentation, especially in citations and referencing.
- Keep in track with the documentation.

Appendix C. Medium Torque Testing

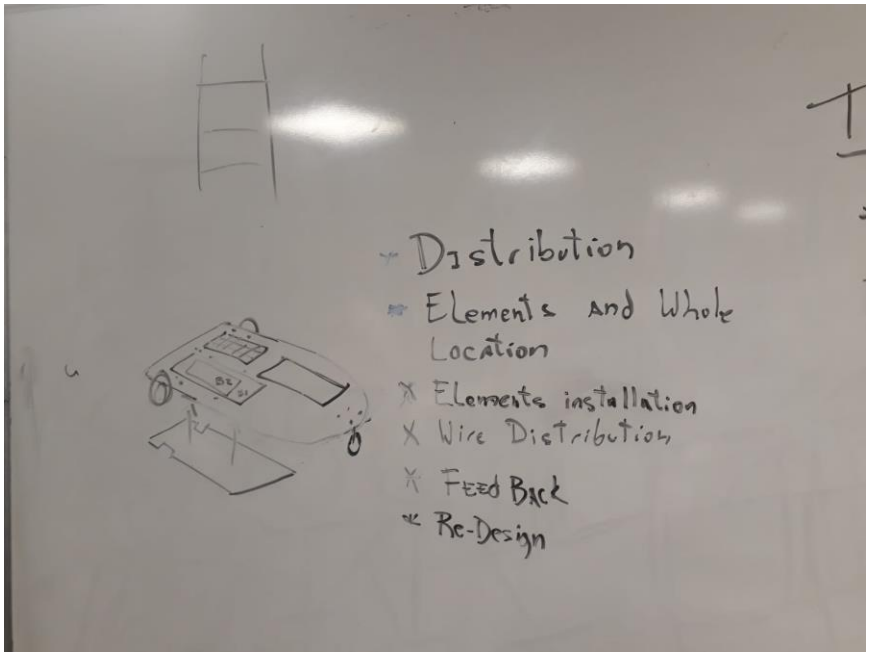




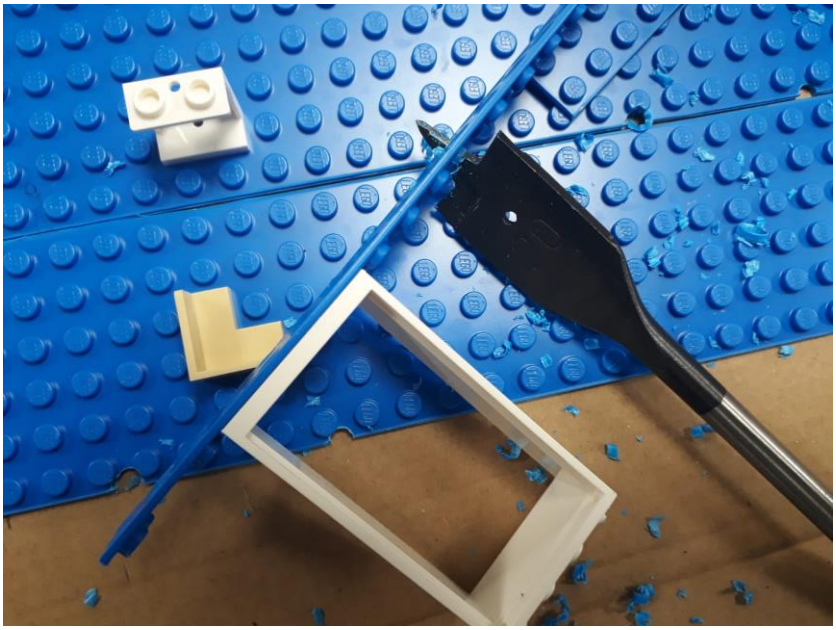
Appendix D. Initial Mechanical design



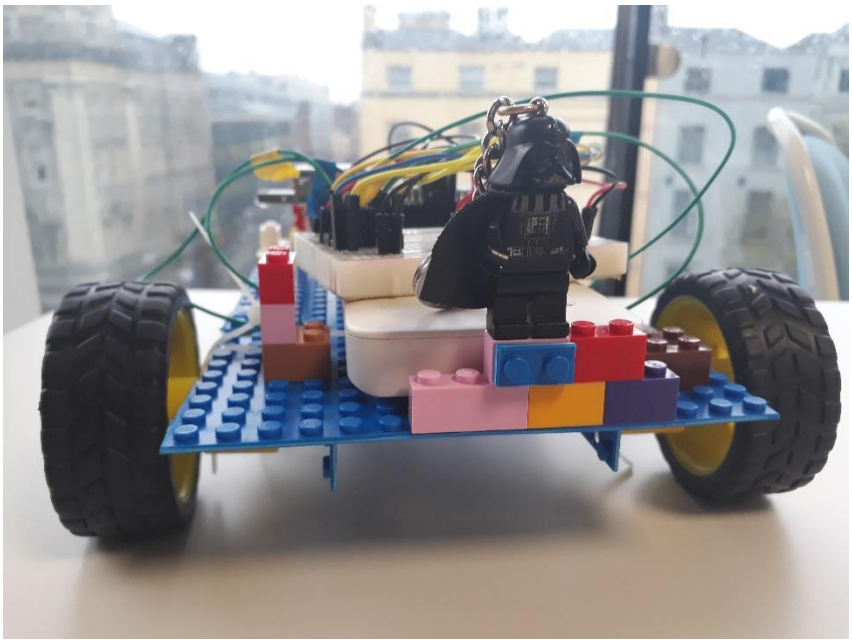
Appendix E. : Sketch of design to improve initial prototype



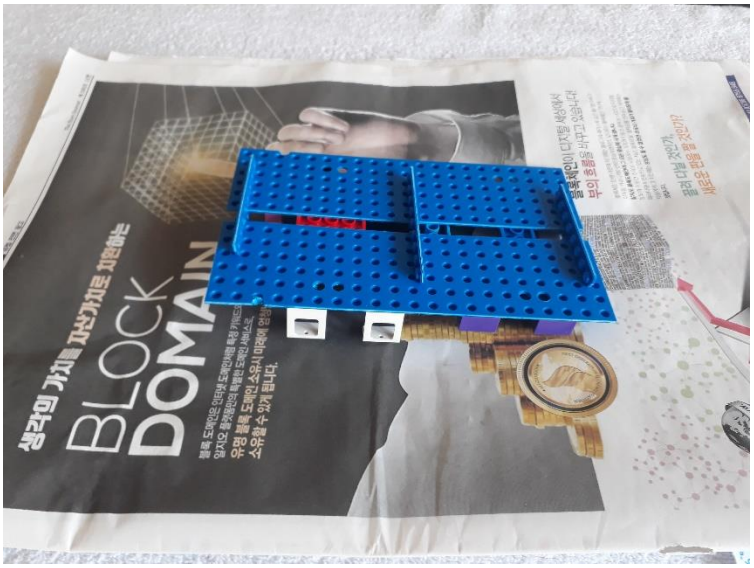
Appendix F. Building chassis



Appendix G. Mechanical design with Right angle gear wheels



Appendix H. Platform for RPi and breadboard



Appendix I. : Building final Mechanical design