Di-Wheel Robot

A Cost Effective, Stair Climbing, Tele-Operated Rescue Robot

University of Cape Town

Department of Electrical Engineering

Final Year Project

Author:

Richard Daniel Powrie

Supervisor:

Justin Pead MSc

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for a Bachelor of Science degree in Mechatronics.

14 October 2019

EXECUTIVE SUMMARY

Urban Search and Rescue environments are dangerous, filled with many obstacles to be overcome. Human and canine responders do a good job in solving these problems, but they often fall short because of their innate weaknesses. Utilising robotics to assist them overcome their weaknesses improves their quality of service, saving lives as a result.

This project began with the aim to determine if a clever epicyclic gear train can be used in a mechanical limb called a Di-Wheel to produce a low cost, stair climbing, tele-operated robot. The function of this robot would be in reconnaissance and primary search for survivors. A responder would operate it, driving the robot into a dangerous area to assess the location and find survivors.

After developing a successful mechanical design, and an electronic system to control the robot, it was found that the Di-Wheel would prove successful in a USAR environment with further development to fine tune all the elements.

The majority of the work covered is in this report, however there are extra features available in the Di-Wheel Google Drive, such as pictures and videos of testing, mathematical models and excel spreadsheets.

PLAGIARISM DECLARATION

I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.

I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this final year project report from the work(s) of other people, has been attributed and has been cited and referenced.

This project report is my own work.

I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

Signature: 

Name: Richard Daniel Powrie

Date: 14 October 2019

ACKNOWLEDGEMENTS

There were many people who have been involved in assisting the author to complete this project, and they are mentioned here along with their contributions.

Emily Dollman for encouraging me through this project, never ceasing in her love and support for me and my goals.

Justin Pead for his excellent mentoring and supervising. This project would not be what it is without his guidance.

Trevor Cloete for his assistance in the conceptual understanding of the motion of the gears.

Irshaad Dodia for assisting in tests and always encouraging me to work and improve.

Joash Naidoo for his supporting influence and humour to keep me motivated.

CONTENTS

LIST OF FIGURES

1 INTRODUCTION

1.1 SUBJECT OF AND MOTIVATION FOR REPORT

Urban Search and Rescue environments are dangerous, filled with many obstacles to be overcome. Human and canine responders do a good job in solving these problems, but they often fall short because of their innate weaknesses. Utilising robotics to assist them overcome their weaknesses improves their quality of service, saving lives as a result.

This project began with the aim to determine if a clever epicyclic gear train can be used in a mechanical limb called a Di-Wheel to produce a low cost, stair climbing, tele-operated robot. The function of this robot would be in reconnaissance and primary search for survivors. A responder would operate it, driving the robot into a dangerous area to assess the location and find survivors.

A Di-Wheel is a mechanical limb which uses internal epicyclic gearing to create a driving motion on flat surfaces, and climbing motion when a step is encountered. This process is described later in the report. It is the aim of this report to show that the Di-Wheel concepts can reduce the complexity of the robot to reduce costs. This can save lives and prevent time being wasted, as more than one robot can be employed simultaneously.

1.2 BACKGROUND TO INVESTIGATION

USAR robots have been developed for a long time, and there are many options out on the market. There are very few that are sufficiently inexpensive for responders to utilise without concern for the price of the robot that may be damaged. This is of great concern, as responders should not have to worry about losing a robot when the alternative would be the loss of a human life. This can be overcome by providing a low cost alternative to the current designs, enabling many robots to be deployed without fear of destruction, as opposed to one single robot that needs to be protected.

The Di-Wheel design was first considered in a UCT project in 2013 by Matthew Wilson. Using his work, the concept was developed further by Jordan Haskel in 2017, and Murray Buchanan in 2018. This present project aims to use their work to prove that the Di-Wheel can actually be used to climb over obstacles, with the second goal of USAR functionality.

1.3 OBJECTIVES OF THE REPORT

The objectives of this report are therefore to:

- Outline a literature review wherein the requirements of USAR robots is studied and the reports of previous Di-Wheel are considered.
- Describe the design methodology.
- Present the design process. This will be done in three phases; a more mechanical phase where the Di-Wheel is modelled, and the robot designed. After being built, an electronic control system will be created in the second phase followed by additional features in the third phase.
- Detail the results of all testing done during the design and assembly.
- Present conclusions and recommendations on the results obtained.

1.4 Limitations and Scope of Investigation

To design a small, remote-observation, light-weight, polymorphic robot that can traverse over urban building terrain using the Di-Wheel concept. It must be built with off-the-shelf components, utilize simple body materials, and be cost effective.

This project does not seek to create a fully functioning USAR robot but merely aims to determine if the Di-Wheel robot could be useful for USAR. Therefore, the focus is on mobility. Additional features will be added as far as possible in the time that is available. The order of priority of desired outcomes are as follows:

1. Prove that the Di-Wheel mechanism is able to provide for the USAR requirements of manoeuvrability.
2. Prove that the motors can be controlled by a microcontroller to climb.
3. Prove that the microcontroller can receive and interpret wireless commands correctly.
4. Prove that a video/audio feed is possible for this robot
5. Prove that a light can be used to illuminate the surrounding area
6. Prove that the wireless commands can be used to reliably control the motors and lights in a full integration of all the above components.

The constraints on the project are related to the price, size and weight of the robot. The set budget is R1500 for development. The size and weight of the robot must be sufficient that a USAR responder can carry it on his/her person, and it must climb standard stairs of height 200mm and be able to fit through small gaps that a dog would be able to.

1.5 Plan of Development

The report begins with a literature review, detailing the background to the project and all theory involved, which is followed by the methodology for the design process to be followed. Concept designs will be considered, and then the design phases will be shown for the chosen concept. These design phases are ordered in terms of the priority user requirements, to obtain a functioning robot, and only once that is achieved to add extra features. The testing results for each design phase will then be shown. Conclusions are then drawn based on these findings, and finally recommendations are made based on these conclusions.

The majority of the work covered is in this report, however there are extra features available in the Di-Wheel Google Drive, such as pictures and videos of testing, mathematical models and excel spreadsheets.

2 LITERATURE REVIEW

A comprehensive understanding of Urban Search and Rescue (USAR), and the robot's mechanism for motion is needed for this project. This literature review aims to provide this knowledge.

2.1 URBAN SEARCH AND RESCUE (USAR) REQUIREMENTS

Emergency situations are very complicated, and the personnel and canine involved are trained to deal with the issues that they encounter. Both human and canine have strengths and limitations, and often compensate for each other's weaknesses. However, some hazards are too dangerous for them. Robotics can assist in a dealing with these complex environments.

This section investigates the requirements that will be placed on the Di-Wheel robot in an urban search and rescue situation, where there will be obstacles that need to be climbed, rough terrain to be overcome, and various hazards present.

2.1.1 A Brief History of Rescue Robotics

It is important to understand the context of USAR (Urban Search and Rescue) robotics and why they are needed. A brief summary will be given as it is not the purpose of this report to outline the full history of rescue robotics.

Tele-operated vehicles are not new to this millennium, with Nikolai Tesla developing a teleoperated boat as early as 1898 [1]. However, it was not until the terrorist attack on the World Trade Center in 2001 that the need for such devices became so apparent. This was because the responders could not get to all the places that they needed to search and if they had robots small enough to go in for them it would have been possible. A summary of the needs identified from 9/11 are:

- Real time data access (site condition, personnel accountability, medical info etc)
- Ability to accurately and non-invasively locate survivors – see through obstacles (walls, smoke etc)
- Communicate around obstacles
- Better monitoring systems
- Integrate multiple functions into one piece of equipment
- Non-human, non-canine systems, with all the strengths but none of the weaknesses

The development of such robots has dramatically increased, along with specific regulations about their requirements set out by different organisations [2]. The Federal Emergency Management Agency (FEMA) have developed a definition of USAR as a result of the burst in development [2].

> *Urban search and rescue (US&R) involves the location, rescue (extrication), and initial medical stabilization of individuals trapped in confined spaces.*

They further explain that it is often due to structural collapse that people get trapped. This may be from various disaster such as "earthquakes, hurricanes, typhoons, storms, tornadoes, floods, dam failures, technological accidents, terrorist activities, and hazardous materials releases" [2]. It is for these situations that USAR robots are developed.

2.1.2 Emergency Situations Requirements

Over the past few decades there has been an increase in literature on USAR robotics. The American Department of Homeland Security, National Institute of Justice and FEMA worked together to identify the requirements for USAR robots. In preparation for this, the National Institute of Standards and Technology (NIST) described the several phases of a rescue operation [3]. These are shown in Figure 1.

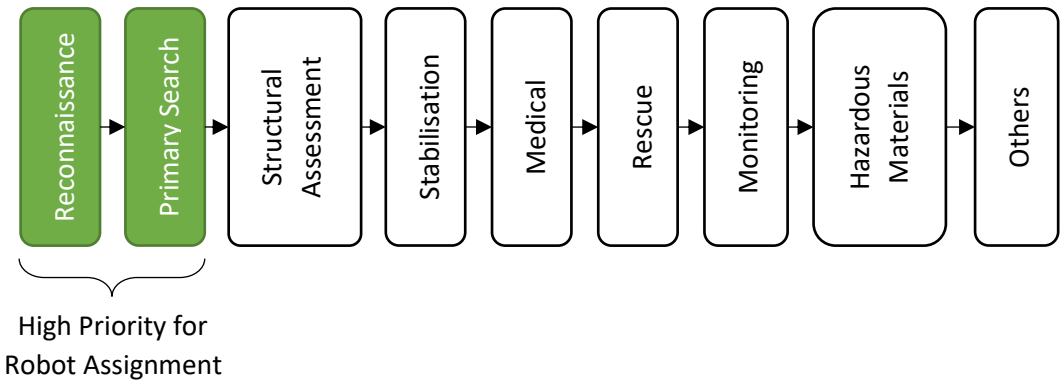


Figure 1: Rescue Operation Phases [3]

Using these phases as a guide, a review of the types of situations involved in USAR emergency response is made. A list of such situations may include [3]:

- Searching for survivors in damaged buildings or under rubble.
- Navigating rough terrain to get to a site, with the risk of beaching on obstacles.
- Disabling explosives.
- Navigating through chemical, biological or radiological hazards.
- Navigating through dusty, wet, or muddy terrain.

The highest priority for robot intervention is in the first two phases [4]. Reconnaissance and primary search. These phases could include all of the situations mentioned above. If robots are deployed in these phases, equipped for these situations, they will enable the responders to find survivors faster, and perform preliminary structural assessment along the way to fast track the entire rescue operation.

The requirements on USAR robots is essentially to utilise the same strengths as human and canine responders, while mitigating their weaknesses. These strengths and weaknesses were studied by Fenton [5]. Humans have good vision, and are very mobile, while dogs can track smells and can move fast on rugged terrain. However, they both require training and are sensitive to hazardous materials.

A robot would reduce the weaknesses significantly as it requires no training. If it is cost effective, then it doesn't matter that it is sensitive to hazardous materials as it is a low cost to pay compared to a human or dog getting killed.

With USAR phases and situations briefly analysed, it is now possible to go deeper to identify exactly what is environmentally required of these robots. For reference, an image of the disaster after an earthquake in Mexico City is shown in Figure 2. These are the typical sorts of obstacles that are encountered in such situations.

Figure 2: Disaster from an earthquake in Mexico City [6]

The requirements of USAR robots are summarised in Table 1 [3].

Table 1: USAR robot requirements [3]

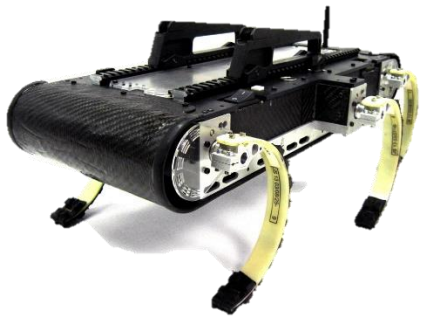
Requirement	Details
Manoeuvring	Overcome obstacles such as stairs, rubble, rocks etc.
Mobility	Move to a desired location, with relative ease.
Dexterity	Using hands to manipulate doors, machinery etc.
Sensing	Sense the environment, avoiding obstacles, and detecting hazards or structural/ atmospheric change.
Endurance	This typifies how long the robot can last in terms of power. This requires these robots to be energy-efficient.
Radio Communication	Communicate with robot and survivor via radio, as a tether is difficult in USAR situations because of the tether getting caught. Used to inspire hope of survival.
Durability	Physical durability in the face of expected damage, and wear and tear.
Reliability	Perform reliably, allowing the responders to trust that it will work properly
Autonomy	This could be any level of autonomy.
Logistics	Simple to obtain, easy to transport.
Safety	Safe to use, not presenting any additional risks for operators or survivors.

This table describes the requirements in detail. It can be summarised as being that the robot must be versatile in facing and overcoming various unexpected situations, whether inside or outside.

2.1.3 Various USAR Designs

There are too many designs in use to go into detail about each of them, so a very condensed summary of some options available is shown in Table 2.

Table 2: USAR robot designs

Image	Description
	**RHex** [7] A hexapod robot designed with six rotating compliant legs to allow robust and agile motion. It is capable of climbing over most obstacles, and ascending stairs. Its mechanical design allows for simple control with fantastic results. It is expensive and large but has many sensors and weatherproofing features.
	**Throwbot** [8] A sturdy, throwable robot developed by ReconRobotics. It is capable of withstanding repeated drops from 9m. It uses two plain wheels on a cylinder body, and a pliable tail for balance. It is small, but quite costly.
	**Firstlook** [9] A small, lightweight, throwable robot developed by iRobot. It is primarily for infantry and special forces. It is used for explosive and bomb detection as well as observation. It is capable of climbing over small objects using the two side arms to leverage itself over. It is also quite costly, despite its small size.
	**Pointman** [10] A rugged, throwable robot, designed with an end-over-end capable function which allows it to climb stairs. This makes it self-righting and versatile in its motion. It is small and less expensive, but only comparatively.

With all of the options in Table 2, each falls short in one or both of the two must desired features:

- Small in Size
- Low in cost

The reason for these being the most desirable is simple. There is no point in using a robot with all these incredible features if you can't carry it, and if you are so scared that it will get damaged then you will

end up putting yourself at risk to rescue it in the case that it gets stuck. Wilson reported the findings from a conference in Turkey in 2012, regarding the existing robots at the time [11]:

> *"Many were both cumbersome and expensive. What was wanted were small, man-portable, low-cost robots that, while perhaps not doing everything many of the existing robots could, would be easily acquired, deployed, and replaced if destroyed or lost."*

The scope of this project is informed greatly by this, as there is a need for small and low-cost robots to fulfil the needs required, without too many additions as this would increase cost.

2.2 Di-Wheel Concepts

The Di-Wheel is a mechanical limb with internal epicyclic gearing developed by Matthew Wilson in 2013 [11]. This section will make reference to Wilson's work and build upon his idea with updated research. A motor is connected to the centre of the Di-Wheel such that the Di-Wheel acts as a small RC car's wheel structure. Figure 3A shows the basic structure of the Di-Wheel, with a central sun gear, idler gears and planet gears that are fastened to the wheels.

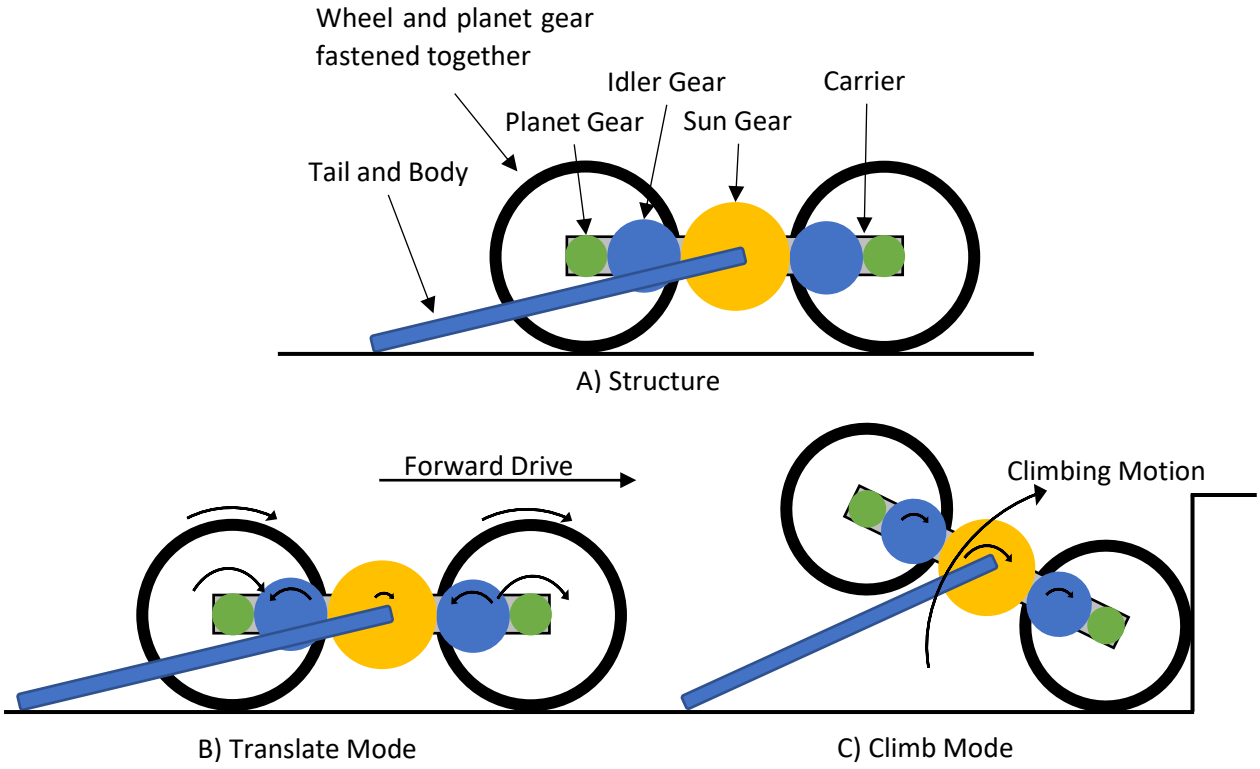


Figure 3: Basic Functional Concepts of a Di-Wheel

Figure 3B shows the translate mode, which occurs on smooth terrain. The Di-Wheel does not rotate but rather the internal gears turn two wheels at each end of the limb, resulting in normal driving. Climbing mode is shown in Figure 3C. When the front wheel encounters resistance (such as a stair) the friction causes it to seize, and as a result the Di-Wheel rotates over the obstacle. This creates a friction intuitive climbing motion. Therefore, it was originally called a Load Intuitive Module (LIM). The name was changed to Di-Wheel because the mathematical modelling and physical testing proved that if a lower

torque were aapplied it would only drive, if a medium torque were applied it would become load intuitive, and it would only climb for high torques. This implies that it was only load intuitive for medium torque inputs. Therefore, the name was changed to Di-Wheel to not confuse the reader into thinking there is only one state of behaviour. This also follows on from the Tri-Wheel name from Smith et al. as explained below [12].

2.2.1 Di-Wheel Development

Wilson took inspiration from the Lockheed TerraStar amphibian for the Di-Wheel gear design [13]. This amphibian vehicle was designed to drive on land using a tri-wheel set up, and then use the same tri-wheel set up to propel itself in water.

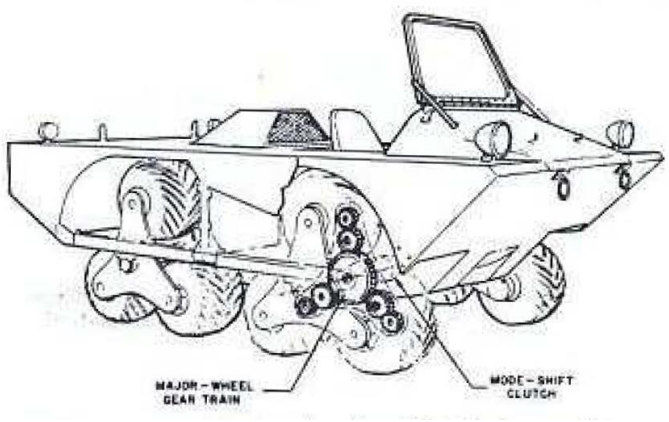


Figure 4: Lockheed TerraStar Amphibian [13]

The gearing in each wheel mechanism represented a major/minor wheel design. The major wheel was the entire mechanism and the minor wheels were the individual wheels. In addition to this mechanism was a clutch to shift between minor wheel drive and major wheel drive (direct drive). This mechanism, along with the clutch was used by Smith et al. to develop a search and rescue robot for fire response [12].

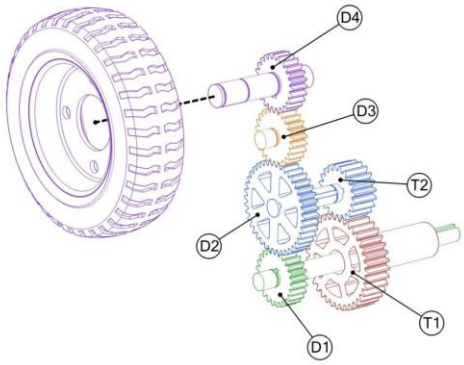


Figure 5: Minor Wheel Gear Train, and Clutch Mechanism [12]

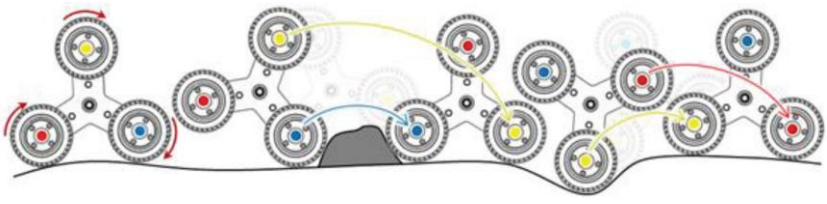


Figure 6: Passive Tumbling of Tri-Wheel [12]

Smith et al. (2015) proposed that a robot could use this mechanism even without the clutch to passively "tumble" over obstacles. This proposal was verified, with the robot climbing larger obstacles using passive tumbling (no clutch) than active tumbling (using the clutch). They called the gear train a Tri-Wheel. This name will be used for the rest of the report.

The Di-Wheel was developed from the Tri-Wheel wheel concept by simply removing the one wheel. The reason for this, is that the lower profile design is desirable for a USAR robot. This is demonstrated in Figure 7.

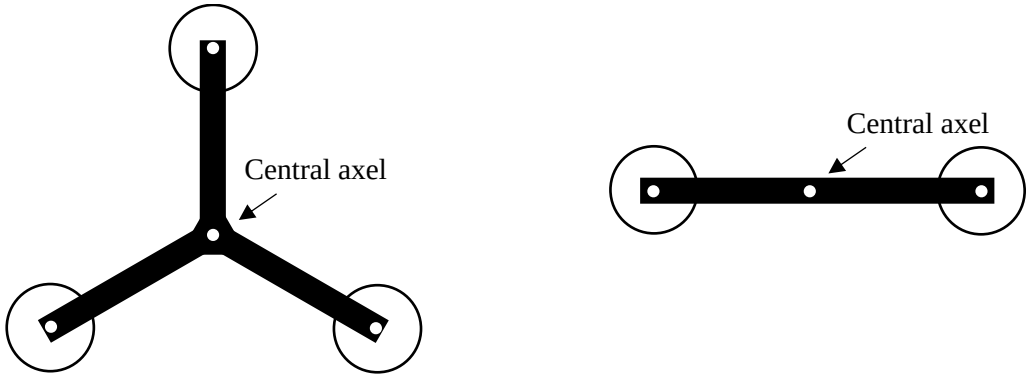


Figure 7:Tri-Wheel and Di-Wheel Concepts

The Di-Wheel concept would work in much the same way as the Tri-Wheel, except that it would have to fight against gravity more. The reason for this is that the central axel is horizontally aligned with the wheel axel so that the moment arm is much larger, meaning that the gearing must overcome this in order to lift that central axel, and thus the body weight. The Tri-Wheel design's central axel is lifted so if the robot is pushed forwards it would facilitate climbing with little torque needed from the sun gear motor. This was seen by Smith et al. when they included a set of active wheels at the rear of their robot that pushed the robot forwards when attempting to climb.

Figure 8: Smith et al. (2015) Tri-Wheel concept using an active rear wheel to assist in pushing the robot forward when climbing.

The Di-Wheel robot removes the third wheel; which reduces the torque needed by the Tri-Wheel motor (the motor driving the Tri-Wheel gear train); the clutch gearing, and the rear wheel; which assists the Tri-Wheel motor by pushing from the back. This would result in a much simpler but seemingly less effective design. However, these reductions also severely reduce the weight so that the robot can be much smaller, have fewer motors, and if made from light materials can possibly still perform the same motion as the Tri-Wheel concept.

2.2.2 Motion of Di-Wheels

In order to understand the physics of why the Di-Wheel system behaves as it does, an analysis of the gearing must be performed. The basics of gears are not covered here, but can be found in Section 10.2 of the Appendix.

Di-Wheels rely on similar principles to epicyclic gear trains. An example is shown in Figure 9. The central sun gear meshes with the surrounding planet gears, which mesh with the ring gear. The planet gears are held in place with a carrier which rotates around the shaft of the sun gear. The output can be a shaft connected to the ring gear or the carrier.

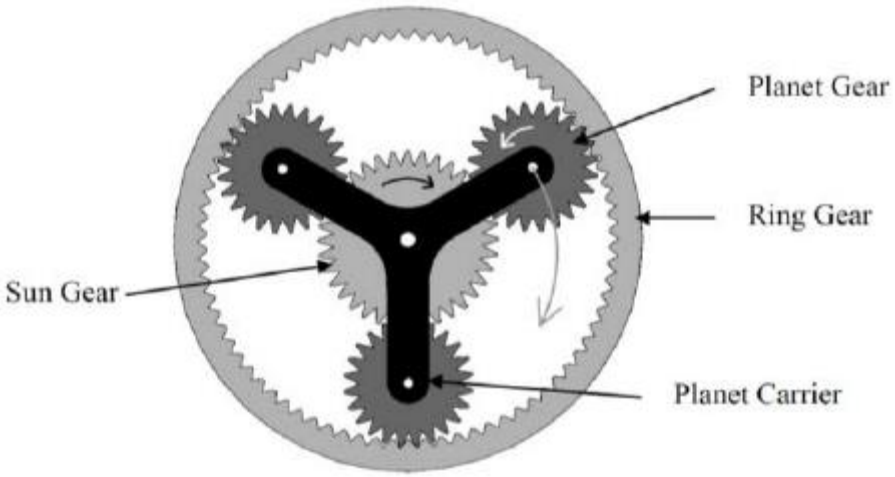


Figure 9: Epicyclic gear train

The Di-Wheel gear train uses these principles, except that the planet gear does not mesh with a ring gear, but rather interacts with the ground through a wheel. The motion of a Di-Wheel gear train can be analysed using the same techniques as planetary gears. The method of calculating the transmission ratio is shown in Section 4.2.1, where a mathematical approach is taken to analyse this motion.

2.2.3 Previous Work on the Di-Wheel Robot

The Di-Wheel robot was initially developed by Matthew Wilson in 2013 [11]. It was later worked on by Jordan Haskel [14] and Murray Buchanan [15] in 2017 and 2018 respectively. The various names that were given to it are:

- LIMs robot [11]
- Theseus robot using LIMs [14]
- Ascender robot using LIMs [15]

As explained in Section 2.2, the name was changed to Di-Wheel to not confuse the reader into thinking the mechanism is purely load intuitive.

The reports from these three students provided valuable insight into the workings of the Di-Wheel mechanism and their recommendations guided the development of the entire Di-Wheel robot. A summary of the various recommendations made is:

- Materials:
    - Haskel found that using large wheels reduced the chance of beaching when climbing [14]. This comes at a cost of not being able to reach over a ledge and pull itself up.
    - Buchanan recommended grousers to reduce slipping [15].
    - Wilson recommended a flat box chassis for better housing of the electronics [11].
    - Use simple materials, that are easy to acquire and assemble.
    - Expensive material can be cheaper. Its strength requires less material [11].
- Sensors [11]
    - The video feed can be low quality as long as the image is reliable. This reduces cost. It is best if the camera is placed such that the whole robot (its limits) can be seen.
    - It is quite useful to have thermal or infrared imaging, although it is costly.
    - Inclinometer, accelerometers and a compass are useful features.
    - Simple Atmosphere and gas detectors can be inexpensive.

Using these valuable insights, the design process was much improved.

2.3 Summary of Literature Review

The study performed on USAR conditions shows that robots are most useful in the initial reconnaissance and search phases. Small, agile robots are needed for these phases that can easily find survivors trapped in an urban disaster zone. The specifications required were outlined and various other robots considered.

The Di-Wheel mechanism was then considered as a possible solution to the USAR problem. With the proposed manoeuvrability and potential for sensory feedback at a cost-effective rate, this project can satisfy various of the requirements outlined.

3 METHODOLOGY OF THE PROJECT

The methodology of the project is discussed in this section. This is a general approach, and more detailed methodologies are discussed in the various design phases, as will be discussed in this section.

This section answers the following questions.

- What was the problem under question?
  - Constraints that are involved;
  - The specifications.
- What was the approach to the problem?
  - How the project was planned out;
  - How the work was broken down (Work Breakdown Structure);
  - Specific tasks and testing procedures to ensure specifications met (V-diagram).
- What process was taken to test the approach taken and the results of the project?
  - The Acceptance Test Protocol that was used to verify the product built against the user requirements.

3.1 PROBLEM DEFINITION

The goal of this project is to determine if the Di-Wheel concept can be used to create a small, cost effective, tele-operated robot that can easily traverse over small obstacles like stairs or rubble. This prototype robot does not have to meet all requirements of USAR robot which include high manoeuvrability, mobility, dexterity, sensing, endurance, radio communication, durability, reliability, autonomy, logistics and safety. These are discussed in depth in Section 2.1.2, Table 1. The problem at hand is to see if the Di-Wheel can actually be of use in USAR environments. Once that has been proven then additional features can be added to help it meet the requirements. The only requirements that this project focuses on are manoeuvrability, mobility, endurance, radio communication and reliability.

The solution to this problem can be divided into its various components. There will need to be a mechanical design incorporating the Di-Wheel mechanism and a body to house the electronics. Then there is the development of the electronics that operates the mechanism using motors. The electronics must be controlled wirelessly using a remote control. There must be a dedicated power supply from a battery source on board.

That is all that is needed to provide a basic prototype to show that the Di-Wheel concept can be used in USAR operations. Additional features to meet additional requirements can be added if time permits, such as a camera/microphone for radio communication. The order of priority of what is expected from this project is:

1. Prove that the Di-Wheel mechanism is able to provide for the USAR requirements of manoeuvrability.
2. Prove that the motors can be controlled by a microcontroller to climb.
3. Prove that the microcontroller can receive and interpret wireless commands correctly.
4. Prove that a video/audio feed is possible for this robot.
5. Prove that a light can be used to illuminate the surrounding area.
6. Prove that the wireless commands can be used to reliably control the motors and lights in a full integration of all the components together.

3.2 USER REQUIREMENTS AND SPECIFICATIONS

Using the problem as defined, the following user requirements have been outlined. These are shown in Table 3.

Table 3: User Requirements

Reference number	Requirement	Description
UR-1	Stair Climbing	The design of the robot must be able to climb standard height stairs or over small obstacles.
UR-2	Driving	On flat surfaces the robot must be able to drive without trying to climb.
UR-3	Cost effective	The total cost of the robot must be under R1500 and provide easily replaceable parts. All parts should come from what is immediately available at UCT if possible, to reduce costs. This includes laser cutting, fasteners and electrical components.
UR-4	Tele-operated	There must be no tether for power or comms, the user must be able to remote control the robot from a distance.
UR-5	Easily portable	The robot must be light and small such that it can be easily carried by a user. It should be easily disassembled and reassembled.
UR-6	Non-beaching	The robot must have little chance of becoming beached with no way to recover it without the user going to fetch it.
UR-7	Complete package	All the components of the robot must be in a closed package and must have no exposed wires that a user could be hurt on. The whole design must not present any hazard to a user.
UR-8	Presentable	Although it is only a proof of concept, it must be aesthetic in design.
UR-9	Robot response must be repeatable and reliable	The response of the robot to commands should be consistent and predictable.
UR-10	Safety	The robot must be electrically and mechanically safe to operate, presenting no danger to a user or survivor.
UR-11	Easily programmable	The robot must be easily programmable
UR-12	Test procedures must be in place	The robot must have test procedures in place to properly measure performance

The specifications for accomplishing these user requirements are shown in Table 4. Certain specifications are marked as low priority and are considered outside the scope of this project and for addition if time permits. Following the table is a discussion on each of the absolute necessity specifications.

Table 4: User Specifications for accomplishing the User Requirements

Reference number	Specification	Description	Link to User Requirements
		General Design	
SG-1	Full implementation	All features must work together at the same time.	UR-1, UR-2, UR-4, UR-7, UR-9
SG-2	Cost<R1500	The total cost must be below R1500.	UR-3
SG-3	All components fit together in body	The components must all completely fit together inside the body of the robot.	UR-5, UR-7
SG-4	Neat assembly	The package and electronics must look neat and presentable.	UR-5, UR-7, UR-8
SG-5	USAR appearance	The appearance of the robot must be a USAR robot, not a toy.	UR-5, UR-6, UR-7, UR-8
SG-6	Open loop control	The user must be the control system.	UR-1, UR-2, UR-9
SG-7	Reliable in response to commands	The robot must always respond in a predictable way to commands.	UR-1, UR-2, UR-4, UR-6, UR-9, UR-10
		Mechanical Design	
SM-1	Minimum Di-Wheel size	220mm long (this is the length of the carrier).	UR-1
SM-2	Wheel diameter	Maximum diameter is the length of the carrier (220mm)Minimum diameter is the diameter of the largest gear.	UR-1, UR-6
SM-3	Wheels must have traction	The wheels need to have sufficient traction to grip when encountering an obstacle.	UR-1, UR-2, UR-6
SM-4	Climb steps of varying heights	The robot must be able to climb over varying step heights.	UR-1, UR-6
SM-5	Body encloses electronics	The body must completely encapsulate all the required electronics with no exposed wires.	UR-5, UR-7, UR-10
SM-6	Maximum Body height	It must only be high enough to fit the motors and or batteries. This reduces chances of beaching underneath.	UR-5, UR-6
SM-7	Maximum Body width	It must only be wide enough for motors to fit, this reduces the chance of beaching in front.	UR-5, UR-6
SM-8	Weight<5Kg	The weight must be less than 5Kg for easy carrying, and climbing.	UR-1
SM-9	Body and tail length	The body and tail length specify the moment arm of the body, this must be longer than half the length of the Di-Wheel mechanism. This will ensure that the mechanism always has something off which to push.	UR-1
SM-10	Easy assembly	The parts must be easy to assemble for manufacture of robot.	UR-3
SM-11	Partial disassembly	The robot must be partially disassembled for transporting, with easy and quick reassembly.	UR-5
SM-12	Shoebox packing space	The partially disassembled robot must fit in a shoebox for transportation.	UR-5
SM-13	No sharp edges	The construction must be safe for users.	UR-5, UR-10
SM-14	Covered Gears	The gears must be covered and not exposed to potentially hurt a user holding the robot.	UR-10

Materials, Parts and Components Procurement			
SP-1	Easily accessed materials	The materials must be easily accessible. This means they must be available at the university or locally.	UR-3
SP-2	Drop resistant materials	The materials must be strong enough to resist falls from steps. The expected drop is 200 mm.	UR-1, UR-3
SP-3	Easy manufacture methods	The methods of manufacture must be easily reproducible by any reader of this report. The simplest methods are through laser cutting of parts, and 3D printing.	UR-3
SP-4	Least expensive materials and components	The materials and components must be inexpensive to reduce the overall cost.	UR-3
SP-5	Builder's Guide	A builder's guide must be generated for any user to easily manufacture their own robot.	UR-3
Electrical Design			
SE-1	Long battery life	In idle: 4 hours. constant driving: 2 hour. constant stair climbing: 1 hour. Must be able to easily change the battery.	UR-9
SE-2	Wireless controlled	Using 2.4GHz for remote control of robot as this is standard for remote controls.	UR-4, UR-9
SE-3	Sufficient motor torque	The motor torque must be sufficient to allow climbing.	UR-1
SE-4	Fastened components	The components must be held in position for when the robot is upside down.	UR-1, UR-10
SE-5	Visual/ Audio feed	Use a camera and mic setup to relay visual and audio feed of the robot. This must run on 5.8GHz as this is standard. This is an extra feature and is not high priority.	UR-4
SE-6	Flashlight	Use high power LEDs to illuminate the area for dark conditions. This is an extra feature and is not high priority.	UR-4
SE-7	Sensory feedback	Temperature, gas, humidity feedback can be implemented. This is an extra feature and is not high priority.	UR-4
SE-8	Communication method with survivors	Using either lights, or audio output with prerecorded messages, the robot must be able to give hope to a trapped survivor.	UR-4
SE-9	No loose wires	For safety there must be no loose wires hanging outside the robot.	UR-10
SE-10	Power toggle switch	There must be a power toggle switch to immediately depower the robot in the event of a problem.	UR-10
SE-11	Batteries must not present danger	Li-Po have a potential to explode, so Li-Ion should be selected instead.	UR-10
SE-12	Onboard Debugger	There must be an onboard debugger to allow easy upload of code.	UR-11

Software Development			
SS-1	Test procedures	There must be test procedures for debugging purposes.	UR-12
SS-2	Reliable outputs with given inputs	The output of the microcontroller must be reliable depending on the input received.	UR-9, UR-12
SS-3	Read in receiver channels	Must be able to read in receiver commands, and map them to a scale of percentage.	UR-4
SS-4	Easily switch channels	Global defined variables must define the channels used as input.	UR-4
SS-5	Intuitive driving	The method of interpreting commands and controlling the motors must be intuitive to a user.	UR-1, UR-2, UR-4, UR-9

Certain of the specifications mentioned in Table 4 require explanation for the values selected. This is explained below:

- Minimum Di-Wheel size

The Di-Wheel size requirements were defined using the NIST standard stair height of 200mm at an inclination of 45^{o} [16]. To ensure that the Di-Wheel is able to climb over the obstacle, the design must be such that the top wheel (the one rotating over) must completely clear the obstacle as seen in Figure 10.

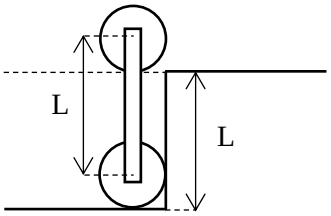


Figure 10: Derivation of size requirements for Di-Wheel

With this standard the design should be a little more than that, so 220mm will be the minimum requirement for Di-Wheel length.

- Maximum Body Size

A reduced body size reduces the likelihood of beaching. Forward beaching is prevented by a narrow body and undercarriage beaching is prevented by making the height less than the wheel size.

- Weight

The weight is defined as 5kg as this is an easy weight to carry over a long distance, as responders often have to travel by foot.

- Tail

There are a few options for the tail, it can be rigid or flexible, fixed or active. To reduce complexity for this project it will only be a simple, lightweight, fixed, rigid tail.

- Battery Life

These are purely based on the expected duration of operations. A single responder won't remain in one area for longer than an hour and so will have the robot back within that time. If it was climbing that entire time (the most energy intensive move) it must be able to return in an hour.

- Cost

This cost is defined by the limit of R1500 that is imposed by the project requirements from the Electrical Department.

- Remote Controlled

2.4GHz is the standard frequency for RC airplanes and cars, which technology will be used extensively in this design.

- Materials

The materials must be cheap enough to be within budget and prove the Di-Wheel concept. Their durability needs to be sufficient to survive sufficiently rough conditions but can be lower quality as it is meant to be an essentially disposable robot.

3.3 PROJECT PLAN

In order to accomplish the user requirements discussed in the previous section, a project plan was developed. This entails the exact approach to the design of the robot, how the work was broken up, and how testing was to be performed in order to validate that the user requirements have been met.

3.3.1 Design Approach

The design approach used in this project was to divide the work into design phases, according to both priority and pre-requisites. The design phases focused first on getting a mechanical platform operational, and then developing the electronics and software to control it.

The V-Diagram in Figure 11 shows the exact project flow that was employed in this project.

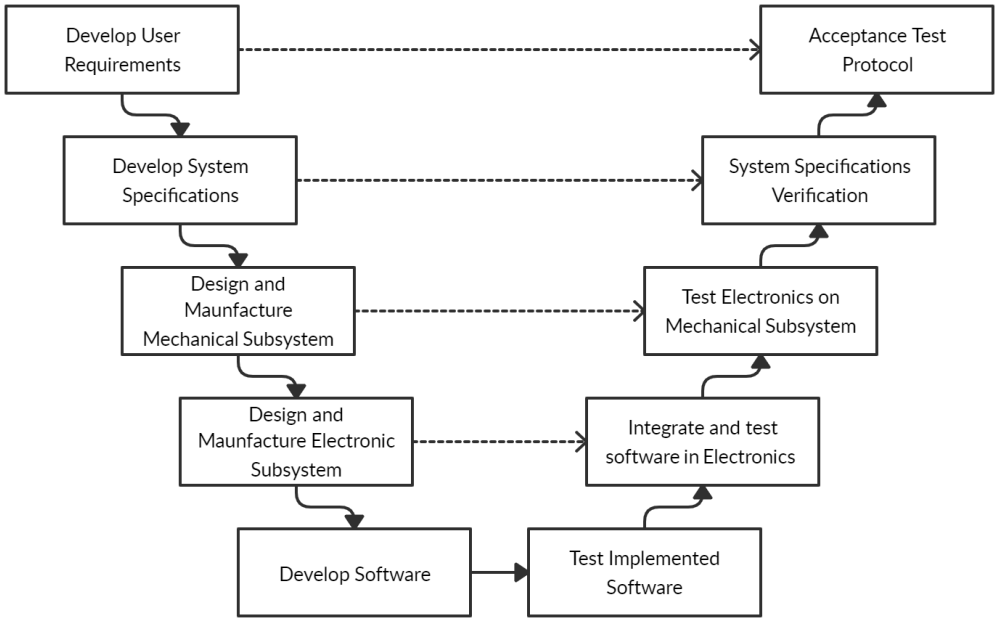


Figure 11: V-Diagram of D-Wheel Project Plan

The task of designing the Di-Wheel robot was broken up into one concept phase, three design phases, and a testing phase. The three design phases selected are:

1) Phase 1: Tethered Driving
2) Phase 2: Untethered Driving
3) Phase 3: Extra Features

The reason for these design phases is to allow iterative learning and development. In this way each component of the design is conceptualised, designed and manufactured and then tested before trying to integrate it with the rest of the robot. The Work Breakdown Structure (WBS) in Figure 12 on the next page shows the summary of tasks to be accomplished in each phase.

As can be seen in the WBS, the concept phase has already been covered in the preliminary sections. Each design phase has its own design and testing sub-phases. Finally, there was a testing phase for the full integration of all these components together.

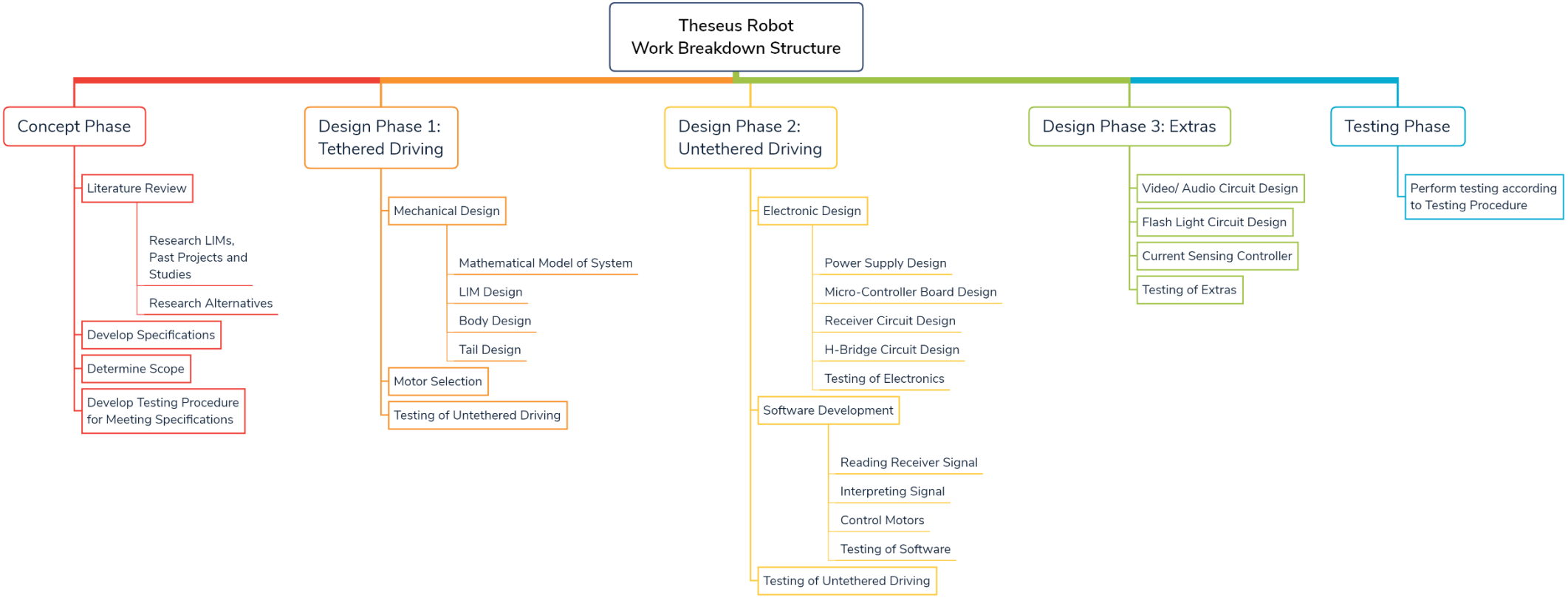


Figure 12: Work Breakdown Structure for the Di-Wheel Robot Project, Showing all Design Phases and Subtasks

3.3.2 Acceptance Test Protocol

The ATP is directly linked to the user specifications to ensure that the user requirements are met. The outline of where each requirement will be tested is shown in Table 5.

Table 5: Acceptance Test Protocol Outline

		Design Phase 1	**Design Phase 2**	**Design Phase 3**
UR-1	Stair Climbing	X		
UR-2	Driving	X		
UR-3	Cost effective	X	X	X
UR-4	Tele-operated		X	X
UR-5	Easily portable	X		
UR-6	Non-beaching	X		
UR-7	Complete package	X		
UR-8	Presentable	X	X	
UR-9	Robot response must be repeatable and reliable	X	X	X
UR-10	Safety	X	X	X
UR-11	Easily programmable		X	
UR-12	Test procedures must be in place	X	X	X

By testing for these User Requirements in the sections shown the success of the project will be guaranteed.

3.4 SUMMARY OF METHODOLOGY

The methodology outlined specifies that three design phases will be followed to build this robot. This included testing phases continuously throughout in order to ensure that the user requirements were met.

4 DESIGN PHASE 1: TETHERED DRIVING

The initial design phase focuses on whether the Di-Wheel design be used to climb stairs. The mechanical design is shown here. Motors are included, with the robot being tethered for power.

4.1 PROPOSED CONCEPT DESIGNS

Three concepts (shown in Figure 13) were created for the Di-Wheel robot and considered qualitatively. The first is a two Di-Wheel design with a tail to introduce stability. The second is a two Di-Wheel design with active wheels at the back to push the robot forward as it drives. The third design uses four Di-Wheels.

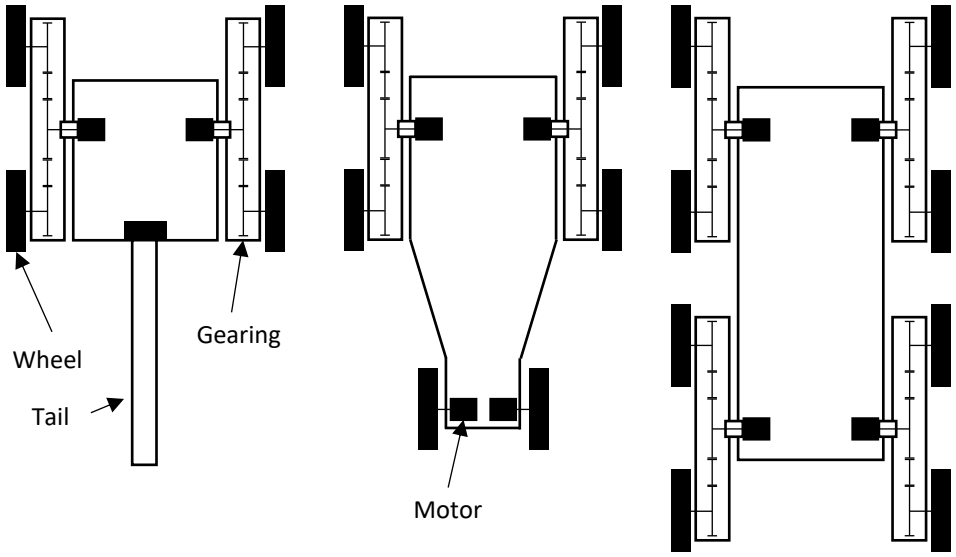


Figure 13: Proposed Concepts

The characteristics and properties of each concept are considered in Table 6. A scale of 0 to 5 is given for the different characteristics. A score of 0 means it is the lowest on the scale compared to the other concepts, 5 is the highest.

	Concept 1	**Concept 2**	**Concept 3**
Complexity	0	2	5
Number of motors	0	5	5
Expected cost	0	2	5
Expected weight	0	4	5
Manoeuvrability (0 is for the best)	0	5	5
Total	0	19	25

Table 6: Comparison of concept designs

The values selected for complexity were based on how easy it would be to build. The Di-Wheels themselves are the most complex, so the fewer Di-Wheels the less complex it is. Concept 2 was still more complex than concept 1 because of the extra motors involved. Concepts 2 and 3 each had more motors, thus their rating of 5 for motors. The expected cost was directly linked to complexity and so received the same values. The weight was estimated based on the number of motors and Di-Wheels and the body size required for it. The extra motors in concepts 2 and 3 require a larger body and thus extra

weight, but concept 2 is just a little lighter because it has only two Di-Wheels. The manoeuvrability values were based on size and whether the wheels would get in the way. Concepts 2 and 3 were both large and thus less manoeuvrable.

This is a very simple comparison which doesn't take many considerations into account. Performing a quantitative analysis of the three designs would provide a more accurate comparison. For the purposes of this project it is satisfactory, as it clearly shows that concept 1 is the easiest and most effective to use as a proof of concept of the Di-Wheel design. Once this has been proven, future projects can consider the other concepts.

4.2 Mathematical Model of Robot

To understand the motion of the Di-Wheel and the body, and how the gearing and the sizes of everything relative to each other effect things, three methods are used. First, the transmission ratio of the gears is considered. Second, a kinematic calculation to understand the impact of the forces on the whole system, and third, a Lagrangian mechanics derivation to obtain the equations of motion. The hope was to perform simulations with this, but after the derivation became messy it was decided that the work done was sufficient to guide the mechanical design.

The transmission ratio calculations revealed that there are three general gearing setups, with the resulting motion shown in Figure 14.

1. Flipping over motion – sun gear larger than the planet gear.
2. Unpredictable motion – sun gear and planet gear are the same size. The resulting motion is unclear but it is expected to be unpredictable, using either option 1 or 3.
3. Ramping up motion – sun gear smaller than the planet gear

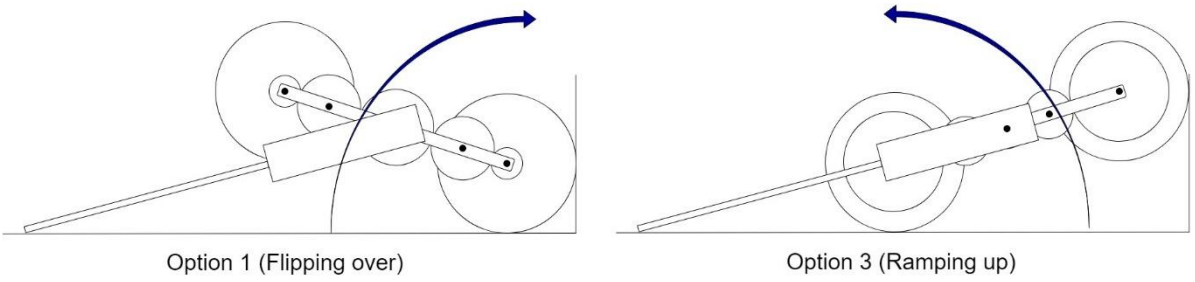


Figure 14: Options for gear sizes and resulting motion

All calculations were done using option 1 (Flipping over). The reason for this is two-fold. The calculations for this were simpler (fewer forces involved with back wheel in the air), and it was unclear whether the Di-Wheel would actually climb or just flip over backward instead of ramping up.

4.2.1 Di-Wheel Gears Transmission Ratios

The method of calculating the transmission ratio is shown below. The gear diagram in Figure 15 has an idler gear in between the sun gear and the planet. The ring gear is equivalent to the ground in the Di-Wheel context. For an understanding of how epicyclic gear trains work, refer to Section 2.2.2.

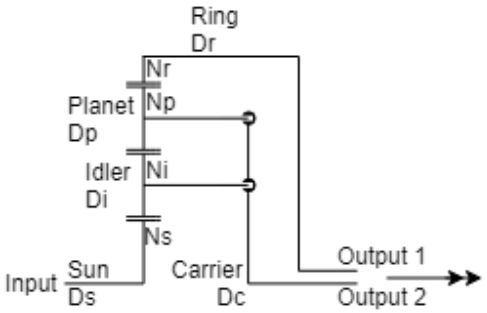


Figure 15: Representation of half a planetary gear train with an idler gear

If the carrier is held still, with an input of 1 radian to the sun gear, the following output angular positions occur for each gear. For a Di-Wheel gear train this is equivalent to translate mode, where the carrier is horizontal, and the ground ("ring gear") is moving relative to the planet gear.

Ds (input)	Di	Dp (output)	Dc
1	$-\frac{Ns}{Ni}$	$-\frac{Ns}{Ni}\left(-\frac{Ni}{Np}\right) = \frac{Ns}{Np}$	0

The output transmission ratio is equal to $TR_{translate} = \frac{output\ rotation}{input\ rotation} = \frac{Ns}{Np}$ as the output is the planet gear in translate mode.

The wheels are turning faster than the sun gear if the planet gear is smaller. This could result in higher speeds than desired, which could cause problems. The planet gear would then need to be as large as possible to reduce translational speed. To counter this, the motor speed was geared down before entering the Di-Wheel mechanism to reduce the speed expected.

In the case that a different gear to the carrier is stationery, you use the values from the table above and add or subtract from all values the amount required to get the desired gear to zero. The resulting values are how much each gear moves for the new calculated input. These are relative motions to the selected stationery gear.

Let's say the planet gear D3 was stationery, you get the following values. For a Di-Wheel gear train this is equivalent to climb mode.

Ds (input)	Di	Dp	Dc (output)
$\mathbf{1 - \frac{Ns}{Np}}$	$-\frac{Ns}{Ni} - \frac{Ns}{Np}$	0	$-\frac{Ns}{Np}$
1	$\frac{Ns(Np+Ni)}{Ni(Ns-Np)}$	0	$\frac{Ns}{Ns-Np}$

The output transmission ratio is equal to $TR_{climb} = \frac{output\ rotation}{input\ rotation} = \frac{Ns}{Ns-Np}$ as the output is the planet gear in translate mode.

Using these gear ratios, an excel spreadsheet was created that analysed the expected forward velocity with no obstacle, compared with the carrier rotational speed if the planet gear was stationary. Refer to the Section 10.3.1 in the Appendix for more details. The resulting graph is shown in Figure 16.

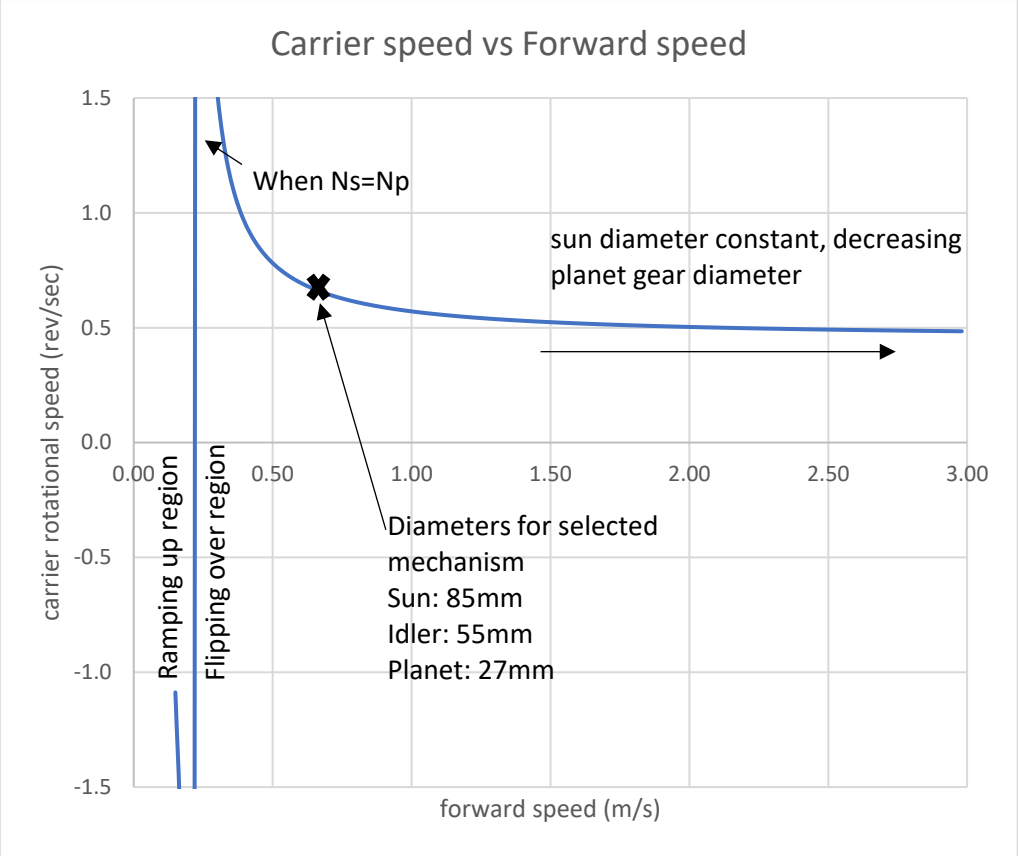


Figure 16: Plot of Carrier speed vs Forward speed calculated using gear ratios

As this project is only dealing with the flipping over motion, this is in the region to the right of the asymptote. Climbing motion seems more likely the when the planet gear is a little smaller than the sun gear, and driving motion is more likely when it is a lot smaller.

The gear diameters selected from this graph were 85mm, 55mm, and 27mm for the sun, idler and planet gear respectively. This is the point where rotational speed and translational speed are equal. This is all with the condition of the carrier being 220mm long, and the wheel being 150mm in diameter.

4.2.2 Kinematic Calculations

The purpose of these calculations was to check two things. What torque is required for each motion (translating and climbing)? And what is the optimum gear ratio to reduce the required torque?

4.2.2.1 Initial Calculations (Few Assumptions)

Initial calculations were performed, but these quickly became overcomplicated and the results were not trusted. This is because the individual forces between gears and the body were evaluated. This gave several insights to the motion of the gears and the body relative to them according to their size, but was not useful for an actual mathematical model. These insights were:

- The idler gear resists the flipping over motion. It tries to rotate down and around the planet gear. The carrier itself rotating faster than the input allows the climbing motion.
- The more the Di-Wheel flips over, the more weight is carried by the front wheel which increases the normal force and friction. This results in it gripping more which is essential to this action working.

The actual results of these initial workings were:

$$\text{Translating mode: } a_x = \frac{\left(-\frac{N_p}{2N_s}\tau_m + f_A R_w\right)R_w}{I_w}$$

$$\text{Climbing mode: } \alpha_c = \frac{\tau_m \cos(20)\left(R_s - R_p\right)}{R_s I_c} - \frac{m_w g A + F_{ax} B + F_{ay} C}{I_c}$$

$$\text{where } A = \left(D_p + 2D_i + D_s\right)\cos(\theta)\,, B = \frac{\left(D_p + 2D_i + D_s\right)}{2}\sin(\theta)\,, C = \frac{\left(D_p + 2D_i + D_s\right)}{2}\cos(\theta)$$

The main reason these were discarded for anything but insights, was that with an input torque of zero, there is a resulting angular acceleration of the planet gear in translating mode, and the carrier in climbing mode. However, it is expected to be at rest in this case, so these results were not initially trusted.

4.2.2.2 Simplified Calculations with Assumptions

Attempts to simplify the calculations were done. After some assumptions and simplifications the following calculations were obtained. A summary of these assumptions is:

- The mass of the gears is negligible;
- These are dynamic calculations, starting from rest and about to start moving as the torque is applied from the motor;
- The coefficient of friction is large enough that the front wheel is not rotating.

The full workings, along with a force diagram is in Section 10.3.2 of the Appendix. The results of these derivations were:

Translating mode:

$$a_x = \mu g\left(1 - \frac{m_b L_{com}}{L_{tax}\left(m_c + 2m_p + m_b\right)}\right) - \frac{\mu}{\sqrt{L_{tax}^2 - R_w^2}\left(m_c + 2m_p + m_b\right)}\tau_m$$

Climbing Mode:

$$\text{let } A_1 = I_B \frac{X_1}{L_{tax}^2 \cos(\theta_b)} + m_b \frac{X_1}{L_{tax}}(L_{tax} - L_{com}),$$

$$\text{and } A_2 = \frac{\left(\tau_m + m_b g L_{com}\cos(\theta_b)\right)}{L_{tax}\cos(\theta_b)} - m_b g$$

$$\text{and } A_3 = X_1 + \frac{R_w(\mu^2 + \mu)}{(\mu^2 + 1)}$$

$$\text{and } A_4 = -\tau_m + A_3\, g\left(m_c + 2m_p\right),$$

$$\text{and } A_5 = I_c - X_1\left(m_c + 2m_p\right)A_3$$

$$\alpha_c = \tau_m \frac{\left(1 + \frac{A_3}{L_{tax}\cos(\theta_b)}\right)}{-A_5 + A_3 A_1} - \frac{A_3\, g\left(m_c + 2m_p\right) + A_3 m_b g\left(1 - \frac{L_{com}}{L_{tax}}\right)}{-A_5 + A_3 A_1}$$

This result is similar to the first calculations' results. This immediately raises the same question, why is there acceleration for zero torque. Perhaps through the simplification there is a form of non-linearity lost. The resulting graph for these calculations is seen in Figure 17.

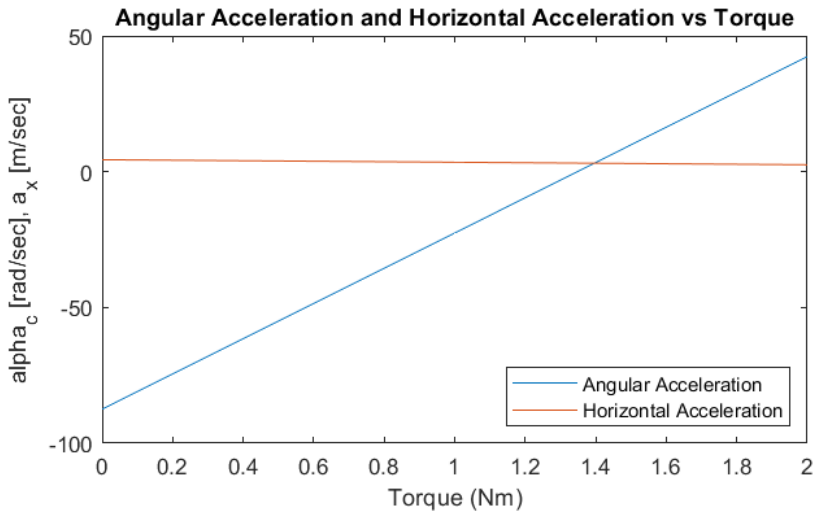


Figure 17: Kinematic Derivation of Angular Acceleration and Horizontal Acceleration vs Torque

This plot was calculated using the dimensions of the robot that was eventually built. If this is approximately correct, it gives insights to the Di-Wheel's behaviour. Translating mode is likely for low torques and climbing mode for high torques. The torque of the motor must be able to exceed the cross-over point on the graph, otherwise the robot will never be able to enter climbing mode. These exact results are not trusted for their numerical value, but the insight given may be useful for further study.

4.2.3 Lagrangian Mechanics Calculations

Lagrangian mechanics is an appealing option as it deals with energy, avoiding the messy kinematics calculations. In the end, the Lagrangian method was abandoned because it is particularly difficult to include dissipative forces (friction). The derivations are provided in Section 10.3.3 of the Appendix for the reader's reference. Before completely abandoning it, a plot was created using the result of the derivation.

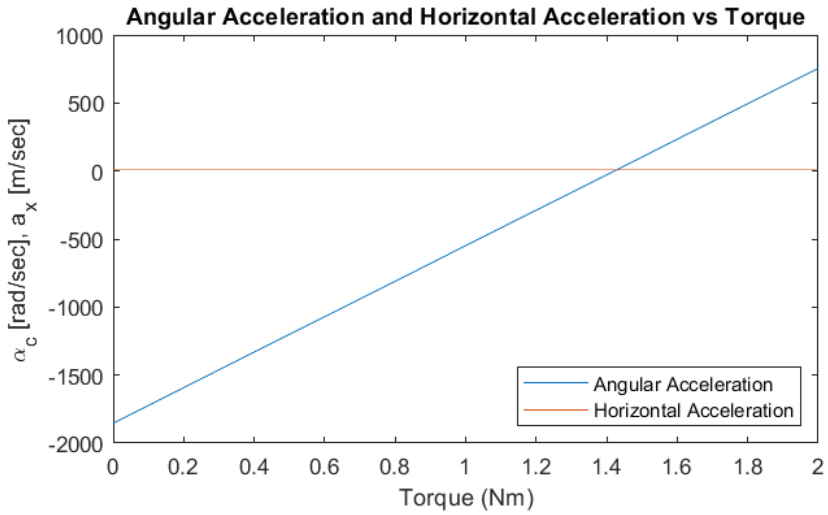


Figure 18: Lagrangian Derivation of Angular Acceleration and Horizontal Acceleration vs Torque

These results are incredibly similar to the kinematic results. Although the derivation had flaws, the results do show that there is a possibility that the assumption made thus far is correct. That being that the robot will drive or climb for low or high torques respectively.

However, in the end these results were not trusted enough to fully guide the decision on gear ratios, so the decision of gear ratios already selected by the transmission ratios calculations was kept.

4.2.4 Summary of Mathematical Model

The transmission ratio results proved to be the most reliable and all design decisions were based off the gear ratios selected from this process. These are 85mm, 55mm, and 27mm for the sun, idler and planet gear respectively.

The kinematic and Lagrangian derivations were not useful for anything other than speculation. They gave some insights, suggesting that driving and climbing were assured with low torque and high torque respectively, but this could only be confirmed through experimentation.

4.3 MECHANICAL DESIGN

The full mechanical design was completed after the gear ratios were selected. Considerations for the placement of electronics will be made to ensure everything has a place in the robot in the later design phases. Using the global variables function in Solidworks, the design was defined completely in terms of variables to allow easy variability and quick redesign according to testing results. This was in case the mathematical model was not accurate and many changes were needed on the design.

All the parts designed were intended to be made with laser cutting either 3mm hardboard or Perspex. Any shafts were designed for 6mm aluminium rod that was owned. Refer to the Bill of Materials for all components involved. The full assembly and files needed for it are in the Builder's Guide.

4.3.1 Di-Wheel Design

The Di-Wheel itself was the most intricate design of the entire robot, with many fasteners and the exact placement of the gears needing to be correct.

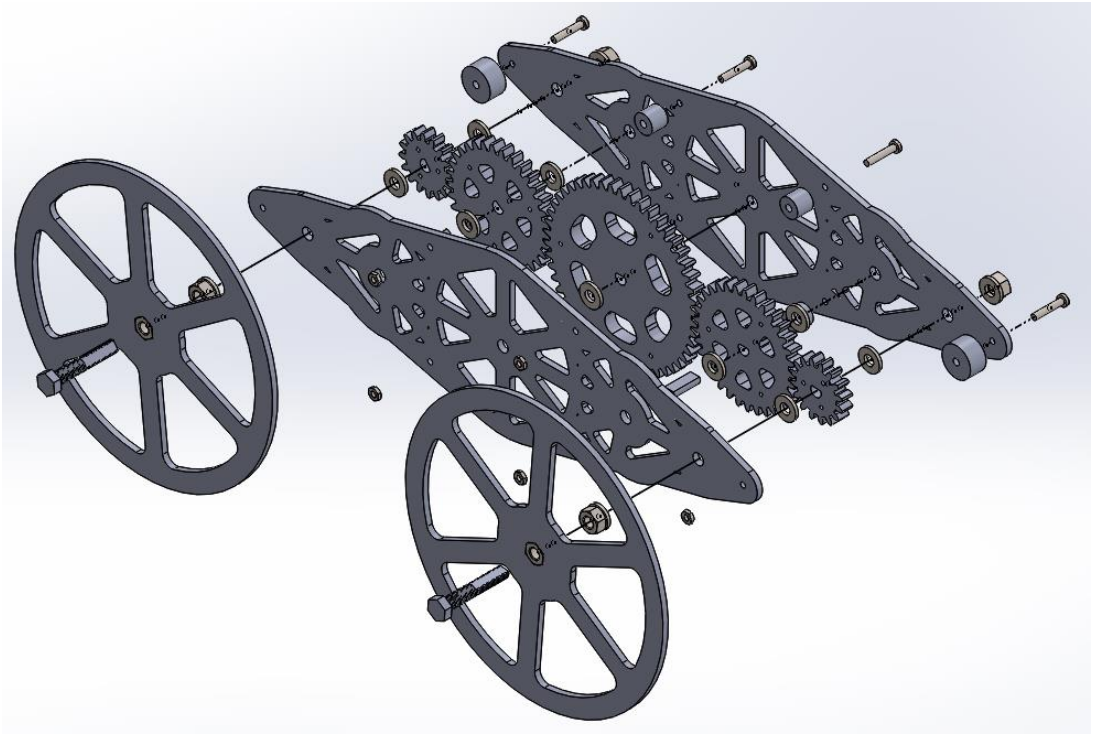


Figure 19: Full Di-Wheel exploded view (final version)

The design shown in Figure 19 is a third iteration. Iteration two added the fasteners on the end of the carrier, and iteration three increased the width of the gears after the initial design broke when too much

torque was applied. This was accomplished by gluing two 3mm gears together to increase the thickness. It was decided to glue them together rather than use thicker material because laser cutting cuts with an angle, as seen in Figure 20. This caused the gears to always sit at an angle, but by adding a mirrored gear this issue was removed.

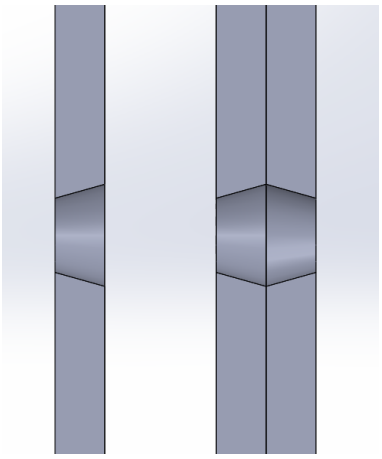


Figure 20: Exaggerated view of angled cut from laser cutter

In order to locate the gears exactly, additional holes for toothpicks were added to allow location of the gears relative to each other. This method resulted in a much less frail structure. These parts were all 3mm Perspex, except for the wheels which were hardboard.

4.3.2 Tail Design

The tail was a simple PVC 20mm wide pipe, with struts to give it support.

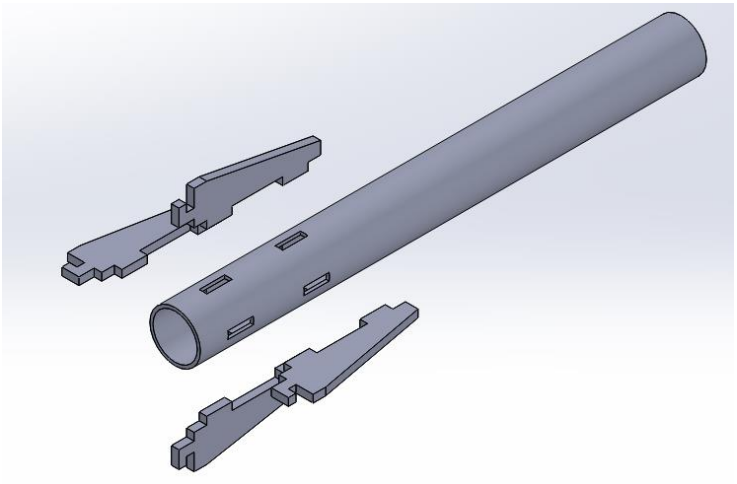


Figure 21: Tail 1.0 exploded view

Using the keys on the struts in Figure 21, the tail clipped into the back of the body. This was essential as the prototype had to be carried around in a small bag requiring a compact size.

This initial design resulted in the rear panel of the body being leveraged off if it the tail made impact with the floor. This was solved by redesigning the tail, as seen in Figure 22.

Figure 22: Tail 2.0, with spine on upper and lower body

The tail was fitted onto a spine structure on the body to distribute the force throughout the body. A squash ball was placed at the end of the tail to create a damping effect instead of the hard plastic of the tail.

4.3.3 Body Design

The actual design of the body was implemented as a simple box structure, to provide the easiest assembly possible.

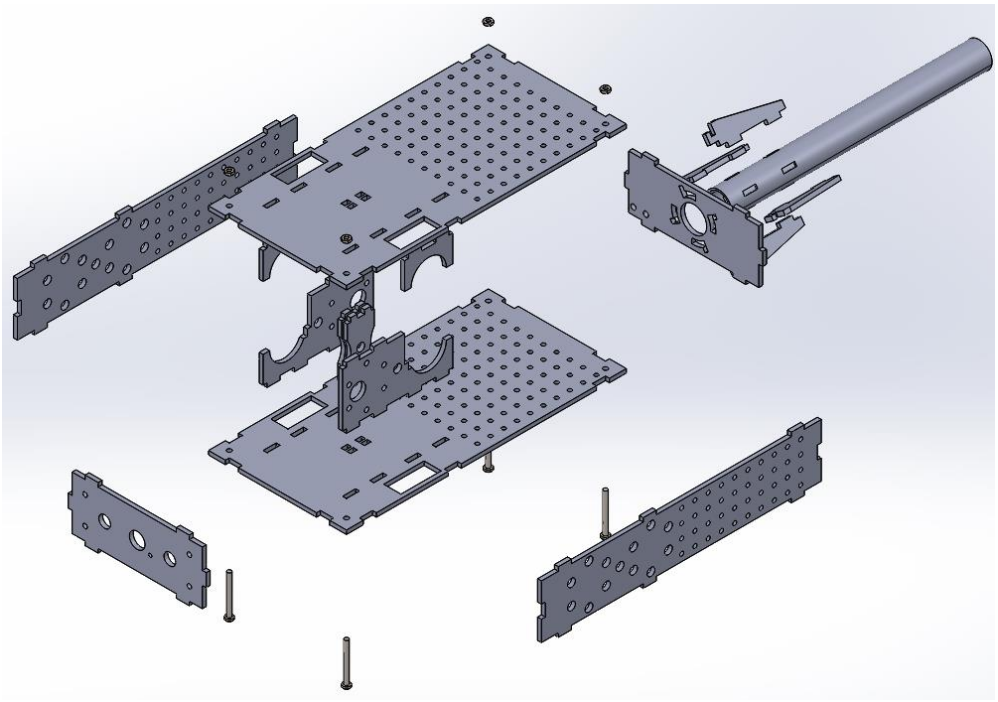


Figure 23: body Assembly Exploded View

The motors were placed side-by-side to allow a narrow body design. This also allowed the motor speed to be geared down, increasing torque output. Space was left at the back for electronics, and holes in the front for a camera and flashlight. These parts were all 3mm hardboard.

4.3.4 Integration of Phase 2 and Phase 3 Requirements into Mechanical Subsystem

In order to avoid a completely new design later in the design phase, the space considerations for the components used later were included in this initial design.

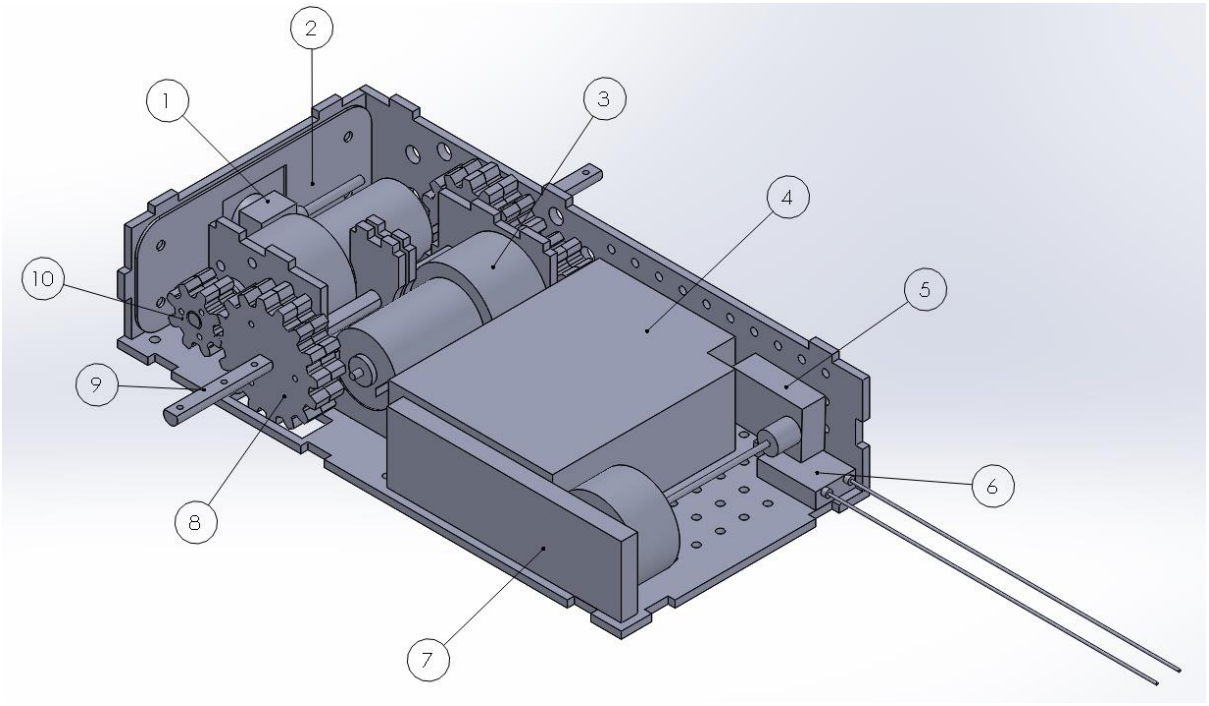


Figure 24: Placement of components in the body

(1) Camera, (2) Power LED and bracket, (3) Motor, (4) Space for general electronics, (5) FT952 5.8GHz transmitter, (6) Orange RX transmitter, (7) Battery, (8) Step-down output gear, (9) Output shaft, (10) Motor gear

As shown in Figure 24, the electronics and Orange RX transmitter required for phase 2, and the camera and FT952 transmitter for phase 3 were considered. By considering this before manufacture a lot of time was saved.

4.3.5 Summary of Mechanical Subsystem Design

The mathematical models were derived systematically using kinematic and Lagrangian mechanics, but these proved overly complicated. The transmission ratio derivations provided the best direction in choosing gear ratios. It was decided to use an initial prototype design to confirm if these ratios were correct.

There were several lessons learned from the initial build that lead to an improved design. These are:

- Need stronger gear teeth to withstand torques.
- Need wider gears to make structure more rigid.
- A spine is needed to distribute the force from the tail

Figure 25 shows the final and complete structure of the robot. It was designed purely for the flipping over motion of the Di-Wheels, with ease of manufacture and assembly/disassembly in mind.

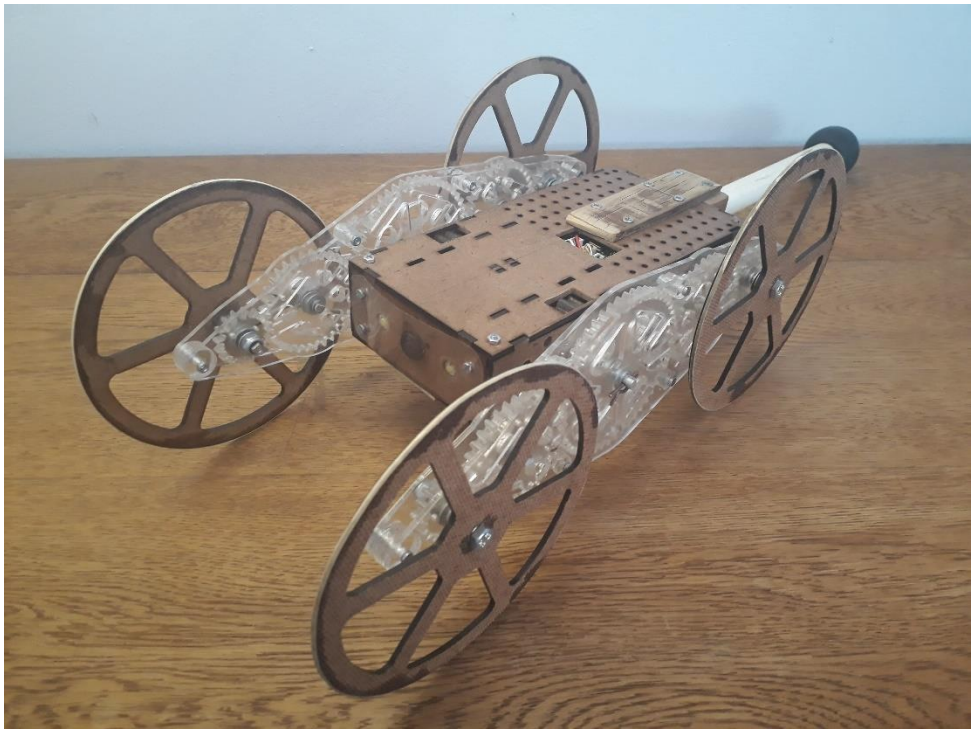


Figure 25: Final assembly of robot

The design was simplified so that anyone can build it using the template provided in the builder's guide in Section 10.6 of the Appendix.

4.4 Motor Selection

The extent of the electronic system in this design phase is tethered power to the motors. The motors used were EGB-380S, which are 12V, 270 RPM with a rated tolerance torque of 2kg.cm. The reason for these motors are that they were already owned and being unsure of the strength required, it was decided to use them and determine if stronger motors would be needed later. This was not the case as they proved strong enough.

4.5 Testing of Design Phase 1

The testing procedure of Design Phase 1 was performed after the full mechanical assembly was complete, and the motors had been installed with a 2.5m tether.

Initial tests were performed by gradually increasing the step input voltage applied. The maximum applied voltage was 12V as this is the voltage rating of the motors.

Step input voltage	Results
6V	Barely moving
8V	First test: both motors on same voltage supply, drove forward 5cm. Second test: motors on separate voltage supply, successful forward driving.
10V	Successful forward driving, but maxed out current supply at 1A. Could not climb step (assumed due to 1A current limit). Higher voltage results in tendency to try climbing before driving.
12V	The same result as for 10V.

Table 7: First Phase 1 tests results

The power supply used was current limited to 1A. This resulted in the motors being unable to climb a step. The Di-Wheel proved to be quite shaky, and the gears did not sit straight on the shaft which affected performance. As outlined in Section 4.3.1 the Di-Wheel was redesigned with thicker gears, and Perspex rather than hardboard sides to reduce friction.

The motors were unevenly balanced in power use. One motor tended to push first which caused the left Di-Wheel to start climbing. This resulted in the other motor being unable to climb as all the weight was resting on it alone. The robot would then tip over. This is shown in Figure 26.

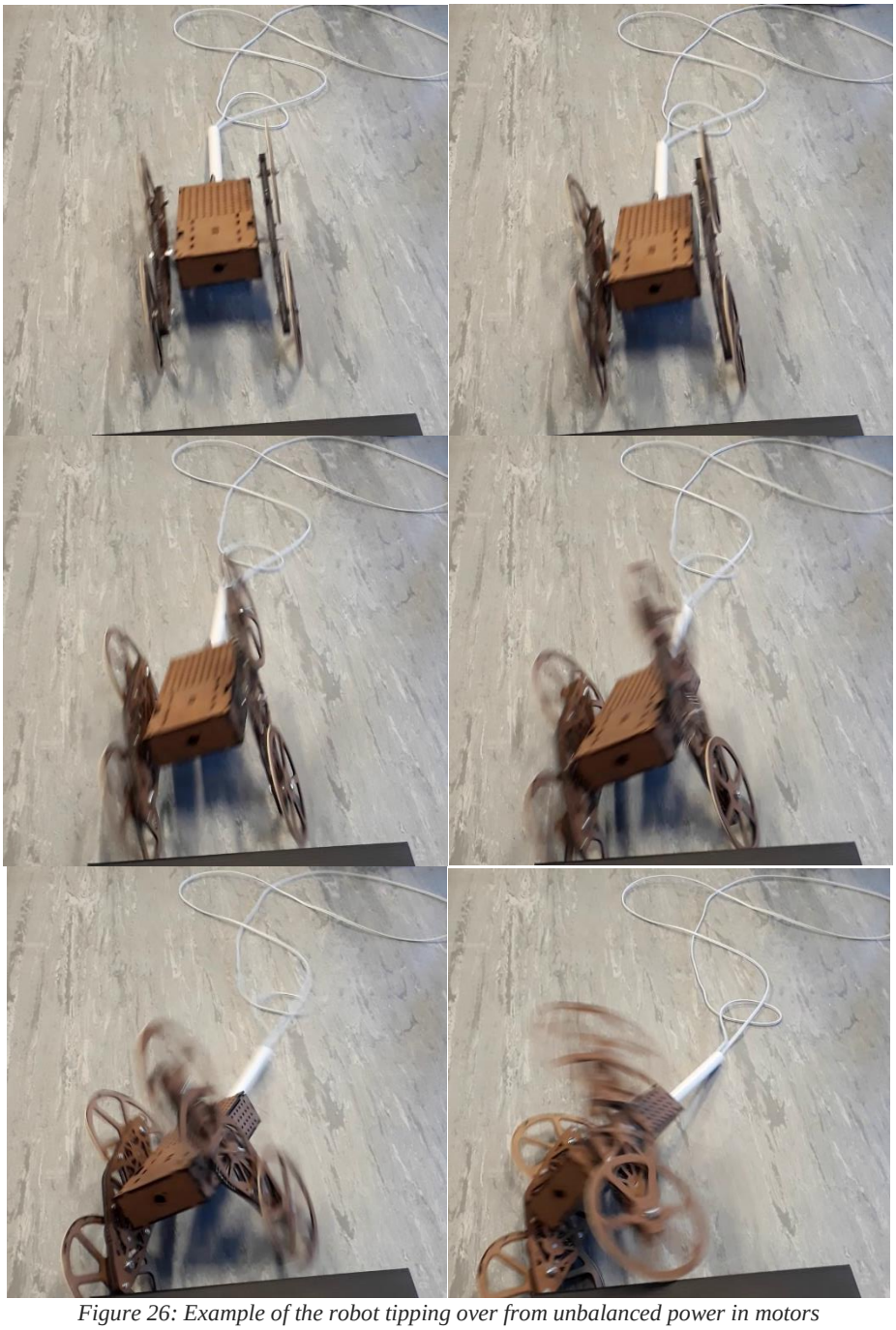


Figure 26: Example of the robot tipping over from unbalanced power in motors

A potential solution to this is discussed in the Summary of Design Phase 1.

Secondary tests were performed with a 6A power supply on the new Di-Wheel design, for which the results were more promising.

Step input voltage	Results
6V	The robot drove very well.
8V	The robot drove very well. Nearly able to climb over a 12cm step.
10V	The robot drove very well. Successfully able to climb a 12cm step.
12V	The full voltage being applied shattered the motor shaft Perspex gears.

Table 8: Second Phase 1 tests results

With the new Di-Wheel design, and the higher current supply, the robot was able to successfully climb over a 12cm step. The maximum current draw was 3.5A total, 1.75A per motor.

The Perspex motor shaft gears shattered with a 12V step input, which prompted a redesign of these gears. This is shown in Figure 27.

Figure 27: Old broken motor gear vs Redesigned motor gear

This new design proved strong and reliable, with fewer and thus larger teeth, as well as making the gears 12mm wide by using 4 gears combined into one (3mm per gear).

4.6 SUMMARY OF DESIGN PHASE 1

Design Phase 1 produced the mechanical design of the robot. After connecting the motors and performing driving and climbing tests, the following important information was gained:

- The gear ratios selected resulted in the desired climbing motion when a step was encountered.
- The gear design was made from two layers of 3mm Perspex to increase strength and reduce friction.
- The body was made from 3mm hardboard which proved useful as it was easily altered.
- Fins on the tail proved to be inadequate for connecting the tail to the body. Instead a spine setup was created, and the tail was press-fit into this.
- Step inputs to the motors are bad for the gears. A simple controller can be implemented to ramp the motor duty cycle to the desired value.
- The maximum current drawn when climbing was 3.5A.

A few solutions are described below in order to overcome the unbalanced motors;

- Favour the weaker motor in software – by giving each motor a predetermined ratio of power, the weaker one can be favoured. This proved less effective after testing this theory later, no definite ratio was found.
- Control theory using current sensing – by sensing the current in each motor, a controller can be implemented to equalise their strength.
- Control theory using an accelerometer – by measuring the body's orientation, a controller can be implemented to equalise their strength.

Therefore, Design Phase 1 shows a successful design of the mechanical aspects of the robot, as well as providing initial proof that the Di-Wheel concept will produce the stair climbing motion desired.

5 DESIGN PHASE 2: UNTETHERED DRIVING AND CONTROLLER

The extent to which The Di-Wheel robot was projected to work by the completion of this phase was to be untethered, with basic controller functions to increase or decrease power in each of the motors independently. This was accomplished with a battery power supply, along with a controller. The electronics subsystem can be broken down as seen in Figure 28.

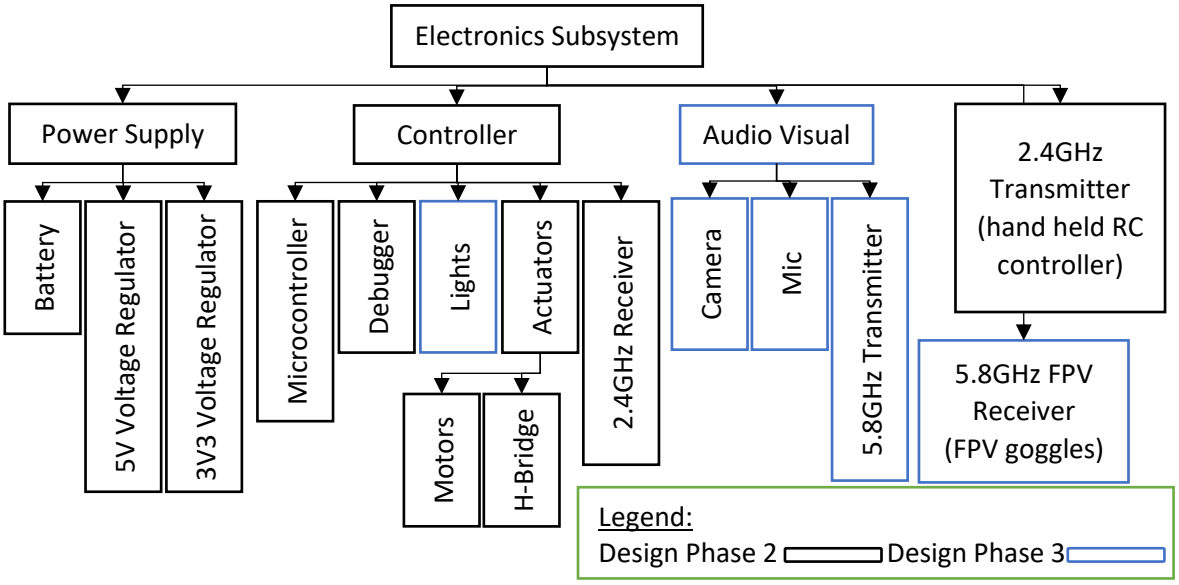


Figure 28: Electronic subsystem breakdown

Design phase 2 and 3 requirements were considered such that any supporting circuitry required for these components would be included in the electronics design. The circuit layout that incorporates these various components was initially sketched as a block diagram, as seen in Figure 29.

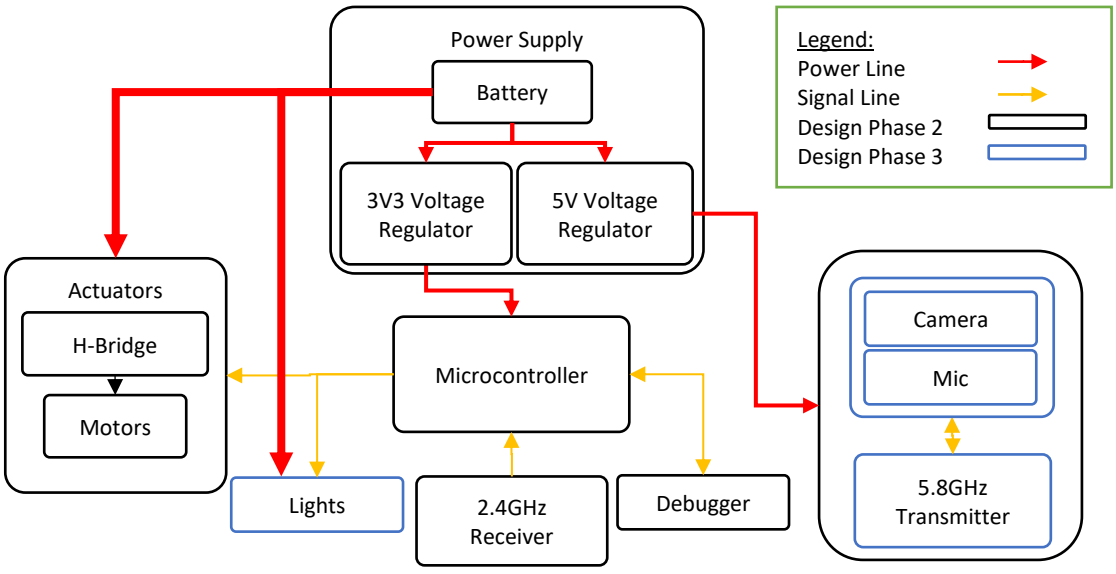


Figure 29: Electronics circuit initial block diagram

The detailed circuit design for these subsystems will be described in Sections 5.1 and 5.2, followed by the software development in Section 5.3

5.1 POWER SUPPLY DESIGN

The power supply consists of a LiIon battery, with a 5V and 3.3V regulator. The voltage and current requirements for each component were analysed below. This includes Design Phase 3 requirements. All current values are at their rated voltage.

Component	Battery Voltage 16.8V	5V	3V3	Typical Current Draw	Climbing Current Draw
STM32F051C6			X	0.12A [17]	-
ST Link Debugger		X	X	0.15A [18]	-
Lights	X			0.9A [19]	-
H-Bridge and motors	X	X		1A	8.2A from calculations, 3.5A from tests
2.4GHz receiver			X	0.035A [20]	-
Camera/mic		X		0.08A [21]	-
5.8GHz transmitter	X			0.12A [22]	-
			Total	2.405A	9.6A from calculations, 4.905A from tests

Table 9: Voltage Levels and current budget per component

The current draw for the motors was calculated by dividing the battery voltage by the motor winding resistance. The average winding resistance was 3.17Ω which would draw 4.1A per motor at 13V (if the batteries are fully charged the motors will receive about 13V, refer to Section 5.2.3 for more detail). However, during Phase 1 testing it was seen that the current draw from the motors was never more than 3.5A, even when the motors were completely stalled.

This difference may be from the tether's resistance. This was measured at 0.8Ω total, resulting in 6.5A total. The decision was made to work with the current draw from calculations. So a power supply rated at 10A was designed.

5.1.1 Battery Selection

In order to provide the current required, at the voltage required, at an affordable rate it was decided to use four 18650 cells to make up the robot battery. There are several advantages to this:

- Less expensive than LiPo
- Large capacity (3000mAh)
- Ubiquitous, they can be obtained easily
- Less likely than LiPo to explode

The exact 18650 cells purchased were rated at 10A max current draw, and for a four cell battery at full charge (4.2V each) the battery would provide 16.8V. As the H-Bridge was expected to drop 4.9V maximum, this leaves 11.9V for the motors as a worst case at full charge.

5.1.2 Voltage Regulator Circuit

The voltage regulator circuit was a standard set up, using LM317T voltage regulators as these were easily accessible in the White Lab store and are reliable enough for the purposes of this project. Figure 30 shows the circuit diagram designed.

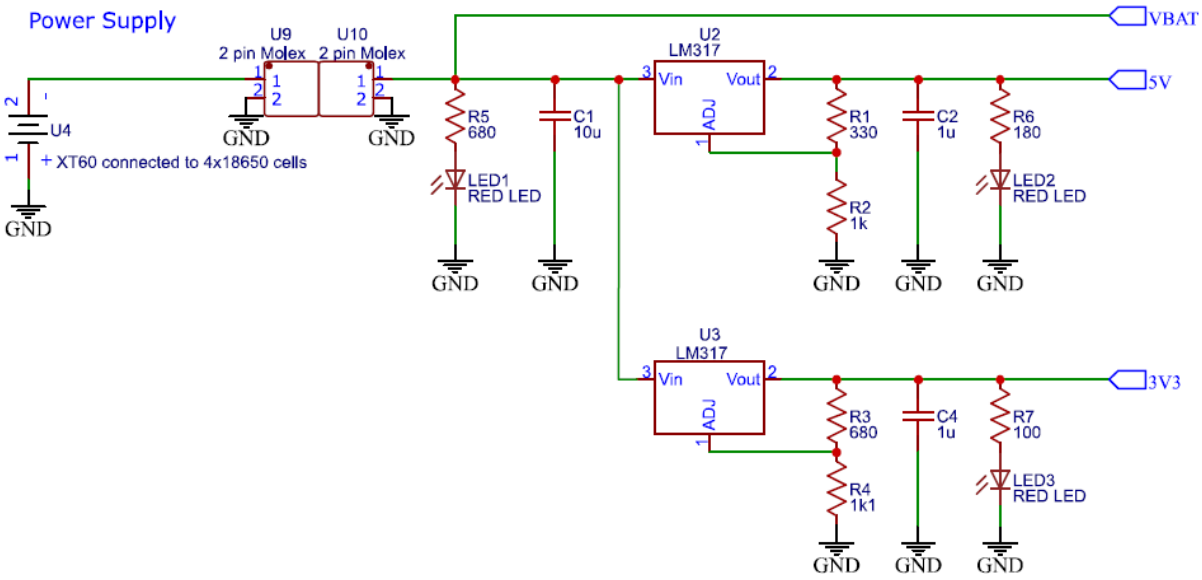


Figure 30: Power supply circuit diagram

The resistor values were calculated as follows:

$$V_{out} = 1.25\left(1 + \frac{R_2}{R_1}\right) + I_{adj}R_2 = 1.25\left(1 + \frac{R_2}{R_1}\right) + (50 \times 10^{-6})R_2$$

$$R_1 = \frac{1.25R_2}{V_{out} - 50 \times 10^{-6}R_2 - 1.25}$$

For 5V, and $R_2 = 1k\Omega$, $R_1 \approx 330\Omega$.

For 3V3, and $R_4 = 1.1k\Omega$, $R_3 \approx 680\Omega$.

These values worked well, producing the voltage required, but the 5V supply was later changed to use a fixed regulator, the 7805 5V regulator. This is simple to wire up as shown below.

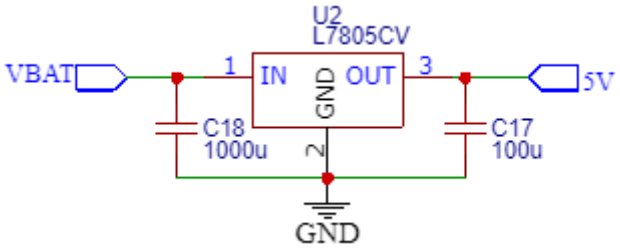


Figure 31: 5V fixed regulator circuit

The reason for this redesign was to provide a more stable 5V supply to help prevent noise from the motors. The Veroboard design for the power supply circuit is shown in Section 5.2.4.

5.2 Controller Design

The controller sub-subsystem consists of an STM32F051C6 as a microcontroller, a debugger for ease of programming, a 2.4GHz OrangeRx R617xl receiver and Spektrum DX8, and an L298 H-bridge with the motors wired up. The transmitter used to communicate with the receiver was a Spektrum DX8. Refer to the Bill of Materials for more details.

These components were used because they were already owned. Other components were investigated, as shown in Section 10.6 of the Appendix. Brief reasons why they were not purchased are explained in the following sections.

5.2.1 Micro-Controller Circuit

The microcontroller selected was the STM32F051C6. This was selected because it was easily accessible, which was desirable in case it blew. Figure 32 shows the microcontroller wiring and set up required for an external 8MHz clock, as well as a reset pushbutton and $10k\Omega$ resistor pulling the BOOT0 pin to ground.

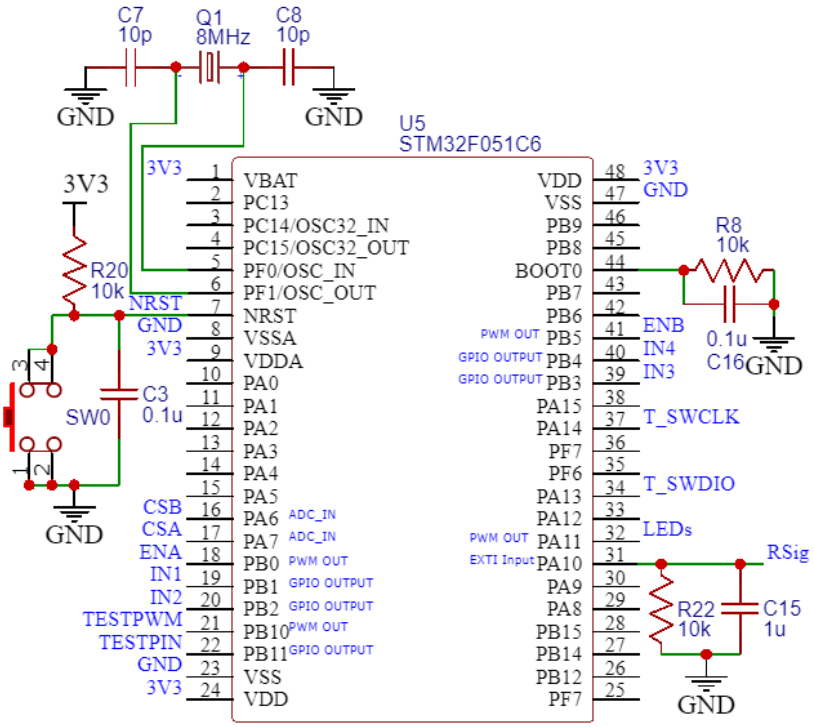


Figure 32: Microcontroller pinout and connections

The pull-up/ pull-down resistor and decoupling capacitors were added as a result of noise from the motors triggering the reset switch and corrupting the receiver signal. The various pins used to connect to other components are explained in more detail in the software section (Section 5.3).

An onboard debugger using ST-Link was used to allow quick updating of onboard software. The circuit diagram for this is shown in Figure 33.

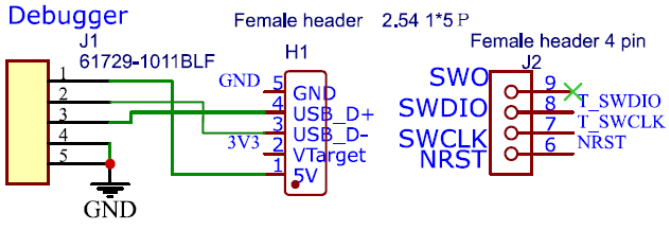


Figure 33: Debugger pinout and connections

The connector type seen in Figure 33 is a USB connector. The nets T_SWDIO and T_SWCLK connect to the microcontroller on the required debugging pins as seen in Figure 32. The Veroboard design for the Controller circuit is shown in Section 5.2.4

5.2.2 2.4GHz Receiver circuit

The 2.4GHz receiver only required power and provided a signal to the microcontroller. The selection of which receiver to use required some thought because there are many different types with different transmitter (Tx - transmitter to receiver) and receiver (Rx - receiver to microcontroller) protocols.

There are many types of protocols, and a summary of them is given in Section 10.6.1 of the Appendix. The RX protocol that was desired was Pulse Position Modulation (PPM) as this would simplify connections needed. The TX protocol was only needed in matching a transmitter and receiver. The OrangeRX R617xl receiver used PPM, and the Spektrum DX8 was compatible with the R617xl, so these were selected. This was convenient as they were already owned.

Figure 34: Spektrum DX8 transmitter and Orange RX R617xl receiver

Figure 35 shows the exact connections required for the receiver.

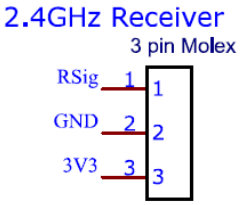


Figure 35: 2.4GHz receiver pinout and connections

The exact meaning of PPM is explained more in depth in the software section (Section 5.3.1).

5.2.3 Actuator Circuit

A high power H-Bridge was required for the bi-directional control of the motors. The L298 was considered, as it is rated at 25W [23]. With a current draw of 8.2A, and a volt drop on the H-bridge of 4.9V, the h-bridge would need 40W. This would have required the construction of a FET h-bridge, for which there was no time. Considering the max current draw was seen to be 3.5A, this would need 17W which the h-bridge could provide. To test if this would work, a readily available L298N module [24] was tested.

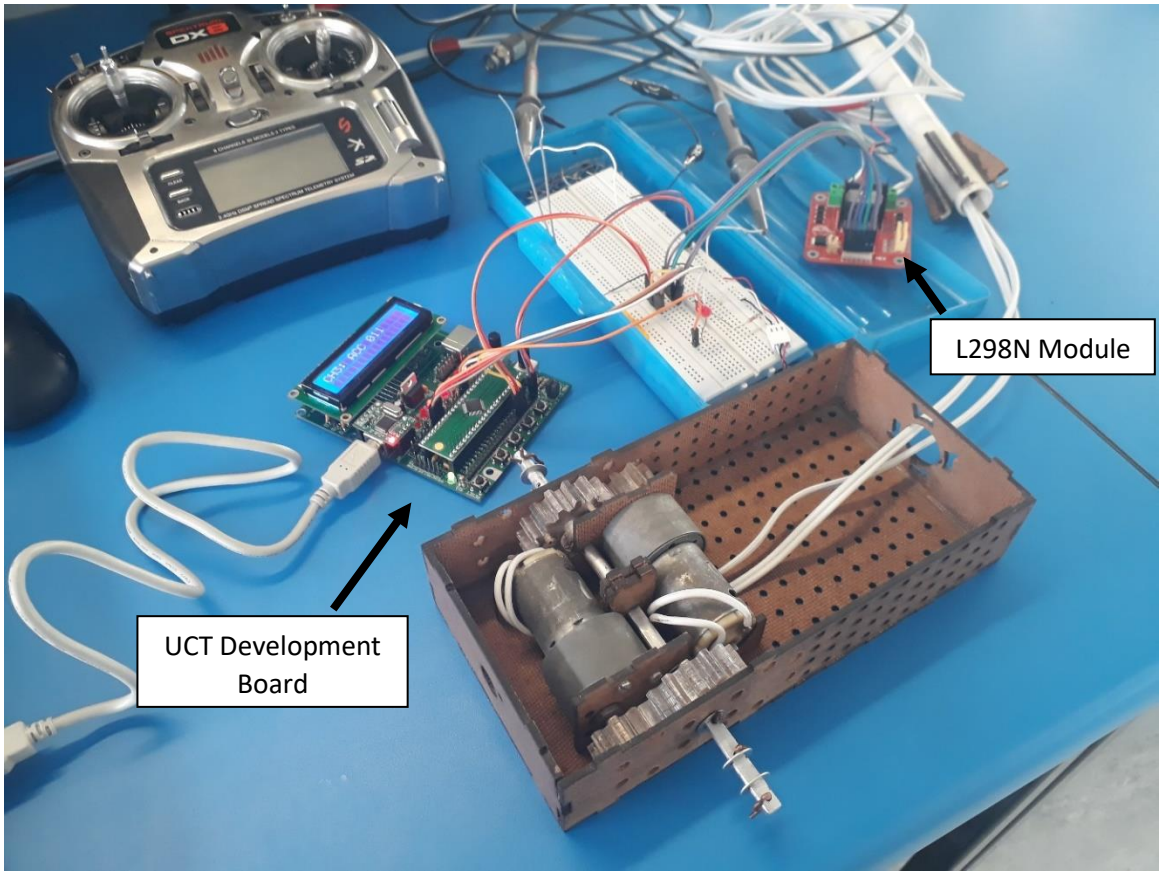


Figure 36: Testing set-up for L298N H-Bridge Module

The test was purely to see if the module could power the robot sufficiently for climbing motion (i.e. 1.75A consumed per motor). These tests proved successful, with the H-bridge getting only a little warm during climbing. The results for these tests are show in Section 5.5.3.

The module was too large and tall (from the heatsink) to be used in the body designed, so it was decided to design a circuit on a Veroboard according to space requirements. The circuit used for the H-bridge, still using the L298N component is shown in Figure 37.

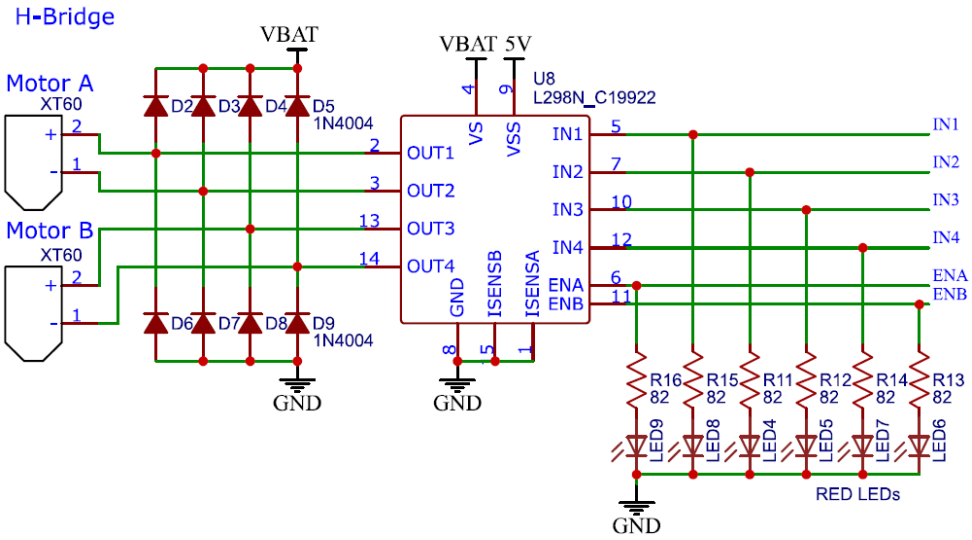


Figure 37: H-Bridge circuit diagram

This h-bridge was not ideal, and a higher power one would be needed for future development. But for proof of concept it was decided to use what was available and make it work. The Veroboard design for the H-bridge circuit is shown in Section 5.2.4

5.2.4 Full Circuit Design

The full circuit design can be found in Section 10.4 of the Appendix. The power supply and controller circuit were designed on Veroboard, specifically to fit into the body of the robot around all the other components. This design can be found in Section 11.6 of the Builder's Guide.

The lessons learned from this Veroboard design are summarised below:

- The USB cable should rather stick out the back of the box (away from the wheels) as this provides more space for other components and it is easier to program without needing to remove the lid.
- A PCB design would definitely be easier to implement, as the board took a lot longer to build due to its complexity, however this can only be done once all the problems from the motor noise in the circuit are cleared. Refer to Section 5.4 for more detail.

5.2.5 Summary of Controller Design

The design of the controller was relatively straightforward with few problems to consider as the methods used are well documented. The microcontroller and its connection to the 2.4GHz receiver and the H-bridge were designed in this section.

5.3 SOFTWARE DEVELOPMENT

The software was used to interface the controller via the receiver with the motors and flashlight. All software is available in the Di-Wheel Google Drive. Before developing the software for the robot, the communication channels and protocols used between components had to be determined. This is summarised in Figure 38. The flashlight was developed in design phase 3, and so the software for it is discussed here.

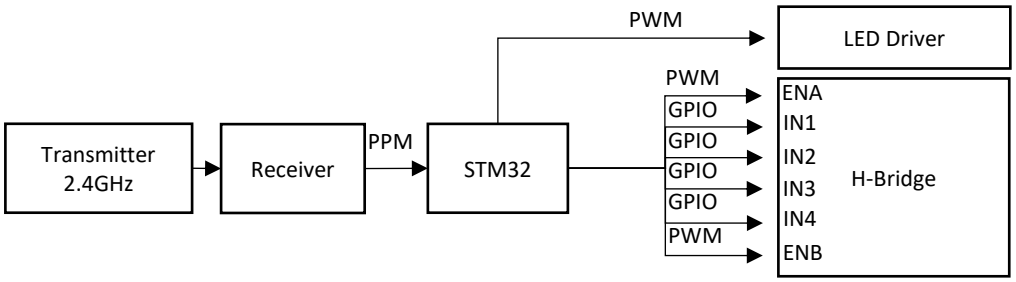


Figure 38 Communication channels and protocols between components

The communication between the transmitter (Spektrum Dx8) and receiver (OrangeRX R617XL) is DSM, which is proprietary to the Spektrum brand [25]. The OrangeRX is compatible with Spektrum products.

The communication between the receiver and the microcontroller (STM32F051C6) is of more concern to the software. The receiver uses PPM (Pulse Position Modulation), which is a universal communication protocol (non-proprietary) between receivers and microcontrollers [25]. This is explained in depth in Section 5.3.1.

The instructions to the motors, communicated via the H-Bridge channels, are shown as ENA/B (enable inputs for motors A and B), IN1/2/3/4 (control inputs 1 through 4). The truth table for the enable and control inputs is shown in Table 10.

EN	IN1	IN2	Function
1	0	1	Forward
1	1	0	Reverse
1	1	1	Fast stop
1	0	0	Fast stop
0	X	X	Free running stop

Table 10: Truth table for L298N H-Bridge instructions

The enable pins are used as the PWM channels to set the speed of rotation, and the input pins are purely GPIO digital channels to set the direction of rotation.

The software for receiving communication from the transmitter, and controlling the motors and lights follows the flow diagram in Figure 39.

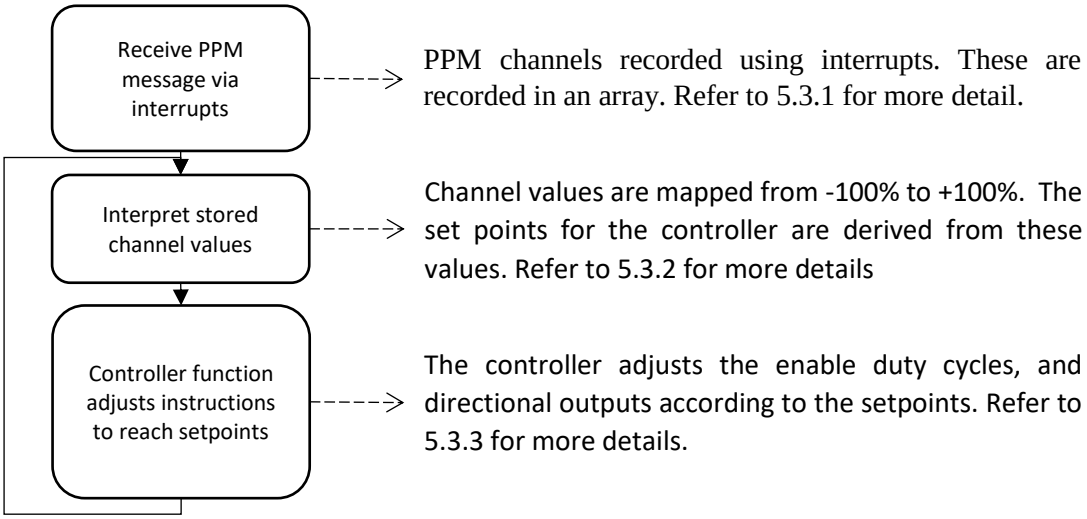


Figure 39: Summarised flow diagram for software on microcontroller

To fast track the project, Atollic TrueStudio and CubeMX was used to autogenerate the initialisation code for the STM32F051C. Several tutorials were used to become familiar with the software [26], after which the following components were set up.

- PWM output pins [27] [28]
- GPIO output pins [29]
- Initialising interrupts [30]

After the code was initialised with the required pin modes set up, the actual software could be written.

5.3.1 Reading PPM

Pulse Position Modulation is a simple form of communicating the values of channels using constant width pulses. The time difference between pulses specifies the exact values of channels [31]. There is a time delay between the final channel pulse and first pulse of next message to separate messages, this is shown as channel 0 in the example shown in Figure 40.

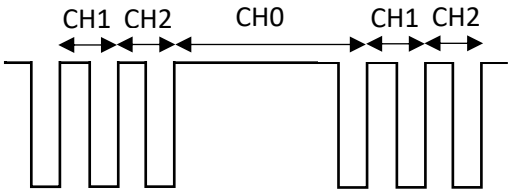


Figure 40: Example of a two channel PPM signal

Using either rising or falling edge triggered interrupts, the time instances of each pulse can be recorded and thus the values of each channel by comparing times.

Synchronisation can be a concern [32]. If the microcontroller start-up time is longer than the receiver, and the first interrupt occurs after the beginning of the message, then the values of the channels will be constantly out. This is shown in Figure 41.

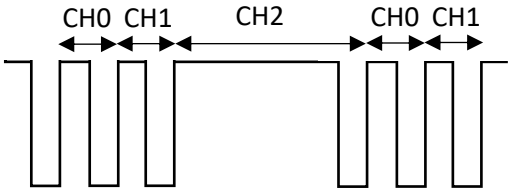


Figure 41: Example of unsynchronised recording of PPM message

This can be fixed by including a synchronisation function that resets all the channels to zero when a CH0 message is received. From that point on it will record CH1, CH2 and so on.

The final flow diagram for the interrupt handling function to record channel values is shown below in Figure 42.

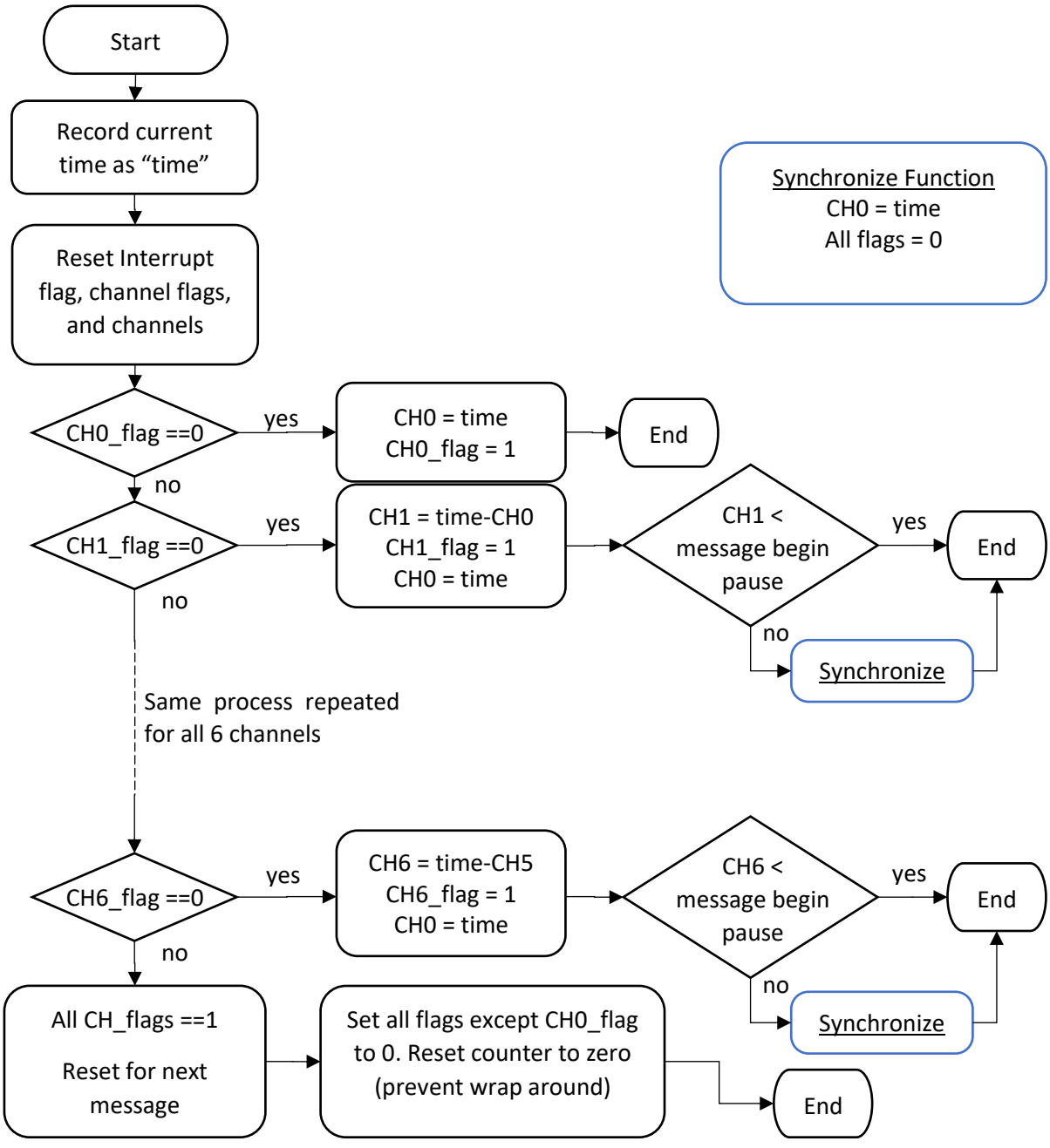


Figure 42: Final flow diagram for interrupt handler function to record PPM channels

The code for the flow diagram in Figure 42 can be found in Section 10.7.1 of the Appendix.

5.3.1.1 Overcoming Low Resolution System Time

Using an oscilloscope, the expected PPM channel lengths were 16ms for CHO and 1.2-2.2ms for all other channels, with the zero position at 1.7ms. The system time on the STM32F051C6 only records time at a resolution of 1ms, so a higher resolution counter was created.

Research showed that it was impractical to increase the system clock resolution, and so a simple counter was set up using one of the system timers, and an interrupt on every pulse to iterate a counter value [33]. This method was preferred as it was customisable and could be reset to ensure there was never a wrap around of values. The final resolution was two decimal places, showing 1.20-2.20ms for each channel.

5.3.1.2 Optimising the Interrupt Handler Function

The initial code for the interrupts was inefficient, using up all of the system resources, causing missed interrupts. The interrupt handler was simplified to the one described above. It originally did the interpret and controller functions too (described below). In order to test that the problem was resolved a test GPIO output pin was set for the duration of an interrupt. Figure 43 shows the result of this test case.

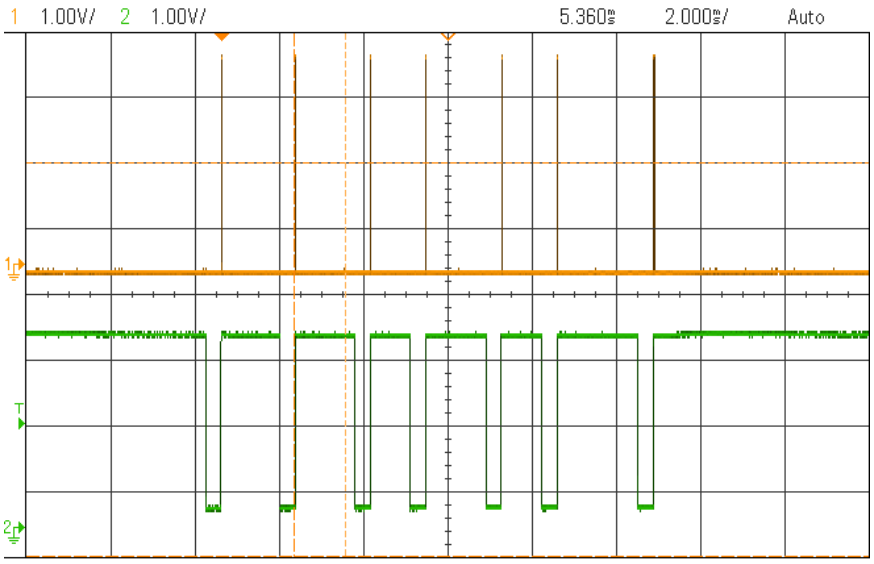


Figure 43: Testing duration of interrupt for each PPM channel

Oscilloscope channel 1 shows the duration of each interrupt and channel 2 shows the PPM message. As can be seen, the interrupts were no longer taking a long time, and the system ran smoothly hereafter.

5.3.1.3 Overcoming Distorted Signal Values

The values read straight from the receiver were not steady, as was evident later in the interpret code. This was concerning, especially when the signals should have been constant. It was thought the timer was not a high enough resolution and noise was causing these variations, so it was increased to two decimal places. This did not fix the problem.

After analysing the software it was realised that the channel values were being set to zero at the end of each message. This is when channel flags were introduced, as described above. The flags were set when a channel was read and reset at the end of a message. Using channel flags solved the distorted signals problem.

5.3.2 Interpreting signals

The interpret function was used to map the channel values to the correct format for the controller function. As only three channels are needed, these are designated as the power (forward or reverse), turning and light channels. Each channel had a designated deadband created in the code to ensure that a small move, or noise, would not result in unexpected motion.

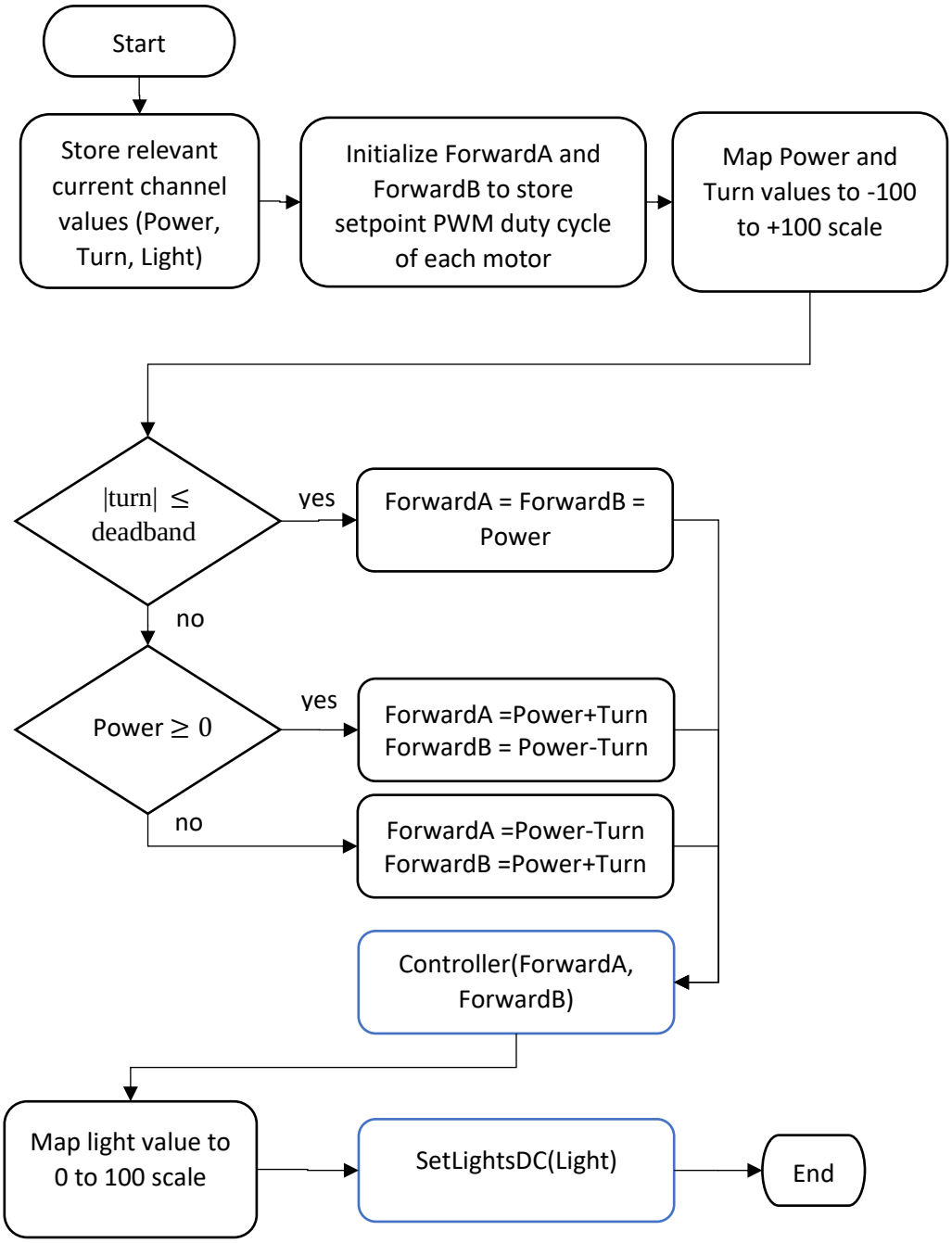


Figure 44: Flow diagram of Interpret function, used to interpret commands and create setpoints for controller function

The code for the flow diagram in Figure 44 can be found in Section 10.7.2 of the Appendix.

ForwardA and ForwardB in the diagram above are the duty cycle setpoints for motor A and B respectively. These values can range from -100% to +100%, with negative referring to a reverse direction. The controller interprets these values as described in Section 5.3.3.

The mapped Light value was simply put straight into the SetLightsDC function, which set the PWM duty cycle of the flashlight signal to be equal to the mapped value.

5.3.2.1 Mapping Values

Before doing anything, the channel values had to be mapped to a percentage scale. In order to this, testing software was set up to read off an LCD the recorded integer value from the timer created corresponding to the channel values in seconds. The values measured, and the mapped value is shown in Table 11.

	Actual value (ms)	**Recorded integer for channel**	**Mapped value**
Minimum	1.2	106	-100
Zero position	1.7	146	0
Maximum	2.2	186	100

Table 11: Channel values in milliseconds, recorded value, and mapped value for interpreter

The mapping of the recorded integer to the final mapped result was done using the following formula

$$power\ mapped\ value = \frac{(CH_{power} - 146) \times 100}{(186 - 146)} \rightarrow -100\% \text{ to } 100\%$$

$$turn\ mapped\ value = \frac{(CH_{turn} - 146) \times 100}{(186 - 146)} \rightarrow -100\% \text{ to } 100\%$$

$$light\ mapped\ value = \frac{(CH_{light} - 106) \times 100}{(186 - 106)} \rightarrow\ 0\% \text{ to } 100\%$$

It is noted that the light channel was mapped from 0 to 100, as it was not logical to have any negative values.

5.3.2.2 Method of Interpreting Signals for Controller

There were several options for using the channel values to calculate the set points for the motors.

1. Multiply Power and Turn

The first option was scrapped immediately as it would not result in the desired motion.

2. Mapping Matrix

The mapping matrix was developed by analysing the Power and Turn values in relation to each other and the states they resulted in. A diagram showing the mapping matrix is found in Figure 45.

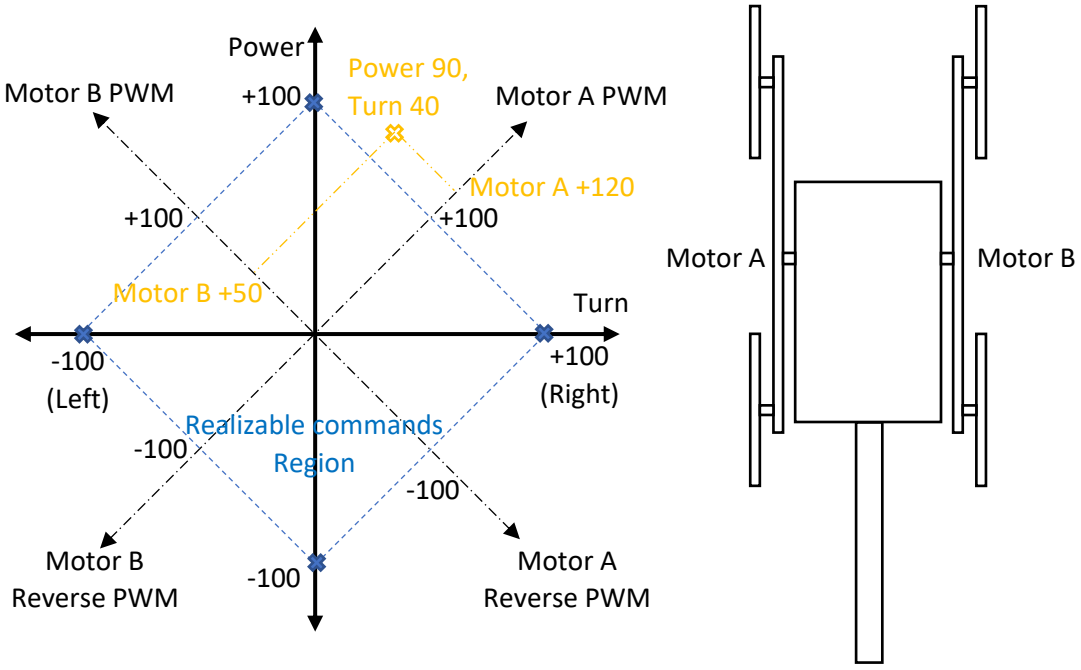


Figure 45: Mapping Matrix shows the Power and Turn value relationships to the PWM duty cycle on each motor

The mapping matrix analyses the expected motor rotations for a given set of commands. For example, when full forward power and zero turning power is given, it is expected that each motor should get equal power in the forward direction. As turning power to the right is given, it is expected that Motor A will turn faster and Motor B will turn slower and the reverse for the left direction.

The initial calculations for this were done using a rotational matrix, using Turn as X, and Power as Y, it was rotated using a 45^{o} rotational matrix with the new X being Motor A PWM, and the new Y being Motor B PWM. This would work quite well, and by analysing the matrix it is seen that the motion would be intuitive, except in reverse where the results would have to be inverted as shown in the diagram.

3. Power Distribution

The third option was derived from the Mapping Matrix to reduce complexity by assuming Power is the actual available power and Turn describes the distribution between the two motors. This resulted in lower power as each motor essentially got a share of 100% power, meaning they only had full power when the other motor was off.

4. Differential Power Distribution

The final option redefined the available power in as 200%, and the Turn value distributed this as before. The motors could not operate above 100%, so their value would cap off at 100% (as seen by the realisable region in the mapping matrix) but the desired motion would occur as the other motor would decrease with respect to it. This is all in line with what happens in the Mapping Matrix, but is done with much simpler calculations. The motion still had to be inverted when power was in reverse to give intuitive motion.

These options had various advantages and disadvantages, which are described in Table 12.

Table 12: Various options for interpreting Turn and Power values

Option for interpreting channel values	Advantages	Disadvantages
Multiply Power (forward and reverse) and Turn (left and right)	Simple	Only allows motion if both turning and moving forward, because if any value is zero then the result is zero.
Mapping matrix (described below)	Would result in intuitive motion	Overly complicated
Power Distribution	Works quite well	Complex as the maths would have to scale values differently otherwise you only have 50% power in each motor when not turning.
Differential Power Distribution	Works well, and it intuitive	There needs to be an if statement to invert action if driving in reverse to keep motion intuitive to user.

The final option was selected as it was simplest to use and resulted in fewer computations and guaranteed intuitive motion for the user. This is the method shown in the Interpret Function flow diagram in Figure 44.

5.3.3 Controller for Motors

The testing in Design Phase 1 showed that step inputs to the system resulted in broken gears, so a controller is needed to ramp the motor PWM duty cycles up to the setpoint given by the interpret function. The exact flow diagram of the desired controller is shown in Figure 46.

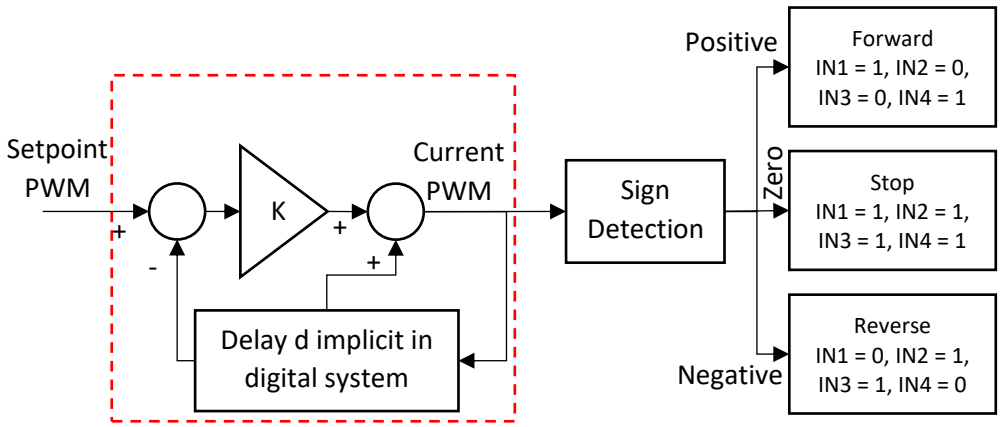


Figure 46: Controller diagram for protecting motors against step inputs

The code for the controller diagram in Figure 46 can be found in Section 10.7.3 of the Appendix.

This is an unconventional control method which merely adds a small value that is proportional to the difference between setpoint and the current PWM value. This method was chosen because it is simple and effective to implement. There is a delay implicit in the system, and this can be used to the advantage of the controller as a delay will not need to be implemented.

It must be remembered that the PWM value can be negative, as this shows reverse rotation.

To determine the optimum gain, the section in the red block is used below:

$$K\big(x(t) - y(t-d)\big) + y(t-d) = y(t)$$

$$Kx(t) + y(t-d)(1-K) = y(t)$$

$$KX(s) + se^{-ds}Y(s)(1-K) = Y(s)$$

$$KX(s) + \left(-\frac{s-\frac{2}{d}}{s+\frac{2}{d}}\right)Y(s)(1-K) = Y(s) \text{ using Pade Approximation}$$

$$H(s) = \frac{Y(s)}{X(s)} = \frac{\frac{K\left(s+\frac{2}{d}\right)}{(2-K)}}{s+\frac{2k}{d(2-K)}}$$

To determine a stable region:

$$s + \frac{2k}{d(2-K)} = 0$$

$$s = \frac{-2k}{d(2-k)}$$

Therefore, for $0 < K < 2$ the system would be stable as s will always be negative. $K = 0.25$ was selected as an initial test, as this would provide the largest step deemed allowed in the worst case (for +12V to -12V, 0.25 would result in the largest step being 6V). As we don't want it to take too long either, its response time should be within 0.5-1 second. Therefore the final requirements are:

$$0 < K < 0.25$$

$$0.5s < \tau < 1s$$

Performing a root locus on this unconventional control method was only possible by calculating the *s*-value for a given array of gain values. The implicit delay chosen was 0.1ms.

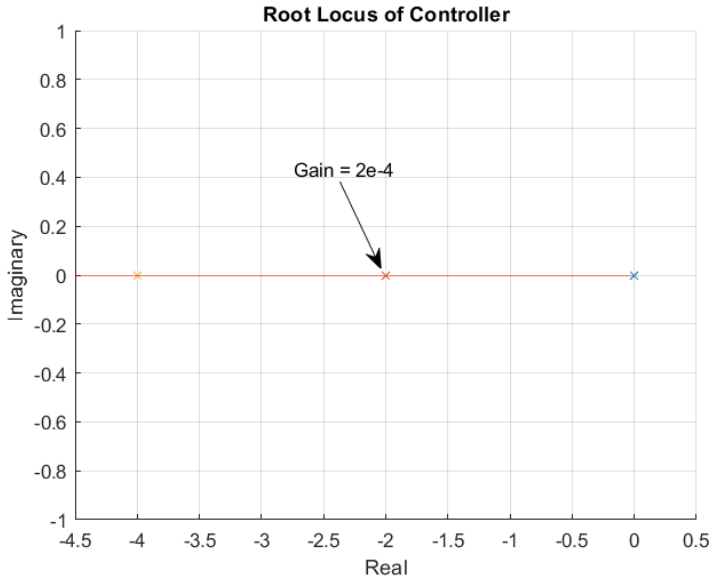


Figure 47: Root locus of controller solution

The root locus shows that the gain required for the desired time constant is 2e-4. After actual implementation this was found to be far too small, and the value of 0.15 was found to be optimal.

The sign of the next PWM value is used to determine the direction of the motors, and then the enable pins and control pins are updated according to the determined PWM (for the enable pins) and digital output (for the control pins).

5.4 Overcoming Problems Integrating Components and Software

The full integration of all components except the motors was a success. All signals were as desired, being fully controller by the remote control. When the motors were plugged in, they created noise that corrupted signals and resulted in erratic behaviour.

Various testing procedures were followed to better understand the problem. These procedures and the results are summarised in Section 10.8 of the Appendix. Due to time constraints, all the weak-points that the noise was getting in the circuit were not found and so the problem remains. However, through the testing procedure did unveil two problems:

1. Reset pin triggering

The reset pin was being triggered and this caused the microcontroller to continually reset. After using a pull up resistor and decoupling capacitor this issue was resolved.

2. False interrupt triggering

It was expected that the rest of the problem was being caused by similar problems to the reset pin, so an analysis of the signals from the microcontroller was performed. This showed that noise on the receiver pin was causing interrupts to occur. To test this, a pin was pulled high for the duration each interrupt handler function call. This is shown below.

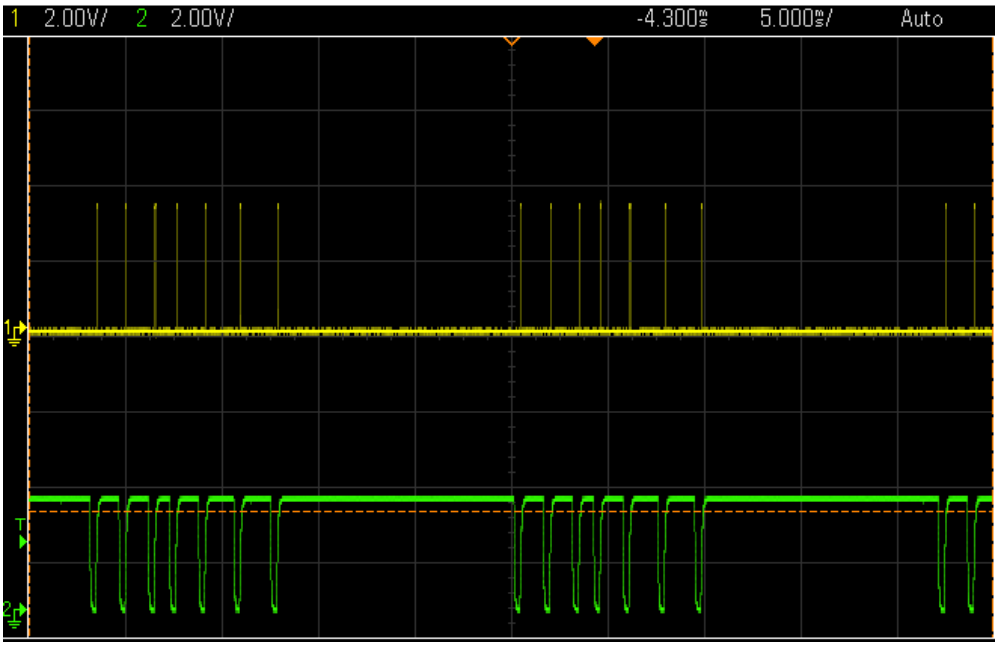


Figure 48: Interrupt pulses in line with PPM message when no motor is connected

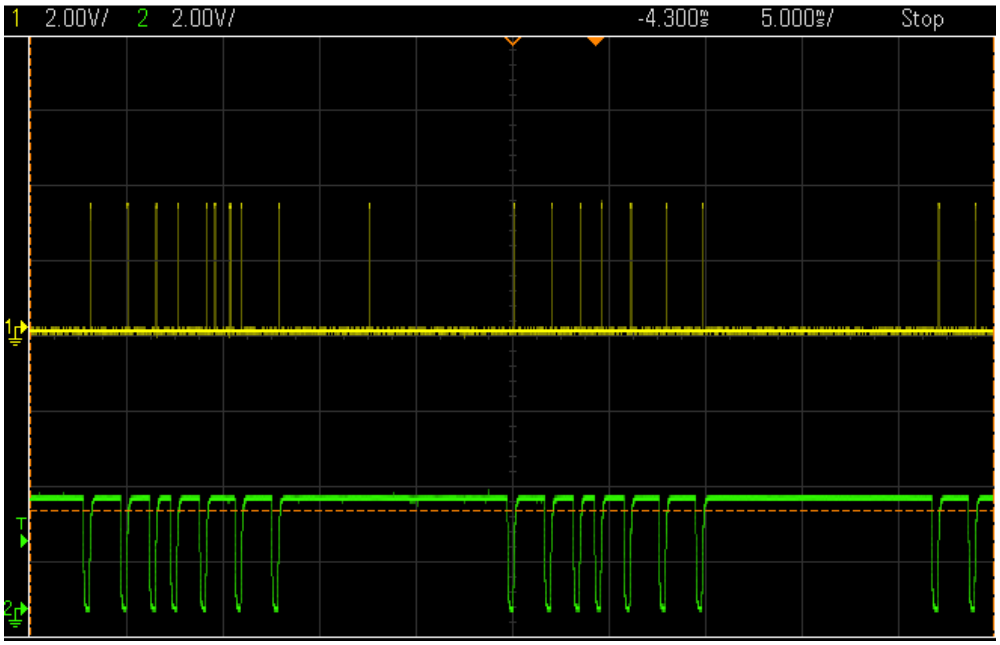


Figure 49: Interrupt pulses show additional interrupts occur when motors are connected

Figure 48 shows a pulse for the duration of each interrupt call when no motor is connected. These pulses are in line with the upwards edge of the PPM messages, as desired. Figure 49 Shows the same thing, but with motors connected. The noise can be seen to cause additional interrupts. This would cause the software to think a channel is larger or smaller than it is, and the interpret function would then create incorrect setpoints for the motor PWM duty cycle. This would constantly be changing as the software resynchs when it notices it is out of synch and the result is the jittery behaviour seen.

With more time, the PPM signal would be debounced adequately in the hopes this would clean up the problem. There is no guarantee that this is the only problem and so in the case that this doesn't fix the problem then other potential avenues to investigate are:

- Check input pins from debugger, or lines around USB area (the micro sends commands when the USB cable is finished debugging the micro);
- Check if microcontroller is somehow faulty by using another one;
- Check signals from light, or the power draw as a result, as the motors randomly jump when the lights are on;

As stated above, due to time constraints it was impossible to completely fix this problem, however the result of the debugging is that the robot can drive fairly well, although it is jittery. Unfortunately, the motors don't receiver full power commands and so the robot cannot reliably climb stairs.

There were various lessons learned from the debugging process.

- The decoupling on power supply for a high power system is crucial. There should be 1000 µF for every amp drawn.
- Leaving pins hanging is never a good idea as they can propogate noise. It is better to ground them, and if necessary to decouple and use a pull down resistor on them
- Although the Veroboard was initially considered a waste of time, the lessons learned with it were invaluable. In future PCBs should be designed iteratively like this to debug problems with a more forgiving setup like a Veroboard.

5.5 TESTING OF DESIGN PHASE 2

The testing procedures of design phase 2 were far more electrical in nature. The power supply, tethered actuator testing, software control testing and untethered driving testing are outlined below.

5.5.1 Power Supply Testing

The power supply as outlined in Section 5.1 was built on a breadboard and tested. The voltage supplied was at 4.85V and 3.32V, which was good enough for the system. This is shown below in Figure 50: Testing results of voltage regulators

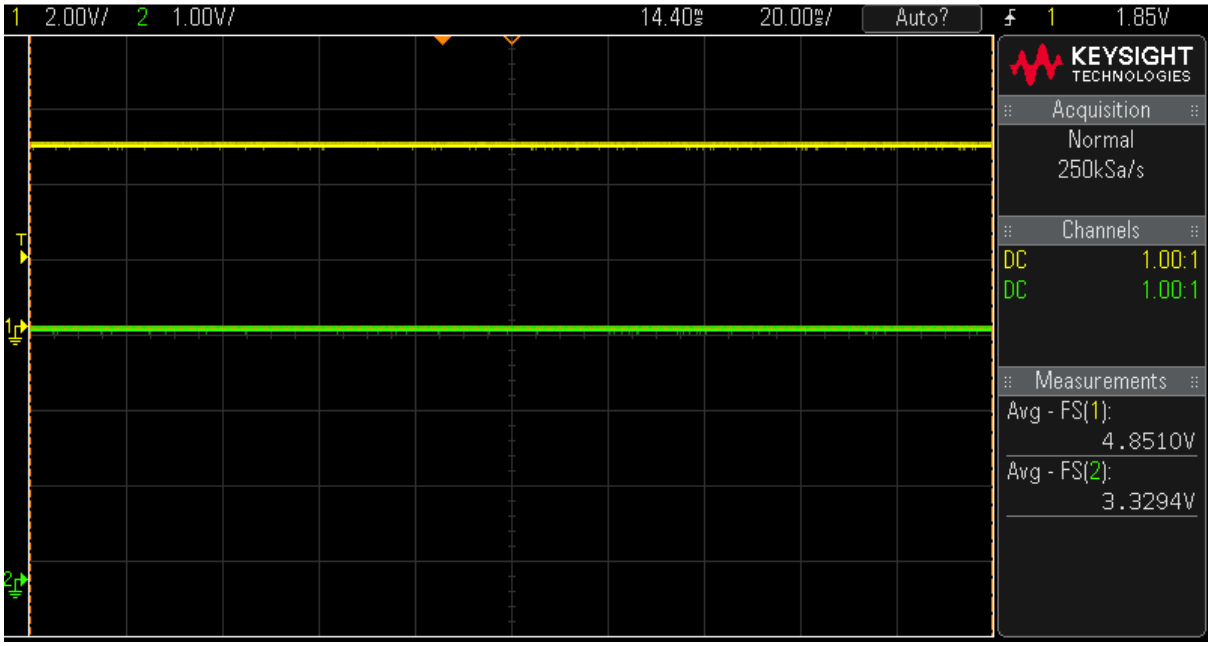


Figure 50: Testing results of voltage regulators

When loaded with the motors running these had little fluctuation, however the 5V variable regulator was changed for a fixed regulator as it was easily available. In future a 3.3V fixed regulator should be used too.

5.5.2 Software Testing

The software was extensively tested as it was written. Only a brief summary of the testing is outlined here. All software can be found in the Di-Wheel Google Drive. An LCD on the UCT development board was used to display testing values. The oscilloscope also proved useful to determine the functioning of the board.

The steps taken to test the software is summarised below:

- Tested pin setup on UCT development board;
  - LEDs used instead of enable and input pins on motors;
  - Button used instead of ppm signal to prove interrupt and channel measurement accuracy;
  - Small LED used for lights.
- Created the TestMotors function to test controller software for motor PWM. The oscilloscope was used to monitor the PWM. The LED lights control software was tested in the same function,

linked to the motor PWM duty cycles. This function tested Forward, Stop, and Reverse commands. Motors and lights responded as expected after development and testing.

- Created the TestInterpret function to test interpreting values for controller. These are normally the mapped receiver values, but a full range of test values was used in this function. LEDs were still used for motor enable and inputs initially. Once the software worked it was tested using the L298n h-bridge module to drive the unloaded motors. Through this testing process the mapping matrix was developed.
- Created the TestChannels function to show exactly what the software was reading for the various ppm signals from the transmitter via the receiver.
- Created the TestMicroTick function to show the exact integer value and PWM signal of the counter to see if it was performing as expected and at a high enough resolution.

The software testing setup shown in Figure 51 was used throughout the software development. To ensure that the software would not break any gears, the Di-Wheels were removed from the robot.

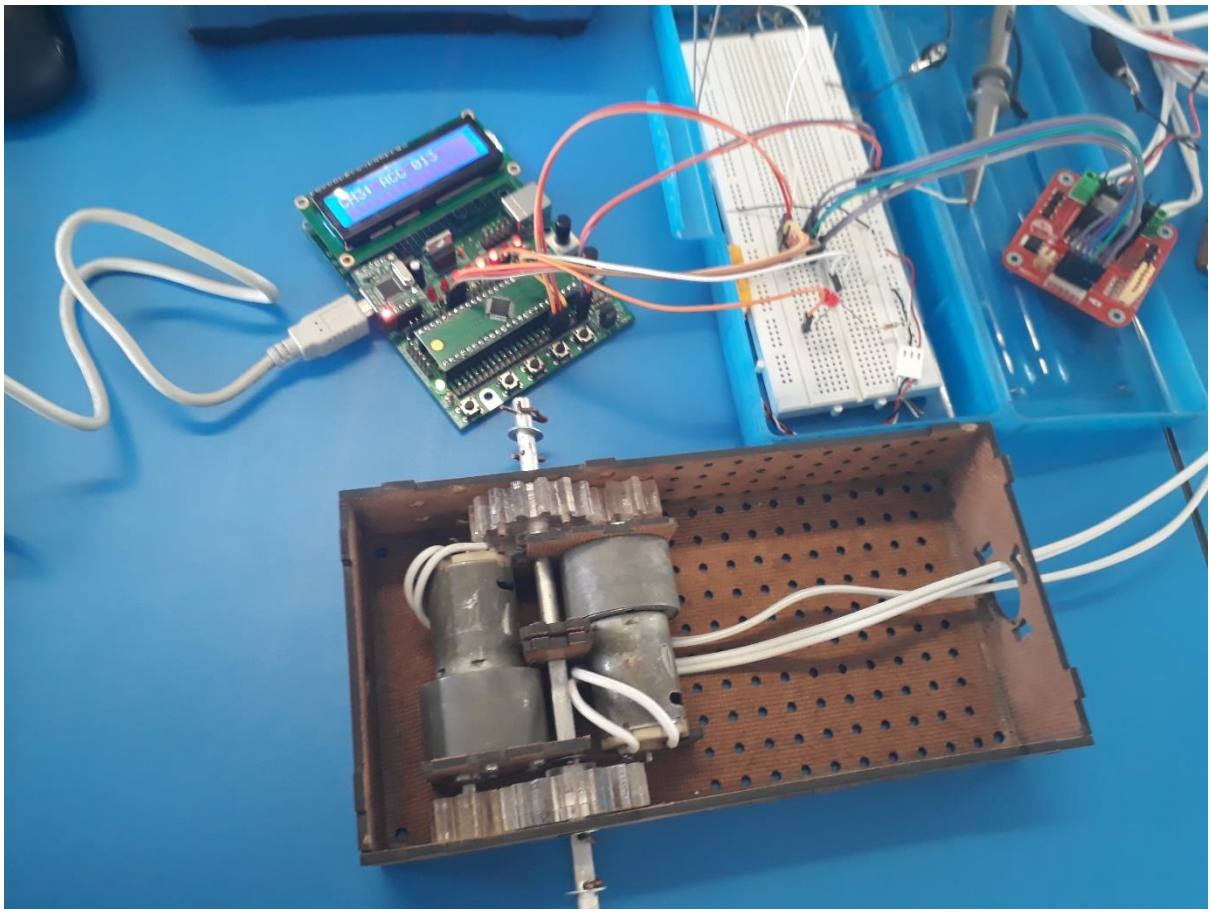


Figure 51: Software testing setup

All the test functions assisted in the development of the software, in debugging issues and ensuring intuitive driving of the robot.

5.5.3 Tethered Actuator Testing

Testing was performed to ensure that the L298N would be able to provide sufficient power for the climbing motion of the robot. The results showed positively that the robot could climb a standard step (standard steps are 200mm, testing height was 220mm). Videos of the testing procedure can be found in the Di-Wheel Google Drive.

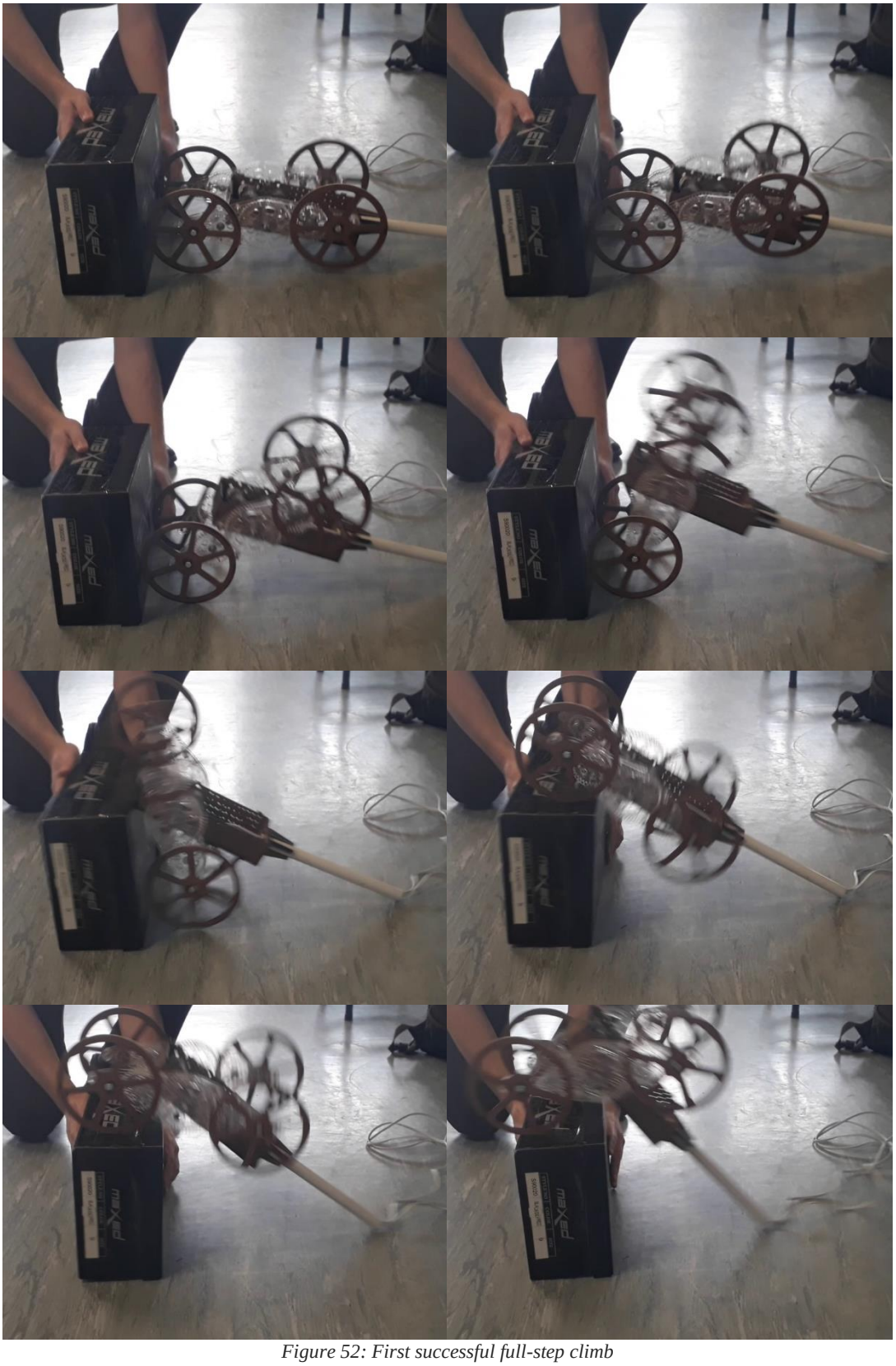


Figure 52: First successful full-step climb

This testing process used the TestMotors() function which just drove the motors using the controller software at 100% duty cycle for three seconds.

It is acknowledged that the testing conditions were ideal, using a smooth floor with no dirt. More extensive testing could not be performed due to the motor noise problem described in Section 5.4.

A result that was unexpected was that the tail kept leveraging off the rear of the robot if the tail smacked into the ground because of going in reverse. When the rear and tail fell off the shorter moment arm caused the robot to constantly climb unless the smallest torque was applied. This poses an interesting question for future development, what if it was possible to change the length of the tail? That might allow you to use more active driving for climbing obstacles (short tail) and more passive driving for flat surfaces (long tail).

5.5.4 Untethered Driving

Once the robot was fully assembled, following the steps shown in the Builder's Guide, and the software was uploaded onto the microcontroller, the robot was tested for untethered driving. Unfortunately, due to time constraints the motor noise problem was not solved, and so untethered driving was not successful. However, the signals were all accurate when the motors were not plugged in.

5.6 SUMMARY OF DESIGN PHASE 2

Design phase 2 ended well with the robot being controlled effectively by the software using the circuits designed.

After completely assembling the robot, the weight was measured for specification SM-8.

Part	Weight
Robot	1600g
• **Body**	1040g
• **Di-Wheel Mechanism**	250g x2
• **Tail**	60g
Transmitter	840g
FPV system	500g
Total	2.94Kg

It was found that the specification was met as the final weight was 2.94Kg for all components of the system, including design phase 3 components. The total cost of all the components also satisfied the budget of R1500 specified in specification SG-2, as the final cost of components purchased was R1182.22. This can be found in the bill of materials in Section 10.4 of the Appendix.

As a result of the work done in this design phase, the following important information was gained:

- The robot can sufficiently run on the L298n H-bridge even though the expected current is much higher than reality. However, for future developments the h-bridge must be redesigned for 10A burst current as this will properly meet the required rating for the motors.
- The wheels need to be stronger and wider. For these tests and prototype they are sufficient, but for future developments the wheels should be stronger and wider.

6 DESIGN PHASE 3: EXTRAS

This section adds two important elements to The Di-Wheel robot. A video/audio feed, as well as a flashlight. A controller using the current sensing feedback from the H-bridge to even out the motors can be found in Section 10.9 of the Appendix. This was not implemented, but a design process went into it for future consideration.

6.1 VIDEO/AUDIO SUBSYSTEM DESIGN

The video/audio subsystem uses First Person View (FPV) technology. A camera and mic are connected wirelessly to FPV goggles through which a user can view the live feed from the camera [34]. This is in comparison to Line of Sight (LOS), where a user controls a remote-control vehicle from their own perspective at a distance.

6.1.1 FPV Components and Implementation

FPV is easy to implement with off-the-shelf components. They are designed to be plug-and-play ready. A full FPV system was already owned at the beginning of this project, these are the FT952 5.8GHz transmitter and the AOMWAY 600 CMOS camera on the robot side, and the RC305 5.8GHz receiver and FatShark FPV goggles on the user side. A comparison of other cameras and transmitters were considered, but it was decided to use what is already owned. This is because this is merely an extra to the project and so its priority was too low to spend sufficient time researching options early enough in the scheduled timeframe to allow components to be purchased. The various other cameras considered are shown in Section 10.6. The components are shown in Figure 53 and Figure 54.

Figure 53: FPV camera and FT952 5.8GHz transmitter

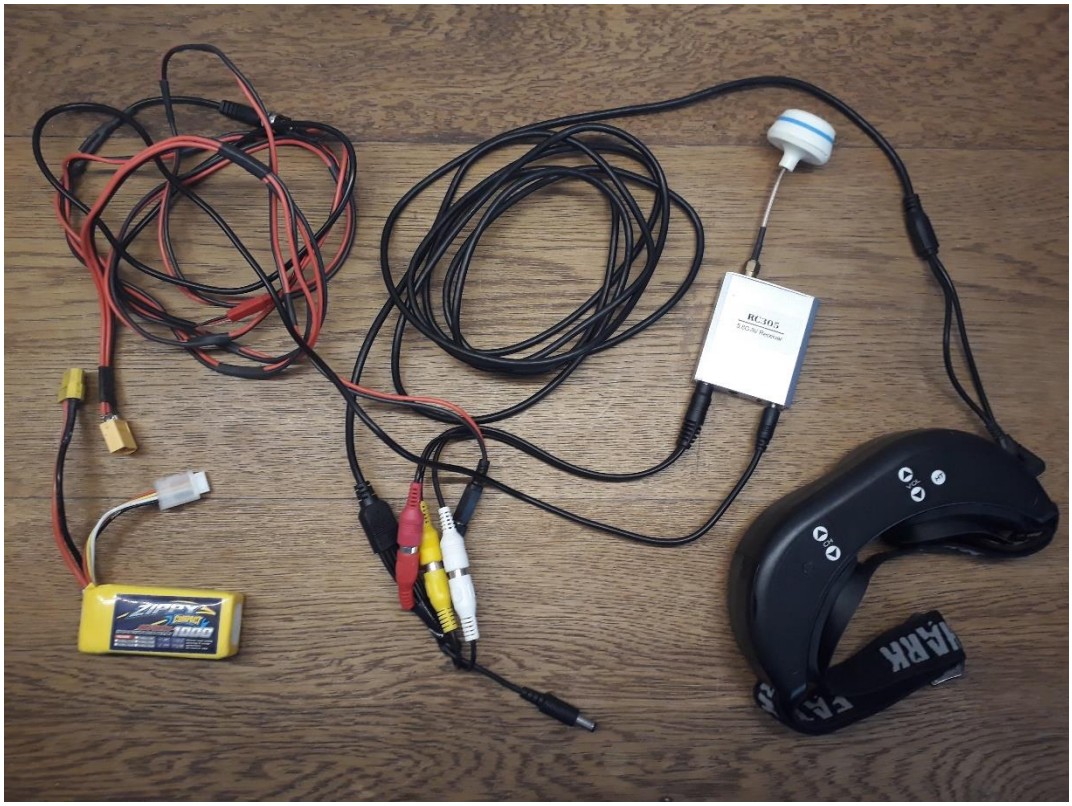


Figure 54: Battery, RC305 FPV 5.8GHz receiver and FatShark FPV goggles

The user side connections are as shown in Figure 54, and are quite intuitive. The connections required for the camera and transmitter to connect on the robot side are shown in Figure 55.

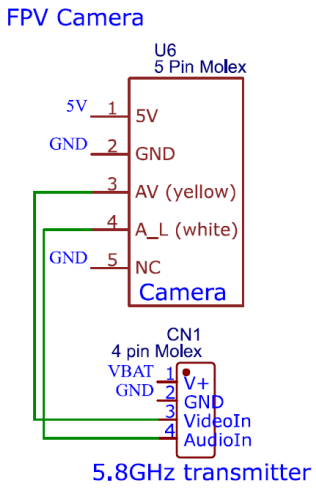


Figure 55: Circuit diagram for FPV camera and 5.8GHz transmitter connection

The circuit board made in Design Phase 2 included the connections necessary for the camera and transmitter, so all that was needed was to plug them in and test if the connections worked.

6.1.2 Video/Audio Testing

The video and audio signals from the camera were tested to ensure that they were correct and had minimal noise. The signals for each are shown in Figure 56.

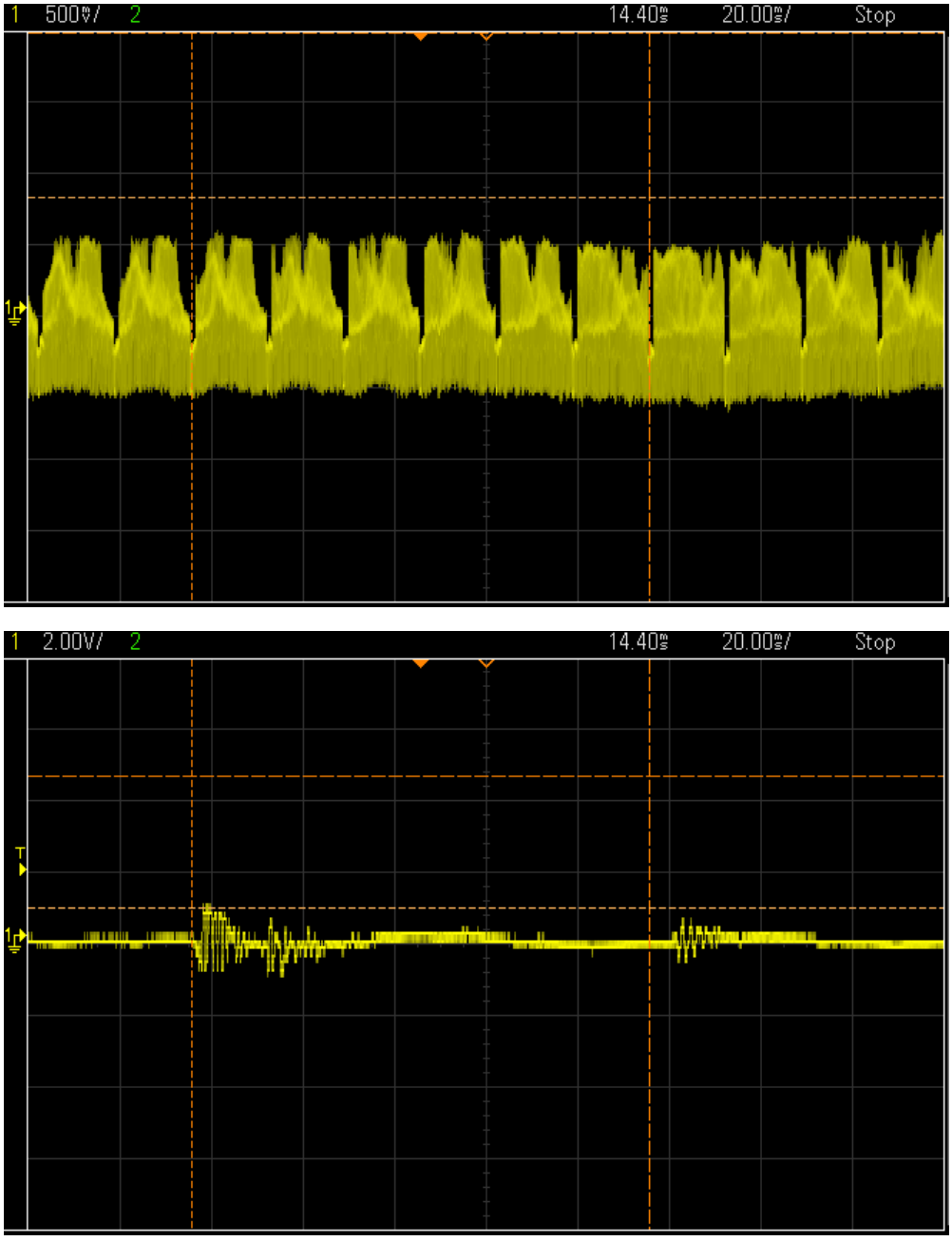


Figure 56: Video (top) and Audio (bottom) signals, responding to a hand wave in front of the camera and a tap on the mic

As can be seen on image 1, the pattern changes as a hand is waved in front of the camera. This was the best way of testing that the signal from the camera was operational before plugging it in. Image 2 shows spikes where a tap on the mic occurs. Both of the signals were as expected, and so the receiver was set up. Figure 57 shows the video feed in the FPV goggles.

Figure 57: Video feed in FPV goggles

The video feed came through clearly as seen above. The audio was also clear, although this cannot be demonstrated.

6.1.3 Summary of Video/Audio Implementation

The video feed was perfect for the conditions expected in USAR environments, without being too high quality. The audio was also a good quality. Range and obstacles tests were not conducted, due to time constraints.

6.2 FLASHLIGHT SUBSYSTEM DESIGN

A flashlight is needed for the system because USAR environments can be dark and require illumination for the camera. It can also be used as a crude means of communicating with survivors, by flashing the light at them to let them know they have been seen. The design for the flashlight subsystem requires an LED driver circuit, and software to link the transmitter commands to the driver. The following sections outline these features.

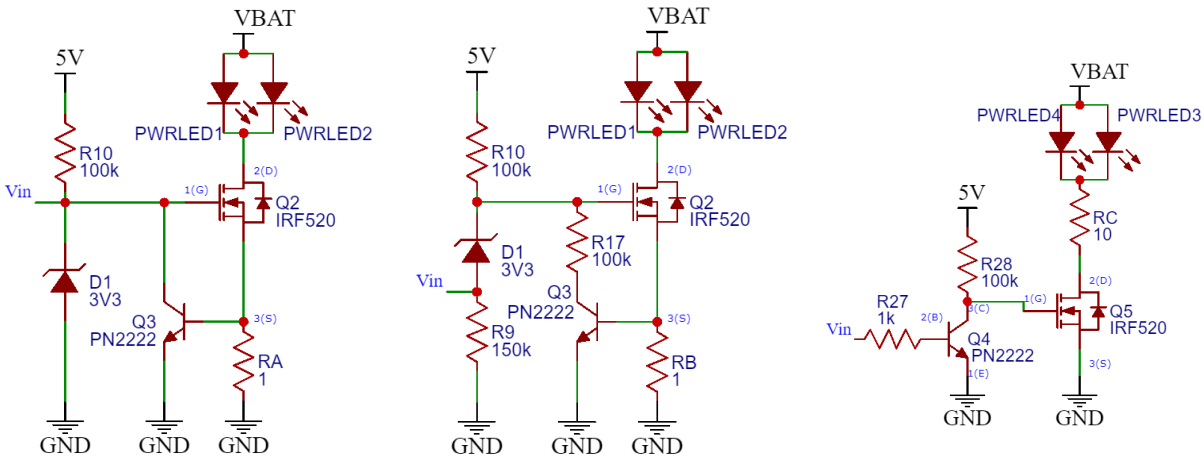
6.2.1 LED Driver Circuit

The LEDs selected were 5W ACM star power white LEDs from Communica. An image of this is shown below.

Figure 58: White Power LED and Bracket

Attaching the LEDs to a bracket as shown above, and then placed at the front of the robot, light could be produced to help a user to see in a dark environment. The LED forward voltage is 10.8V with 0.4A forward current [19].

A high-power LED driver circuit is needed because of the expected current draw. This circuit was designed to allow the LEDs to be switched on or off completely, or with varied brightness. There are various options for the LED driver circuit as shown in Figure 59.



(A) [35] *(B)* *(C)*

Figure 59: Power LEDs driver circuit options

The IRF520 MOSFET has the following parameters:

- $V_{GS(th)} = 4V$ Maximum
- $R_{DS(on)} = 0.27\Omega$ at $V_{GS} = 10V$ and $I_D = 5.5A$

The resistors in series with the LEDs were calculated as follows:

$$R_A = R_B = \frac{V_{BE}}{I_{Diode}} = \frac{0.6V}{0.8A} = 0.75\Omega$$

$$R_C = \frac{V_{BAT} - V_{GS(on)}}{I_{diode}} = \frac{16.8V - 10.8V}{0.8A} = 7.5\Omega$$

Each resistor was rounded up, R_A and R_B to 1Ω, and R_C to 10Ω. In order to make these values, multiple higher value resistors in parallel were used to distribute the current. All other resistor values were selected as large values to reduce current flow.

Option A was taken from Instructables [35], never worked as the LEDs never fully turned on. Option B was a modification of option A. The LEDs turned fully on and off with this circuit. This was because the gate was switching between 3.3V and 6.6V, and with the $V_{GS(on)}$ for the IRF520 MOSFET at 4V it was definitely turning on [36]. To ensure it turned on fully, the resistor R21 was added to introduce a voltage divider with the NPN. This helped push the MOSFET gate just that little higher, as well as limiting current flow in the NPN and conserving power. This option felt messy, and so option c was designed.

Option C was the best design and the one ultimately implemented. Using a simple NPN as a NOT function to switch the voltage at the MOSFET gate high or low, the LEDs could be turned fully off or on.

6.2.2 Software for Flashlight

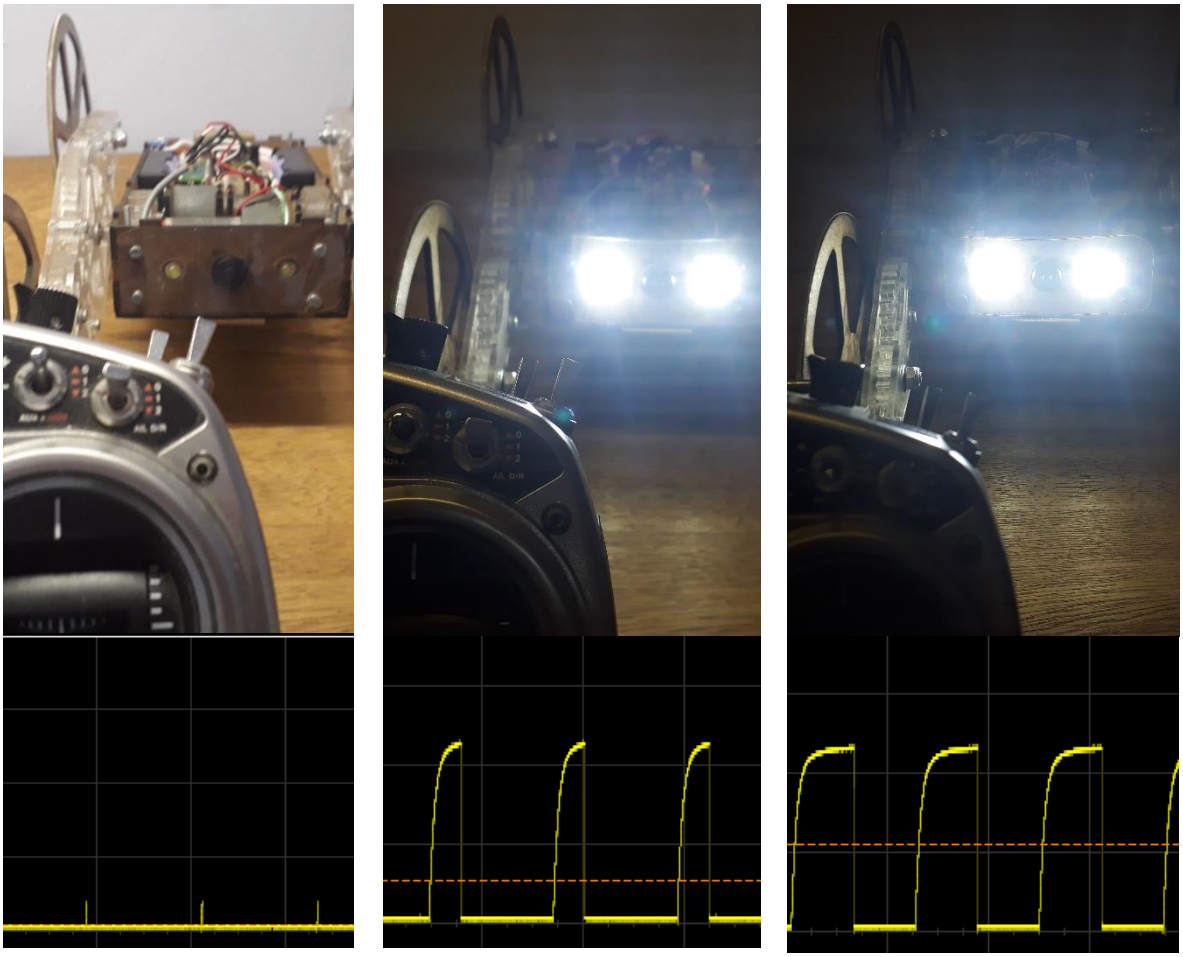
The microcontroller was connected to the input of the LED driver circuit using GPIOA pin 11 with PWM. One of the channels for a toggle switch on the transmitter already set up in Design Phase 2 was used to read in commands from the user. The code for reading the light channel, interpreting it and setting the duty cycle value can be found in Section 10.7 of the Appendix.

The input from the receiver was mapped like before, except this time it was only on the scale 0-100. This was then used to set the LED duty cycle. There were three regions defined with PPMMAXLIGHT, PPMMIDLIGHT and PPMMINLIGHT, and the brightness of the LEDs were defined according to already defined brightness levels MAXLIGHT, MIDLIGHT and MINLIGHT.

Using this code, the LEDs were reliably controlled using the transmitter.

6.2.3 Flashlight Testing

The LED driver circuit in option C worked very well, turning the LEDs on reliably whenever the microcontroller desired it. The testing results can be seen in Figure 60.



(A) Minimum light (B) Medium light (25%) (C) Maximum light (50%)

Figure 60: Testing results of LED driver circuit and the PWM duty cycles for each mode

The PWM signal from the microcontroller, and the resultant light output was as expected. This was consistent under all conditions.

6.2.4 Summary of Flashlight Subsystem Development

The flashlight implemented was a high-power LED, driven by the microcontroller through a simple LED driver circuit. Using the same communication channel implemented in design phase 2, the software was able to set the PWM duty cycle of the light output pin. There were only three levels of brightness used, 0%, 50% and 100%. The results showed that the implementation of the flashlight was a success. No further development was needed for this.

6.3 Integrated Testing of Design Phase 3

The testing of design phase 3 was not as intensive as the other two design phases. This is because they are simple features that either work or they don't. The testing of each extra feature individually was reported in their respective sections, 6.1.2 and 6.2.3.

6.4 SUMMARY OF DESIGN PHASE 3

The implementation of the video/audio and flashlight subsystems was a success, showing reliability in their use after they were tested. The video quality from the camera feed is perfect for USAR conditions, and the audio is loud and clear enough to hear the surroundings. The flashlight is very bright, with variable brightness. It only turns on when the microcontroller commands it to turn on, and reliably turns on when the signal is sent.

7 CONCLUSIONS

Overall the project was a success, with the highest priority specifications met. The order of priority for the scope of this project is shown below, as well as a discussion of any that were not satisfied.

1. Prove that the Di-Wheel mechanism is able to provide for the USAR requirements of manoeuvrability. – Satisfied
2. Prove that the motors can be controlled by a microcontroller to climb. – Satisfied
3. Prove that the microcontroller can receive and interpret wireless commands correctly. – Satisfied
4. Prove that a video/audio feed is possible for this robot – Satisfied
5. Prove that a light can be used to illuminate the surrounding area – Satisfied
6. Prove that the wireless commands can be used to reliably control the motors and lights in a full integration of all the components together. – Not Satisfied

The results of this project were able to prove that all the individual components could be developed, and that the Bi-Wheel mechanism can be used for climbing obstacles. The integration of all the components was also a success in all regards except for the signal noise when the motors were connected.

Overall, the project is considered a success because of how the various components were able to function as desired in all conditions except when the motors are connected. With a little more time, this could have been solved too.

7.1 REQUIREMENT VERIFICATION AND ANALYSIS

The user specifications specified in Section 3.2 will now be analysed to show what specifications were not met and reasons will be given as to why. Each specification has a status next to it. For those that are partially or not fulfilled, a reason is given.

General Design			
SG-1	Full implementation	All features must work together at the same time	**Partially Satisfied**
	The problem encountered in Section 5.4 of Design Phase 2 was not solved. This meant that the response of the system when the motors were connected was not reliable. In all other respects this was fulfilled.		
SG-2	Cost<R1500	The total cost must be below R1500.	Satisfied
SG-3	All components fit together in body	The components must all completely fit together inside the body of the robot.	Satisfied
SG-4	Neat assembly	The package and electronics must look neat and presentable.	Satisfied
SG-5	USAR appearance	The appearance of the robot must be a USAR robot, not a toy.	Satisfied
SG-6	Open loop control	The user must be the control system.	Satisfied

SG-7	Reliable in response to commands	The robot must always respond in a predictable way to commands.	**Partially Satisfied**
	The same reason as SG-1. The system is reliable in all respects except when noise from the motors is introduced.		
		Mechanical Design	
SM-1	Minimum Di-Wheel size	220mm long (this is the length of the carrier).	Satisfied
SM-2	Wheel diameter	Maximum diameter is the length of the carrier (220mm). Minimum diameter is the diameter of the largest gear.	Satisfied
SM-3	Wheels must have traction	The wheels need to have sufficient traction to grip when encountering an obstacle.	Satisfied
SM-4	Climb steps of varying heights	The robot must be able to climb over varying step heights.	Satisfied
SM-5	Body encloses electronics	The body must completely encapsulate all the required electronics with no exposed wires.	Satisfied
SM-6	Maximum Body height	It must only be high enough to fit the motors and or batteries. This reduces chances of beaching underneath.	Satisfied
SM-7	Maximum Body width	It must only be wide enough for motors to fit, this reduces the chance of beaching in front.	Satisfied
SM-8	Weight<5Kg	The weight must be less than 5Kg for easy carrying, and climbing.	Satisfied
SM-9	Body and tail length	The body and tail length specify the moment arm of the body, this must be longer than half the length of the Di-Wheel mechanism. This will ensure that the mechanism always has something off which to push.	Satisfied
SM-10	Easy assembly	The parts must be easy to assemble for manufacture of robot.	Satisfied
SM-11	Partial disassembly	The robot must be partially disassembled for transporting, with easy and quick reassembly.	Satisfied
SM-12	Shoebox packing space	The partially disassembled robot must fit in a shoebox for transportation.	Satisfied
SM-13	No sharp edges	The construction must be safe for users.	Satisfied
SM-14	Covered Gears	The gears must be covered and not exposed to potentially hurt a user holding the robot.	Satisfied
SM-15	Righting ability	If the robot falls on its side it must be able to right itself.	**Not Satisfied**
	The robot had a tendency to tip over on its side if the one motor became dominant. It was always unable to right itself in this condition. This can be solved with a bracket on the Di-Wheel to prevent it from falling flat on its side.		
		Materials, Parts and Components Procurement	
SP-1	Easily accessed materials	The materials must be easily accessible. This means they must be available at the university or locally.	Satisfied
SP-2	Drop resistant materials	The materials must be strong enough to resist falls from steps. The expected drop is 200 mm.	Satisfied
SP-3	Easy manufacture methods	The methods of manufacture must be easily reproducible by any reader of this report. The simplest methods are through laser cutting of parts, and 3D printing.	Satisfied
SP-4	Least expensive materials and components	The materials and components must be inexpensive to reduce the overall cost.	Satisfied

SP-5	Builder's Guide	A builder's guide must be generated for any user to easily manufacture their own robot.	Satisfied
		Electrical Design	
SE-1	Long battery life	In idle: 4 hours. constant driving: 2 hour. constant stair climbing: 1 hour. Must be able to easily change the battery.	**Partially Satisfied**
	Because of the motor noise problem in Section 5.4 this could not be tested. The batteries are estimated to last that long, but it was never verified.		
SE-2	Wireless controlled	Using 2.4GHz for remote control of robot as this is standard for remote controls.	Satisfied
SE-3	Sufficient motor torque	The motor torque must be sufficient to allow climbing.	**Partially Satisfied**
	This was verified in tethered tests, but only climbed a few times untethered due to the motor noise problem in Section 5.4.		
SE-4	Fastened components	The components must be held in position for when the robot is upside down.	Satisfied
SE-5	Visual/ Audio feed	Use a camera and mic setup to relay visual and audio feed of the robot. This must run on 5.8GHz as this is standard. This is an extra feature and is not high priority.	Satisfied
SE-6	Flashlight	Use high power LEDs to illuminate the area for dark conditions. This is an extra feature and is not high priority.	Satisfied
SE-7	Sensory feedback	Temperature, gas, humidity feedback can be implemented. This is an extra feature and is not high priority.	**Not Satisfied**
	There was not sufficient time to include these features. This was considered lowest priority.		
SE-8	Communication method with survivors	Using either lights, or audio output with prerecorded messages, the robot must be able to give hope to a trapped survivor.	**Partially Satisfied**
	There is a method of communication, by using the flashlight, however an audio output would be preferred.		
SE-9	No loose wires	For safety there must be no loose wires hanging outside the robot.	Satisfied
SE-10	Power toggle switch	There must be a power toggle switch to immediately depower the robot in the event of a problem.	Satisfied
SE-11	Batteries must not present danger	Li-Po have a potential to explode, so Li-Ion should be selected instead.	Satisfied
SE-12	Onboard Debugger	There must be an onboard debugger to allow easy upload of code.	Satisfied
		Software Development	
SS-1	Test procedures	There must be test procedures for debugging purposes.	Satisfied
SS-2	Reliable outputs with given inputs	The output of the microcontroller must be reliable depending on the input received.	Satisfied
SS-3	Read in receiver channels	Must be able to read in receiver commands, and map them to a scale of percentage.	Satisfied
SS-4	Easily switch channels	Global defined variables must define the channels used as input.	Satisfied
SS-5	Intuitive driving	The method of interpreting commands and controlling the motors must be intuitive to a user	Satisfied

With this analysis of the Technical Specifications it can be seen that in general the project was a resounding success, and with further development can show a promising product for commercialisation in USAR conditions. The highest priority requirements and project outcomes were met.

8 RECOMMENDATIONS

On the basis of the above conclusions, the following recommendations are made. In order to improve the design of the Bi-Wheel robot, several suggestions were developed throughout the project. These are summarised below.

8.1 CONTROL LOOP RECOMMENDATIONS

An augmented open loop control system should be employed to equalise motors and help the user in getting a more responsive and intuitive feel. An initial design of this can be found in Section 10.9 of the Appendix. This will also prevent one motor becoming dominant and turning the robot on its side. This can be employed with feedback from and accelerometer, current sensing or both.

8.2 CONSTRUCTION RECOMMENDATIONS

The mechanical and electrical construction can be much improved. The recommendations for this are:

- Manufactured the Di-Wheel in aluminium, with steel shafts for strength. This is light and will provide additional strength. It can also be thinner material as a result.
- Add a set of fins on the side of the Di-Wheel that will prevent it from beaching on its side.
- Make metal gears to get better strength and stop it from wobbling.
- Use bearings to support shafts.
- Redesign the body to be longer, to allow more space for electronics. The tail can be shorter as a result;
    - The H-bridge and batteries must be closer to the motors, and the microcontroller can be at the rear, with all other supporting circuitry.
- The debugger component should be placed on the edge of the board, with a hole out the back so that it is easier to stick the USB in to program the robot without having to remove the lid
- Once a finalised electronics design works, it should be designed on a PCB
- Different motors can be considered, but it may not be necessary if the gear ratios are optimised
- Use higher current rated batteries to ensure consistent results.
- A higher current H-Bridge should be built for guaranteed climbing.
- There should be much more decoupling in future designs. Consider decoupling for different frequencies, and at least 1000 μF per Amp drawn from the motors.
- Redesign for stronger, wider wheels. Possibly buy actual RC tyres.

With these considerations, the mechanical and electronic designs can be much improved.

8.3 FUTURE OPPORTUNITIES

This project was limited in time and scope, but there are many variations of the design that could be tested. The mathematical model can be further developed and tested by add different types of gear ratio Di-Wheel mechanisms to compare the performance of flipping over motion and ramping up motion (refer to Figure 14) with the expected result of the mathematical model. Using these mechanisms, the attempt should be to try and break the model to see its weak points. If these recommendations are followed, there is great promise for the Di-Wheel robot.

9 REFERENCES

[1] N. Tesla, “Method of and apparatus for controlling mechanism of moving vessels”. United States of America Patent 613809, 11 1898.

[2] FEMA, “Urban Search and Rescue,” FEMA, 1 8 2018. [Online]. Available: https://www.fema.gov/urban-search-rescue. [Accessed 30 7 2019].

[3] NIST, “Emergency Response Robots, National Institute of Standards and Technology,” NIST, 7 2018. [Online]. Available: https://www.nist.gov/programs-projects/emergency-response-robots. [Accessed 30 8 2019].

[4] E. Messina et al, “Statement of Requirements for Urban Search and Rescue Robot Performance Standards, National Institute of Standards and Technology and Department of Homeland Secuirty,” 5 2005. [Online]. Available: https://www.nist.gov/sites/default/files/documents/el/isd/ks/Prelim_Requirements_Report.pdf. [Accessed 30 8 2019].

[5] V. Fenton, “The use of dogs in search, rescue and recovery,” *Journal of Wilderness Medicine,* vol. 3, pp. 292-300, 1992.

[6] N. Godlewski, “Natural Disasters: Hurricanes, Earthquakes Likely Unrelated,” 21 9 17. [Online]. Available: https://www.ibtimes.com/natural-disasters-hurricanes-earthquakes-likely-unrelated-2592684. [Accessed 12 10 2019].

[7] R. IEEE, “RHEX,” 2001. [Online]. Available: https://robots.ieee.org/robots/rhex/. [Accessed 12 10 2019].

[8] R. IEEE, “ThrowBot,” 2012. [Online]. Available: https://robots.ieee.org/robots/throwbot/. [Accessed 12 10 2019].

[9] I-Robot, “iRobot 110 FirstLook Robot,” 2019. [Online]. Available: https://www.army-technology.com/projects/irobot-110-firstlook-robot/. [Accessed 12 10 2019].

[10] ARA, “Pointman,” 2011. [Online]. Available: http://www.wilkoaero.com/uploaded/product/870/catalog_5a265a4ff38988ac67387ce1a19a1a5e0.pdf. [Accessed 12 10 2019].

[11] M. Wilson, “Development of a Low-Cost, Mid-Sized, Tele-Operated, Wheeled Robot for Rescue Reconnaissance,” University of Cape Town, Cape Town, 2013.

[12] L. M. Smith, R. D. Quinn, K. A. Johnson and W. R. Tuck, “The Tri-Wheel: A Novel Wheel-Leg Mobility Concept,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems,* Hamburg, Germany, 2015.

[13] R. W. Forsyth and J. P. Forsyth, “Design and Development of the TerraStar Marginal-Terrain Amphibian,” Society of Automotive Engineers, San Francisco, 1968.

[14] J. A. Haskel, “THESEUS: A COST EFFECTIVE WHEELED RESCUE ROBOT SOLUTION,” University of Cape Town, Cape Town, 2017.

[15] M. Buchanan, “Ascender: A cost effective, mass producible, stair-climbing solution for urban search and rescue,” University of Cape Town, Cape Town, 2018.

[16] SANS10400, “Stairways,” SANS10400 Building Regulations, 4 2011. [Online]. Available: https://www.sans10400.co.za/stairways/. [Accessed 22 7 2019].

[17] ST, “STM625051C6 Datasheet,” 1 2017. [Online]. Available: https://www.st.com/resource/en/datasheet/stm32f051t8.pdf. [Accessed 30 9 2019].

[18] ST, “ST-Link Debugger processor Datasheet,” 8 2015. [Online]. Available: https://www.st.com/resource/en/datasheet/stm32f103c8.pdf. [Accessed 30 9 2019].

[19] Communica, “ACM STAR POWER LED WHITE 5W 11V,” Communica, 2019. [Online]. Available: https://www.communica.co.za/products/acm-star-power-led-white-5w-11v. [Accessed 30 9 2019].

[20] HobbyKing, “OrangeRx R617XL cPPM DSM2/DSMX Compatible 6ch Receiver,” HobbyKing, 2019. [Online]. Available: https://hobbyking.com/en_us/r617xl-dsm2-x-6ch-receiver-with-long-antenna-and-cppm.html. [Accessed 30 9 2019].

[21] AOMWAY, “AOMWAY Mini CMOS Camera,” AOMWAY, 2019. [Online]. Available: https://cdn-global-hk.hobbyking.com/media/file/206291797X1709446X53.pdf. [Accessed 30 9 2019].

[22] FlyingTech, “Boscam FT952 5.8Ghz 200mW 32CH Mini FPV Transmitter with GoPro 3 AV Lead,” FlyingTech, 2019. [Online]. Available: https://www.flyingtech.co.uk/fpv-camera-gimbals/ft952-58ghz-200mw-32ch-mini-fpv-video-transmitter. [Accessed 30 9 2019].

[23] ST, “L298 Dual Full Bridge Driver,” 1 2000. [Online]. Available: https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf. [Accessed 30 9 2019].

[24] Geeetech, “L298N Motor Driver Board,” Geeetech Wiki, 8 1 2013. [Online]. Available: https://www.geeetech.com/wiki/index.php/L298N_Motor_Driver_Board. [Accessed 1 10 2019].

[25] O. Liang, “RC TX RX PROTOCOLS EXPLAINED: PWM, PPM, SBUS, DSM2, DSMX, SUMD,” oscarliang.com, 2 2018. [Online]. Available: https://oscarliang.com/pwm-ppm-sbus-dsm2-dsmx-sumd-difference/. [Accessed 28 9 2019].

[26] STMicroelectronics, “How to build a “Blink LED” project from STM32CubeMX for ST/Atollic TrueSTUDIO® for STM32™,” YouTube, 27 3 2018. [Online]. Available: https://www.youtube.com/watch?v=6RqUkFIeN6w. [Accessed 28 9 2019].

[27] “Using STM32 HAL Timer and Adjusting the Duty Cycle of a PWM signal,” Stack Overflow, 21 4 2017. [Online]. Available: https://stackoverflow.com/questions/43483762/using-stm32-hal-timer-and-adjusting-the-duty-cycle-of-a-pwm-signal. [Accessed 28 9 2019].

[28] L. T. Phuc, “STM32F0 Tutorial 4: Timer with CubeMX - PWM Generation,” YouTube, 27 6 2015. [Online]. Available: https://www.youtube.com/watch?v=Ql6klWiROik. [Accessed 28 9 2019].

[29] S. Bukhardt, “GPIO Operations on STM32 Microcontrollers using HAL,” 20 5 2016. [Online]. Available: https://simonmartin.ch/resources/stm32/dl/STM32%20Tutorial%2001%20-

%20GPIO%20Operations%20using%20HAL%20(and%20FreeRTOS).pdf. [Accessed 28 9 2019].

[30] L. T. Phuc, “STM32F0 Tutorial 3: External Interrupt (EXTI) with CubeMX, Keil and Source Insight,” 10 3 2015. [Online]. Available: https://www.youtube.com/watch?v=HqeNed6Y4qQ. [Accessed 28 9 2019].

[31] O. Liang, “PWM AND PPM DIFFERENCE AND CONVERSION,” oscarliang.com, 2018. [Online]. Available: https://oscarliang.com/pwm-ppm-difference-conversion/. [Accessed 28 8 2019].

[32] K. Ahmad, “Reading PPM Receiver Signal with Arduino using Interrupts,” Kamran Ahmad - YouTube, 7 10 2015. [Online]. Available: https://www.youtube.com/watch?v=63JmO4Mc8NM. [Accessed 28 9 2019].

[33] J. Gowans, “Timer Interrupt,” Git Hub, 18 5 2014. [Online]. Available: https://github.com/jgowans/stm32f0_devel/blob/master/jgowans-sample-projects/timer_interrupt/main.c. [Accessed 28 9 2019].

[34] C. Montgomery, “Multi-Rotors, First-Person View, and the Hardware You Need,” Toms Hardware, 3 6 2014. [Online]. Available: https://www.tomshardware.com/reviews/multi-rotor-quadcopter-fpv,3828.html. [Accessed 2 10 2019].

[35] Dan, “High Power LED Driver Circuits,” Instructable, 2016. [Online]. Available: https://www.instructables.com/id/Circuits-for-using-High-Power-LED-s/. [Accessed 2 10 2019].

[36] Vishay, “IRF520 Power MOSFET,” 21 3 2011. [Online]. Available: https://www.vishay.com/docs/91017/91017.pdf. [Accessed 2 10 2019].

[37] “Spur Gears,” NPTEL, [Online]. Available: https://nptel.ac.in/courses/116102012/spur%20gears/design%20aspects%20of%20spur%20gear.html. [Accessed 31 7 2019].

[38] “Unit IV Epicyclic gear boxes,” [Online]. Available: https://www.svce.ac.in/departments/auto/Lesson%20plan/III%20YEAR%20CLASS%20NOTES/AT2301/UNIT%20IV.pdf. [Accessed 31 7 2019].

[39] “What Is the General Function of an Idler Gear?,” [Online]. Available: https://www.reference.com/vehicles/general-function-idler-gear-3353870a32135094. [Accessed 31 7 2019].

[40] “Does HAL_GetTick() return ticks or milliseconds? (and how to measure in microseconds),” Stack Overflow, 12 3 20173. [Online]. Available: https://stackoverflow.com/questions/42747128/does-hal-gettick-return-ticks-or-milliseconds-and-how-to-measure-in-microsec/42749284. [Accessed 28 9 2019].

[41] Antonius, “How to use HAL_GetTick ?,” community.st.com, 6 6 2019. [Online]. Available: https://community.st.com/s/question/0D50X0000Ave1cJ/how-to-use-halgettick-. [Accessed 28 9 2019].

10 APPENDIX

This appendix contains information relevant to the project, but not included in the main text. There is additional information contained in the Di-Wheel Google Drive for this project.

10.1 PROPOSED TIMELINE

The following was the initial proposed timeline for this project.

Legend: RED – UCT deadlines BLACK – personal deadlines

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
14 July	15	16	17	18	19	20
21	22	23	24	25	26	27
					Ethics form – complete	
28	29	30	31	1 August	2	3
		Lit review	Lit review original due date			
4	5	6	7	8	9	10
			Proposed concepts - removed			
11	12	13	14	15	16	17
18	19	20	21	22	23	24
		Mechanical design calculations/ simulations				
25	26	27	28	29	30	31
		Mechanical Design				
1 Sep	2	3	4	5	6	7
		Electronic design		Design phase 1		
8	9	10	11	12	13	14
			Power Supply			
15	16	17	18	19	20	21
		Controller		Design Phase 2		
22	23	24	25	26	27	28
	ELO tracking		Video/audio			
29	30	1 Oct	2	3	4	5
	Flashlight			Testing	Pre-submission	Report writing
6	7	8	9	10	11	12
13	14	15	16	17	18	19
	Final submission			Poster		
20	21	22	23			

Open day: oral and poster presentation

10.2 BASIC GEARING

The main text assumes the reader has full understanding of a the basics of gearing, this is explained here for a reader who is less experienced. The actual specifications of a gear are related to the size of the gear, the teeth and various other measurements from the gear. A summarised version of a gear's specifications is shown in Table 13

Specification	Symbol	Definition
Diametral pitch	P	$P = \frac{N}{d}$
Number of teeth	N	
Pitch circle diameter (PCD)	d	
Module	M	$m = \frac{d}{N}$
Contact angle	ϕ	Typically 20°
Radius of base circle	r_b	$r_b = \frac{d}{2}\cos\phi$
Circular pitch	p	$p = \pi m$
Tooth width at PCD	t	$t = \frac{\pi m}{2}$

Table 13: Gear specifications

A physical representation of these specifications is shown in Figure 61.

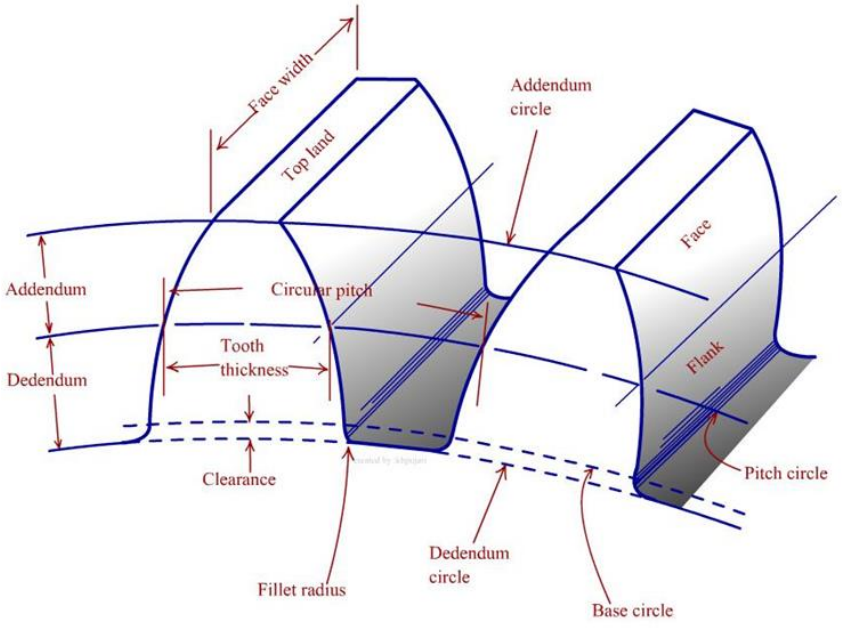


Figure 61: Nomenclature of spur gears [37]

In the design of gears, the only thing you need to specify is the number of teeth and the pitch circle diameter. It is very important that all the gears that mesh have the same module, so the exact value of the diameters and number of teeth is a function of all gears required.

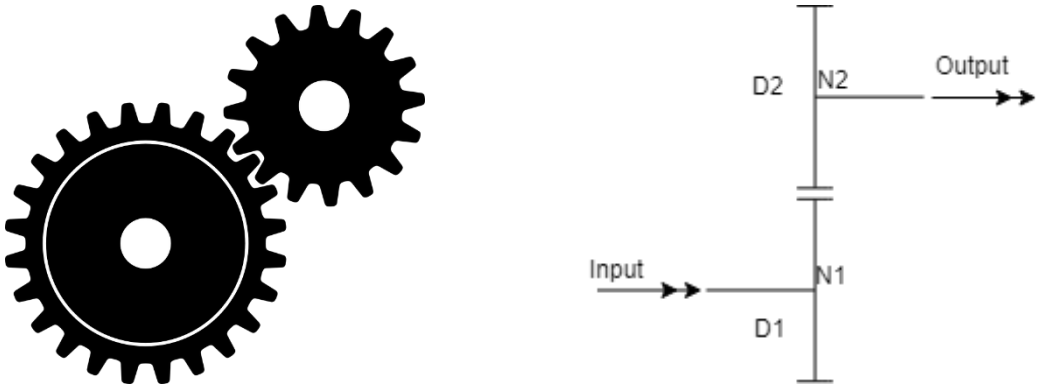
Gear trains are when two or more gears mesh together, which transmits power [38]. There are various types of gear trains [38]:

- Simple: only one gear on each shaft
- Compound: more than one gear on a shaft

- Reverted: axes of first and last gear are coaxial
- Epicyclic: axes of shafts over which gears are mounted, may move relative to a fixed axis

Idler gears are inserted between two other gears in a gear train. Its purpose is to change the direction of output and/or maintaining the spacing between shafts [39].

Gears caan be shown using two methods, namely the standard toothed gear, and the side view representation. These representations are shown in Figure 62.



A) *Standard view* *B) Side view representation*

Figure 62: Gear representations

In Figure 62B, D1 and D2 indicate the gears, where D1 is the input gear and D2 is the output gear. N1 and N2 are placeholders for the number of teeth on each respective gear.

10.3 Mathematical Calculations and Diagrams

10.3.1 Gear ratio spreadsheet

The following spreadsheet shows the rotational velocity of the carrier vs the forward velocity of the robot, all calculated using gear ratios. For access to the excel spreadsheet that this was calculated in, go to the Di-Wheel Google Drive:

		sun (D1)		idler (D2)		planet (D3)		driving mode rotations							Climbing mode rotations					
module	forward speed (m/s)	PCD1	N1	PCD2	N2	PCD3	N3	sun	idler	planet	carrier	TR	wheel rotational speed (rad/s)	forward speed	sun	idler	planet	carrier	TR	Carrier rotational speed (rad/sec)
0.00170	0.15	0.085	50.0	0.007	4.4	0.120	70.7	1.0	-11.5	0.7	0.0	0.7	2.0	0.2	0.3	-12.2	0.0	-0.7	-2.4	-1.1
0.00170	0.17	0.085	50.0	0.014	8.3	0.107	62.7	1.0	-6.0	0.8	0.0	0.8	2.3	0.2	0.2	-6.8	0.0	-0.8	-3.9	-1.8
0.00170	0.19	0.085	50.0	0.020	11.5	0.096	56.4	1.0	-4.3	0.9	0.0	0.9	2.5	0.2	0.1	-5.2	0.0	-0.9	-7.8	-3.5
0.00170	0.21	0.085	50.0	0.024	14.1	0.087	51.2	1.0	-3.5	1.0	0.0	1.0	2.8	0.2	0.0	-4.5	0.0	-1.0	-40.9	-18.4
0.00170	0.23	0.085	50.0	0.028	16.2	0.080	46.9	1.0	-3.1	1.1	0.0	1.1	3.0	0.2	-0.1	-4.1	0.0	-1.1	16.2	7.3
0.00170	0.25	0.085	50.0	0.031	18.1	0.074	43.3	1.0	-2.8	1.2	0.0	1.2	3.3	0.2	-0.2	-3.9	0.0	-1.2	7.4	3.3
0.00170	0.26	0.085	50.0	0.033	19.6	0.068	40.2	1.0	-2.5	1.2	0.0	1.2	3.5	0.3	-0.2	-3.8	0.0	-1.2	5.1	2.3
0.00170	0.28	0.085	50.0	0.036	21.0	0.064	37.5	1.0	-2.4	1.3	0.0	1.3	3.8	0.3	-0.3	-3.7	0.0	-1.3	4.0	1.8
0.00170	0.30	0.085	50.0	0.038	22.2	0.060	35.1	1.0	-2.3	1.4	0.0	1.4	4.0	0.3	-0.4	-3.7	0.0	-1.4	3.4	1.5
0.00170	0.32	0.085	50.0	0.039	23.2	0.056	33.0	1.0	-2.2	1.5	0.0	1.5	4.3	0.3	-0.5	-3.7	0.0	-1.5	2.9	1.3
0.00170	0.34	0.085	50.0	0.041	24.1	0.053	31.2	1.0	-2.1	1.6	0.0	1.6	4.5	0.3	-0.6	-3.7	0.0	-1.6	2.7	1.2
0.00170	0.36	0.085	50.0	0.042	24.9	0.050	29.5	1.0	-2.0	1.7	0.0	1.7	4.8	0.4	-0.7	-3.7	0.0	-1.7	2.4	1.1

0.00170	0.38	0.085	50.0	0.044	25.7	0.048	28.0	1.0	-1.9	1.8	0.0	1.8	5.0	0.4	-0.8	-3.7	0.0	-1.8	2.3	1.0
0.00170	0.40	0.085	50.0	0.045	26.4	0.045	26.7	1.0	-1.9	1.9	0.0	1.9	5.3	0.4	-0.9	-3.8	0.0	-1.9	2.1	1.0
0.00170	0.42	0.085	50.0	0.046	27.0	0.043	25.5	1.0	-1.9	2.0	0.0	2.0	5.5	0.4	-1.0	-3.8	0.0	-2.0	2.0	0.9
0.00170	0.44	0.085	50.0	0.047	27.5	0.041	24.4	1.0	-1.8	2.1	0.0	2.1	5.8	0.4	-1.1	-3.9	0.0	-2.1	2.0	0.9
0.00170	0.45	0.085	50.0	0.048	28.0	0.040	23.4	1.0	-1.8	2.1	0.0	2.1	6.1	0.5	-1.1	-3.9	0.0	-2.1	1.9	0.8
0.00170	0.47	0.085	50.0	0.048	28.5	0.038	22.4	1.0	-1.8	2.2	0.0	2.2	6.3	0.5	-1.2	-4.0	0.0	-2.2	1.8	0.8
0.00170	0.49	0.085	50.0	0.049	28.9	0.037	21.6	1.0	-1.7	2.3	0.0	2.3	6.6	0.5	-1.3	-4.0	0.0	-2.3	1.8	0.8
0.00170	0.51	0.085	50.0	0.050	29.3	0.035	20.7	1.0	-1.7	2.4	0.0	2.4	6.8	0.5	-1.4	-4.1	0.0	-2.4	1.7	0.8
0.00170	0.53	0.085	50.0	0.050	29.7	0.034	20.0	1.0	-1.7	2.5	0.0	2.5	7.1	0.5	-1.5	-4.2	0.0	-2.5	1.7	0.8
0.00170	0.55	0.085	50.0	0.051	30.0	0.033	19.3	1.0	-1.7	2.6	0.0	2.6	7.3	0.5	-1.6	-4.3	0.0	-2.6	1.6	0.7
0.00170	0.57	0.085	50.0	0.052	30.4	0.032	18.7	1.0	-1.6	2.7	0.0	2.7	7.6	0.6	-1.7	-4.3	0.0	-2.7	1.6	0.7
0.00170	0.59	0.085	50.0	0.052	30.7	0.031	18.1	1.0	-1.6	2.8	0.0	2.8	7.8	0.6	-1.8	-4.4	0.0	-2.8	1.6	0.7
0.00170	0.61	0.085	50.0	0.053	31.0	0.030	17.5	1.0	-1.6	2.9	0.0	2.9	8.1	0.6	-1.9	-4.5	0.0	-2.9	1.5	0.7
0.00170	0.63	0.085	50.0	0.053	31.2	0.029	17.0	1.0	-1.6	2.9	0.0	2.9	8.3	0.6	-1.9	-4.5	0.0	-2.9	1.5	0.7
0.00170	0.64	0.085	50.0	0.054	31.5	0.028	16.5	1.0	-1.6	3.0	0.0	3.0	8.6	0.6	-2.0	-4.6	0.0	-3.0	1.5	0.7
0.00170	0.66	0.085	50.0	0.054	31.7	0.027	16.0	1.0	-1.6	3.1	0.0	3.1	8.8	0.7	-2.1	-4.7	0.0	-3.1	1.5	0.7
0.00170	0.68	0.085	50.0	0.054	31.9	0.026	15.5	1.0	-1.6	3.2	0.0	3.2	9.1	0.7	-2.2	-4.8	0.0	-3.2	1.5	0.7
0.00170	0.70	0.085	50.0	0.055	32.1	0.026	15.1	1.0	-1.6	3.3	0.0	3.3	9.3	0.7	-2.3	-4.9	0.0	-3.3	1.4	0.6
0.00170	0.72	0.085	50.0	0.055	32.3	0.025	14.7	1.0	-1.5	3.4	0.0	3.4	9.6	0.7	-2.4	-4.9	0.0	-3.4	1.4	0.6
0.00170	0.74	0.085	50.0	0.055	32.5	0.024	14.3	1.0	-1.5	3.5	0.0	3.5	9.9	0.7	-2.5	-5.0	0.0	-3.5	1.4	0.6
0.00170	0.76	0.085	50.0	0.056	32.7	0.024	14.0	1.0	-1.5	3.6	0.0	3.6	10.1	0.8	-2.6	-5.1	0.0	-3.6	1.4	0.6
0.00170	0.78	0.085	50.0	0.056	32.9	0.023	13.6	1.0	-1.5	3.7	0.0	3.7	10.4	0.8	-2.7	-5.2	0.0	-3.7	1.4	0.6
0.00170	0.80	0.085	50.0	0.056	33.0	0.023	13.3	1.0	-1.5	3.8	0.0	3.8	10.6	0.8	-2.8	-5.3	0.0	-3.8	1.4	0.6
0.00170	0.82	0.085	50.0	0.056	33.2	0.022	13.0	1.0	-1.5	3.8	0.0	3.8	10.9	0.8	-2.8	-5.3	0.0	-3.8	1.4	0.6
0.00170	0.83	0.085	50.0	0.057	33.3	0.022	12.7	1.0	-1.5	3.9	0.0	3.9	11.1	0.8	-2.9	-5.4	0.0	-3.9	1.3	0.6
0.00170	0.85	0.085	50.0	0.057	33.5	0.021	12.4	1.0	-1.5	4.0	0.0	4.0	11.4	0.9	-3.0	-5.5	0.0	-4.0	1.3	0.6
0.00170	0.87	0.085	50.0	0.057	33.6	0.021	12.2	1.0	-1.5	4.1	0.0	4.1	11.6	0.9	-3.1	-5.6	0.0	-4.1	1.3	0.6
0.00170	0.89	0.085	50.0	0.057	33.8	0.020	11.9	1.0	-1.5	4.2	0.0	4.2	11.9	0.9	-3.2	-5.7	0.0	-4.2	1.3	0.6
0.00170	0.91	0.085	50.0	0.058	33.9	0.020	11.7	1.0	-1.5	4.3	0.0	4.3	12.1	0.9	-3.3	-5.8	0.0	-4.3	1.3	0.6
0.00170	0.93	0.085	50.0	0.058	34.0	0.019	11.4	1.0	-1.5	4.4	0.0	4.4	12.4	0.9	-3.4	-5.9	0.0	-4.4	1.3	0.6

0.00170	0.95	0.085	50.0	0.058	34.1	0.019	11.2	1.0	-1.5	4.5	0.0	4.5	12.6	0.9	-3.5	-5.9	0.0	-4.5	1.3	0.6
0.00170	0.97	0.085	50.0	0.058	34.2	0.019	11.0	1.0	-1.5	4.6	0.0	4.6	12.9	1.0	-3.6	-6.0	0.0	-4.6	1.3	0.6
0.00170	0.99	0.085	50.0	0.058	34.3	0.018	10.8	1.0	-1.5	4.6	0.0	4.6	13.1	1.0	-3.6	-6.1	0.0	-4.6	1.3	0.6
0.00170	1.01	0.085	50.0	0.059	34.4	0.018	10.6	1.0	-1.5	4.7	0.0	4.7	13.4	1.0	-3.7	-6.2	0.0	-4.7	1.3	0.6
0.00170	1.02	0.085	50.0	0.059	34.5	0.018	10.4	1.0	-1.4	4.8	0.0	4.8	13.7	1.0	-3.8	-6.3	0.0	-4.8	1.3	0.6
0.00170	1.04	0.085	50.0	0.059	34.6	0.017	10.2	1.0	-1.4	4.9	0.0	4.9	13.9	1.0	-3.9	-6.4	0.0	-4.9	1.3	0.6
0.00170	1.06	0.085	50.0	0.059	34.7	0.017	10.0	1.0	-1.4	5.0	0.0	5.0	14.2	1.1	-4.0	-6.4	0.0	-5.0	1.2	0.6
0.00170	1.08	0.085	50.0	0.059	34.8	0.017	9.8	1.0	-1.4	5.1	0.0	5.1	14.4	1.1	-4.1	-6.5	0.0	-5.1	1.2	0.6
0.00170	1.10	0.085	50.0	0.059	34.9	0.016	9.6	1.0	-1.4	5.2	0.0	5.2	14.7	1.1	-4.2	-6.6	0.0	-5.2	1.2	0.6
0.00170	1.12	0.085	50.0	0.059	35.0	0.016	9.5	1.0	-1.4	5.3	0.0	5.3	14.9	1.1	-4.3	-6.7	0.0	-5.3	1.2	0.6
0.00170	1.14	0.085	50.0	0.060	35.0	0.016	9.3	1.0	-1.4	5.4	0.0	5.4	15.2	1.1	-4.4	-6.8	0.0	-5.4	1.2	0.6
0.00170	1.16	0.085	50.0	0.060	35.1	0.016	9.2	1.0	-1.4	5.5	0.0	5.5	15.4	1.2	-4.5	-6.9	0.0	-5.5	1.2	0.6
0.00170	1.18	0.085	50.0	0.060	35.2	0.015	9.0	1.0	-1.4	5.5	0.0	5.5	15.7	1.2	-4.5	-7.0	0.0	-5.5	1.2	0.5
0.00170	1.20	0.085	50.0	0.060	35.3	0.015	8.9	1.0	-1.4	5.6	0.0	5.6	15.9	1.2	-4.6	-7.1	0.0	-5.6	1.2	0.5
0.00170	1.21	0.085	50.0	0.060	35.3	0.015	8.7	1.0	-1.4	5.7	0.0	5.7	16.2	1.2	-4.7	-7.1	0.0	-5.7	1.2	0.5
0.00170	1.23	0.085	50.0	0.060	35.4	0.015	8.6	1.0	-1.4	5.8	0.0	5.8	16.4	1.2	-4.8	-7.2	0.0	-5.8	1.2	0.5
0.00170	1.25	0.085	50.0	0.060	35.5	0.014	8.5	1.0	-1.4	5.9	0.0	5.9	16.7	1.3	-4.9	-7.3	0.0	-5.9	1.2	0.5

10.3.2 Kinematic Derivations

The kinematic derivations can be found in the Di-Wheel Google Drive, in a PDF scanned document.

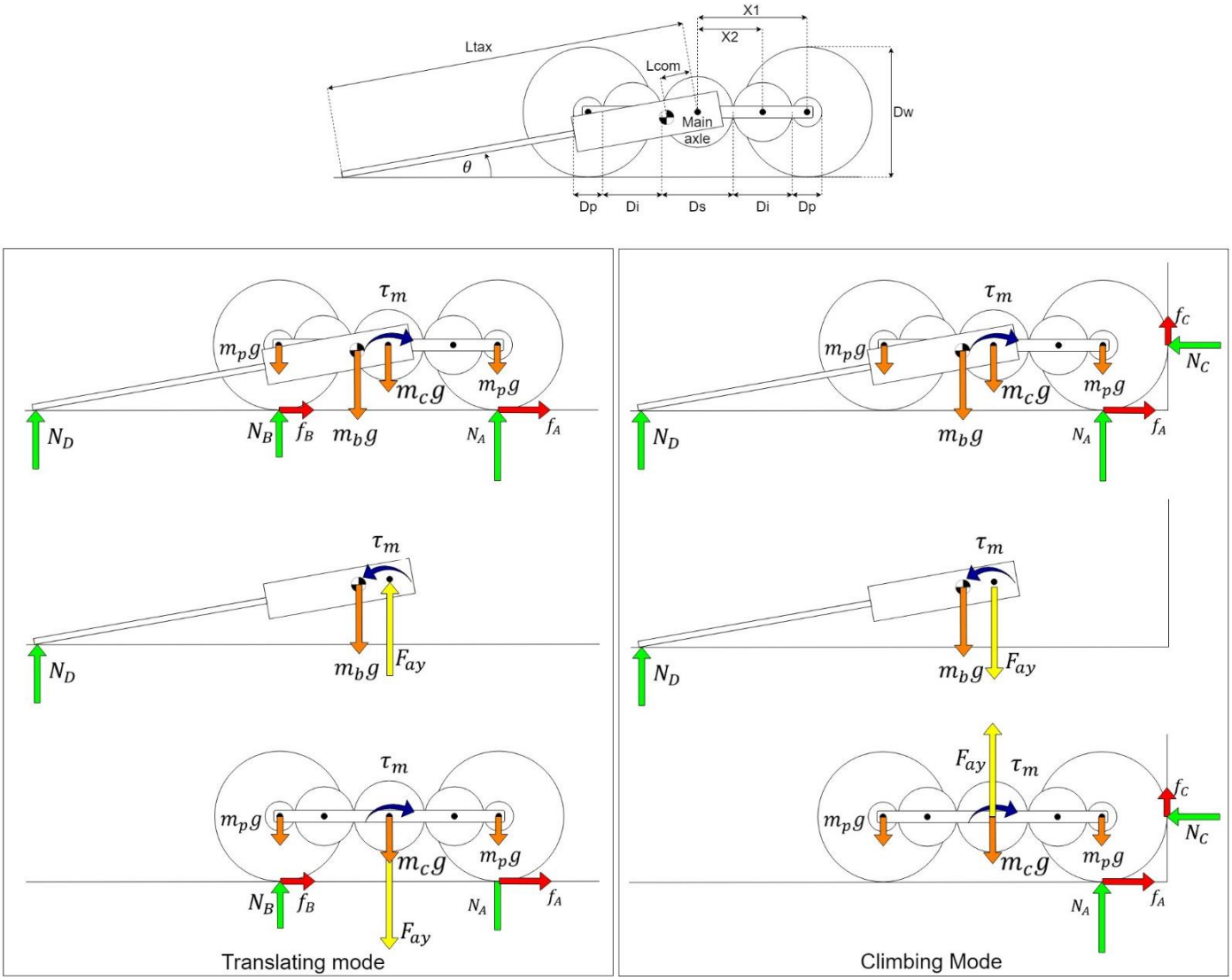


Figure 63: Depiction of the forces acting on the whole system, the body and Di-Wheel in translating and climbing mode (vectors to approx. scale)

10.3.2.1.1 Simplified Calculations using Assumptions

General Assumptions	
The mass of the gears is negligible (except for the planet which is included in the wheel mass)	
Translating Mode Assumptions	**Climbing Mode Assumptions**
Dynamic calculations, starting from rest and about to start moving as the torque is applied from the motor.	Dynamic calculations, starting from rest and about to start climbing.
	The coefficient of friction is large enough that the front wheel is not rotating.

All the calculations were made with reference to Figure 63, with the following definition of terms. Where there is a subscript x, it can be replaced by b – body, p – planet, i – idler, s – sun, c – carrier, w

– wheel, and subscript y can be replaced by A, B, C, D which are points of contact with the ground or stair

$\boldsymbol{m_x}$	Mass of bodies (mass of planet includes wheel mass)	$\boldsymbol{\tau_m}$	Motor torque	$\boldsymbol{N_y}$	Normal force at points of contact
$\boldsymbol{g, \mu}$	Gravity and coefficient of friction	$\boldsymbol{X_1}$	Distance from main axel to planet gear axel	$\boldsymbol{f_y}$	Friction force at points of contact
$\boldsymbol{L_{tax}}$	Distance from main axle to end of tail	$\boldsymbol{X_2}$	Distance from main axel to idler gear axel	$\boldsymbol{F_{ax}\, F_{ay}}$	The x and y components of the force on the main axle
$\boldsymbol{L_{com}}$	Distance from main axle to centre of mass	$\boldsymbol{D_x}$	Diameter of gears and wheels	$\boldsymbol{R_x}$	Radius of gears and wheels

Translating mode:

Considering the carrier:

$$\circlearrowleft \sum M_{axle(carrier)} = 0 = -\tau_m - N_B X_1 + m_p g X_1 + F_B R_w - m_p g\, X_1 + N_A X_1 + f_A R_w$$
$$\therefore \tau_m = -N_B(X_1 - \mu R_w) + N_A(X_1 + \mu R_w)$$

$$\sum F_y = 0 = N_B + N_A - 2m_p g - m_c g + F_{ay}$$
$$\therefore N_A = 2m_p g + m_c g - N_B - F_{ay}$$
$$\sum F_x = (m_c + 2m_p)a_x = -F_{ax} + f_A + f_B = -F_{ax} + \mu(N_A + N_B)$$
$$\therefore (m_c + 2m_p)a_x = -F_{ax} + \mu\left((m_c + 2m_p)g - F_{ay}\right)$$

Considering the Body:

$$\sum F_x = m_b a_x = F_{ax}$$
$$\sum F_y = 0 = N_D - m_b g - F_{ay}$$
$$\therefore N_D = m_b g + F_{ay}$$

$$\circlearrowleft \sum M_{axle(body)} = 0 = \tau_m + m_b g L_{com}\cos(\theta_b) - N_D L_{tax}\cos(\theta_b)$$
$$= \tau_m + m_b g L_{com}\cos(\theta_b) - (m_b g + F_{ay})L_{tax}\cos(\theta_b)$$
$$\therefore F_{ay} = \frac{\tau_m + m_b g L_{com}\cos(\theta_b)}{L_{tax}\cos(\theta_b)} - m_b g$$

Substitute in:

$$\therefore (m_c + 2m_p)a_x = -m_b a_x + \mu\left((m_c + 2m_p)g - \frac{\tau_m + m_b g L_{com}\cos(\theta_b)}{L_{tax}\cos(\theta_b)} + m_b g\right)$$
$$(m_c + 2m_p + m_b)a_x = \mu\left((m_c + 2m_p + m_b)g - \frac{\tau_m + m_b g L_{com}\cos(\theta_b)}{L_{tax}\cos(\theta_b)}\right)$$
$$a_x = \mu g + \frac{-\mu(\tau_m + m_b g L_{com}\cos(\theta_b))}{L_{tax}\cos(\theta_b)\,(m_c + 2m_p + m_b)}$$
$$a_x = \mu g\left(1 - \frac{m_b L_{com}}{L_{tax}(m_c + 2m_p + m_b)}\right) - \frac{\mu}{L_{tax}\cos(\theta_b)\,(m_c + 2m_p + m_b)}\tau_m$$

$$\text{as}\cos(\theta_b) = \frac{\sqrt{L_{tax}^2 - R_w^2}}{L_{tax}}$$

$$a_x = \mu g\left(1 - \frac{m_b L_{com}}{L_{tax}(m_c + 2m_p + m_b)}\right) - \frac{\mu}{\sqrt{L_{tax}^2 - R_w^2}(m_c + 2m_p + m_b)}\tau_m$$

Climbing Mode:

Carrier:

$$\circlearrowleft \sum M_{axle(carrier)} = I_c\alpha_c = m_p g X_1 - m_p g X_1 - \tau_m + F_c(X_1 + R_w) + N_A X_1 + F_A R_w$$
$$I_c\alpha_c = -\tau_m + F_c(X_1 + R_w) + N_A(X_1 + \mu R_w)$$

$$\sum F_y = (m_c + 2m_p)a_y = \mu N_c + N_A - 2m_p g - m_c g - F_{ay}$$
$$(m_c + 2m_p)\alpha_c X_1 = \mu N_c + N_A - 2m_p g - m_c g - F_{ay}$$

$\sum F_x = 0 = F_A - N_C$ as there is no acceleration in the x direction for that moment
$$\therefore \mu N_A = N_C$$
$$\therefore (m_c + 2m_p)\alpha_c X_1 = (\mu^2 + 1)N_A - (2m_p + m_c)g - F_{ay}$$
$$N_A = \frac{(m_c + 2m_p)(\alpha_c X_1 + g) + F_{ay}}{(\mu^2 + 1)}$$

substitute into moment equation

$$I_c\alpha_c = -\tau_m + \mu^2 N_A(X_1 + R_w) + N_A(X_1 + \mu R_w)$$
$$= -\tau_m + N_A\left(X_1(\mu^2 + 1) + R_w(\mu^2 + \mu)\right)$$
$$= -\tau_m + \frac{\left[(m_c + 2m_p)(\alpha_c X_1 + g) + F_{ay}\right]}{(\mu^2 + 1)}\left(X_1(\mu^2 + 1) + R_w(\mu^2 + \mu)\right)$$
$$= -\tau_m + (m_c + 2m_p)(\alpha_c X_1 + g)\left(X_1 + \frac{R_w(\mu^2 + \mu)}{(\mu^2 + 1)}\right) + F_{ay}\left(X_1 + \frac{R_w(\mu^2 + \mu)}{(\mu^2 + 1)}\right)$$

$$= -\tau_m + \alpha_c X_1\left(X_1(m_c + 2m_p) + \frac{R_w(\mu^2 + \mu)(m_c + 2m_p)}{(\mu^2 + 1)}\right)$$
$$+ g\left(X_1(m_c + 2m_p) + \frac{R_w(\mu^2 + \mu)(m_c + 2m_p)}{(\mu^2 + 1)}\right)$$
$$+ F_{ay}\left(X_1 + \frac{R_w(\mu^2 + \mu)}{(\mu^2 + 1)}\right)$$

$$\alpha_c = \frac{\left[-\tau_m + g(m_c + 2m_p)\left(X_1 + \frac{R_w(\mu^2 + \mu)}{(\mu^2 + 1)}\right) + F_{ay}\left(X_1 + \frac{R_w(\mu^2 + \mu)}{(\mu^2 + 1)}\right)\right]}{\left(I_c - X_1(m_c + 2m_p)\left(X_1 + \frac{R_w(\mu^2 + \mu)}{(\mu^2 + 1)}\right)\right)}$$

$$\alpha_c = \frac{\left[-\tau_m + \left(X_1 + \frac{R_w(\mu^2 + \mu)}{(\mu^2 + 1)}\right)(g(m_c + 2m_p) + F_{ay})\right]}{\left(I_c - X_1(m_c + 2m_p)\left(X_1 + \frac{R_w(\mu^2 + \mu)}{(\mu^2 + 1)}\right)\right)}$$

Body:

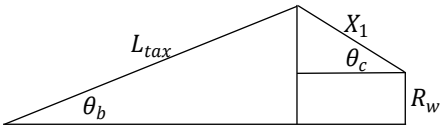
$$\circlearrowleft \sum M_{axle(body)} = I_b\alpha_b = \tau_m + m_b g L_{com}\cos(\theta_b) - N_D L_{tax}\cos(\theta_b)$$

$$\sum F_x = 0$$

$$\sum F_y = m_b a_y = N_D + F_{ay} - m_b g$$
$$m_b \alpha_b (L_{tax} - L_{com}) = N_D + F_{ay} - m_b g$$

$$\therefore F_{ay} = m_b \alpha_b (L_{tax} - L_{com}) - N_D + m_b g$$

The relationship between α_b and α_c (using small angle approximation)



$$\sin(\theta_b) = \frac{R_w + X_1 \sin(\theta_c)}{L_{tax}} \Longrightarrow \theta_b = \frac{R_w}{L_{tax}} + \frac{X_1}{L_{tax}}\theta_c$$
$$\omega_b = \frac{X_1}{L_{tax}}\omega_c \implies \alpha_b = \frac{X_1}{L_{tax}}\alpha_c$$

$$\therefore N_D = m_b \frac{X_1}{L_{tax}}\alpha_c (L_{tax} - L_{com}) - F_{ay} + m_b g$$

$$\therefore I_B \frac{X_1}{L_{tax}}\alpha_c = \tau_m + m_b g L_{com}\cos(\theta_b) - \left(m_b \frac{X_1}{L_{tax}}\alpha_c (L_{tax} - L_{com}) - F_{ay} + m_b g\right) L_{tax}\cos(\theta_b)$$

$$\therefore F_{ay} = \left(I_B \frac{X_1}{L_{tax}^2 \cos(\theta_b)} + m_b \frac{X_1}{L_{tax}}(L_{tax} - L_{com})\right)\alpha_c - \frac{\left(\tau_m + m_b g L_{com}\cos(\theta_b)\right)}{L_{tax}\cos(\theta_b)} + m_b g$$

let $A_1 = I_B \frac{X_1}{L_{tax}^2 \cos(\theta_b)} + m_b \frac{X_1}{L_{tax}}(L_{tax} - L_{com})$,

and $A_2 = \frac{\left(\tau_m + m_b g L_{com}\cos(\theta_b)\right)}{L_{tax}\cos(\theta_b)} - m_b g$

$$F_{ay} = A_1 \alpha_c - A_2$$

Substitute into final carrier equation:

$$\alpha_c = \frac{\left[-\tau_m + \left(X_1 + \frac{R_w(\mu^2 + \mu)}{(\mu^2 + 1)}\right)\left(g\left(m_c + 2m_p\right) + F_{ay}\right)\right]}{\left(I_c - X_1\left(m_c + 2m_p\right)\left(X_1 + \frac{R_w(\mu^2 + \mu)}{(\mu^2 + 1)}\right)\right)}$$

$$let\ A_3 = X_1 + \frac{R_w(\mu^2 + \mu)}{(\mu^2 + 1)}$$
$$and\ A_4 = -\tau_m + A_3\, g\left(m_c + 2m_p\right),$$
$$and\ A_5 = I_c - X_1\left(m_c + 2m_p\right)A_3$$

$$\alpha_c = \frac{\left[A_4 + A_3 F_{ay}\right]}{A_5}$$
$$\alpha_c = \frac{[A_4 + A_3(A_1\alpha_c - A_2)]}{A_5} = \frac{[A_4 + A_3 A_1 \alpha_c - A_3 A_2]}{A_5}$$

$$\alpha_c = \frac{A_4 - A_3A_2}{A_5 - A_3A_1} = \frac{-\tau_m + A_3\, g(m_c + 2m_p) - A_3\left(\frac{(\tau_m + m_b g L_{com}\cos(\theta_b))}{L_{tax}\cos(\theta_b)} - m_b g\right)}{A_5 - A_3A_1}$$

$$= \frac{-\tau_m\left(1 + \frac{A_3}{L_{tax}\cos(\theta_b)}\right) + A_3\, g(m_c + 2m_p) + A_3 m_b g\left(1 - \frac{L_{com}}{L_{tax}}\right)}{A_5 - A_3A_1}$$

$$= \tau_m \frac{\left(1 + \frac{A_3}{L_{tax}\cos(\theta_b)}\right)}{-A_5 + A_3A_1} - \frac{A_3\, g(m_c + 2m_p) + A_3 m_b g\left(1 - \frac{L_{com}}{L_{tax}}\right)}{-A_5 + A_3A_1}$$

Change of sign due to expected sign of A_5, as I_c should be smaller than the second term

10.3.3 Lagrangian Derivations

The Lagrangian Derivations can be found in the Di-Wheel Google Drive, in a PDF scanned document. There was also MATLAB code run to calculate the results due to the complexity of the matrices involved. This is also available in the Google drive.

10.4 Bill of Materials

The full bill of materials is shown below. There were various items that were already owned, and these are shown as sunk costs. The alternatives that would have been purchased if these were not already owned is shown as well and the comparative cost difference between the two options. The only cost actually spent on this project is the current cost, however the actual component worth of the robot is the final total cost.

Table 14: Bill of Materials

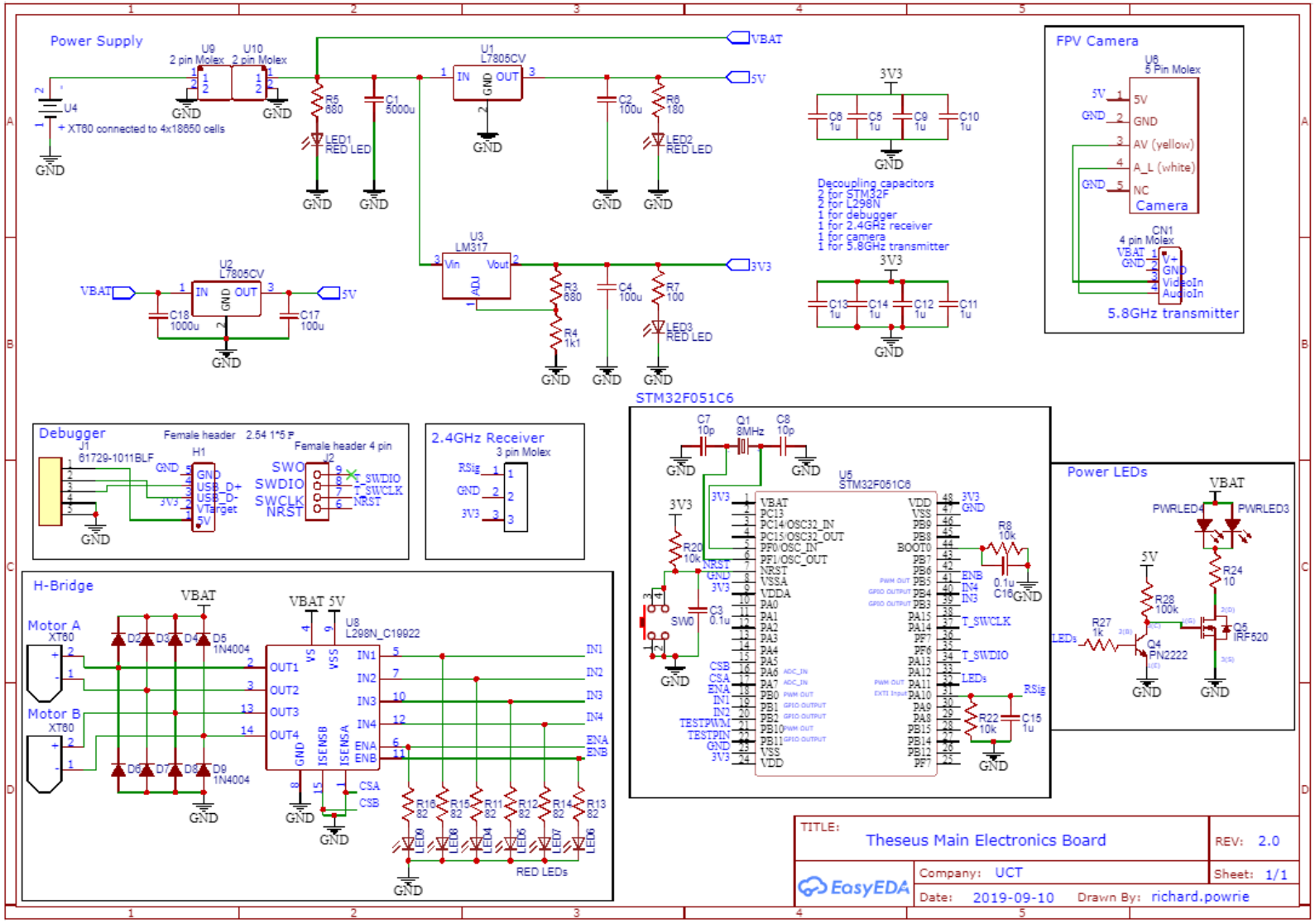
				Current Cost		Sunk Cost		Alternative to sunk Cost		
Item	Description	Source	Quantity	Per unit cost	Total price	Per unit cost	Total price	Alternative Item	Per unit cost	Total price
motor	12V motors, 270RPM	Mantech	2			R250.47	R500.94	same	R250.47	R500.94
orange RX R617xl	2.4GHz receiver	hobbyking	1			R166.50	R166.50	don't need		
Spektrum DX8	spektrum dx8 transmitter	ubuy	1			R5 594.00	R5 594.00	FlySky FS i6	R608.00	R608.00
AOMWAY 600 CMOS	Camera	hobbyking	1			R600.00	R600.00	aomway camera	R640.00	R640.00
FT952	5.8GHz transmitter	flyingtech	1			R480.00	R480.00	don't need		
STM32F051C6	STM microcontroller	White Lab	1	R38.00	R38.00					
STM32fl	Debugger	White Lab	1	R75.00	R75.00					
61729-0010BLF	USB connector	White Lab	1	R14.90	R14.90					
veroboard	veroboard	White Lab	1	R70.00	R70.00					
acm star power led white	Power white LED	communica	2	R45.00	R90.00					
18650 holder	18650 battery holder	communica	2	R28.00	R56.00					
18650 cells	18650 cells	micro robotics	4	R112.70	R450.80					
18650 charger	2-cell charger	communica	1	R85.00	R85.00					
L298 H-bridge chip	L298N H-Bridge chip	mantech	1	R62.52	R62.52					
LM317	voltage regulator	White Lab	2							
8MHz crystal	8MHz quartz crystal	White Lab	1							
pushbutton	pushbutton	White Lab	1							
zener diode	3v3 zener diode	White Lab	1							
PN222	NPN transistor	White Lab	1							
Irf520	power mosfet	White Lab	1							
Red	3mm Red LEDs	White Lab	9							

1n4004	power diodes	White Lab	8							
lazer cutting										
hardboard	3mm A4 sheet	White Lab	6	R10.00	R60.00					
perspex	3mm A4 sheet	White Lab	6	R30.00	R180.00					
Resistors										
680R		White Lab	2							
1k1		White Lab	1							
1k		White Lab	1							
10k		White Lab	1							
330R		White Lab	1							
82R		White Lab	6							
180R		White Lab	1							
100R		White Lab	1							
150k		White Lab	1							
10R		White Lab	4							
100k		White Lab	2							
Capacitors										
1uF		White Lab	10							
10pF		White Lab	2							
10uF		White Lab	1							
Connectors										
5 pin molex	male	White Lab	1							
5 pin molex	female	White Lab	1							
4 pin molex	male	White Lab	1							
4 pin molex	female	White Lab	1							
3 pin molex	male	White Lab	1							
3 pin molex	female	White Lab	1							
2 pin molex	male	White Lab	4							
2 pin molex	female	White Lab	4							
Headers										
5 pin	female	White Lab	2							
5 pin	male	White Lab	2							
6 pin	male	White Lab	2							

4 pin	female	White Lab	1							
4 pin	male	White Lab	2							
3 pin	female	White Lab	4							
3 pin	male	White Lab	5							
24 pin	female	White Lab	2							
24 pin	male	White Lab	2							
Fasteners										
M3x6	motor fasteners	Power lab	12							
M3x16	light and Di-Wheel fasteners	Power lab	16							
M3x30	circuit board fasteners	Power lab	4							
M3 nuts	light and Di-Wheel nuts	Power lab	16							
M5 nuts	nuts on wheels	Power lab	8							
M5 locknuts	locknuts on wheels	Power lab	4							
M5 washers	Di-Wheel washers	Power lab	28							
1.5mm Hitch pins	main shaft pins	Power lab	8							
			Total costs	Current	R1 182.22	Sunk	R7 341.44		Alternative	R1 748.94
						Final total	R8 523.66		**Alternative final total**	R2 931.16

10.5 Circuit Diagrams

The full circuit diagram is shown below. This can also be accessed in [Di-Wheel Google Drive].



10.6 Component Selection

The following table shows the options for the component selection.

Battery holder					
18650 4-cell holder	micro robotics	1	50	50	in stock
18650 2-cell holder	communica	2	31.05	62.1	out of stock
18650 2-cell holder	communica	2	28	56	
Battery Charger					
18650 charger 2-cell	flying robot	1	150	150	2 cell
18650 charger 2-cell	communica	1	85	85	2 cell
18650 charger 4-cell	communica	1	175	175	4 cell
Camera					
tbs tiny camera	flying robot	1	250	250	
fx798 vtx camera	flying robot	1	500	500	comes with transmitter
EWRF-TS5823	banggood	1	263	263	comes with transmitter
LS-F200KT	banggood	1	330	330	
aomway fpv camera	rotorlogic	1	640	640	comes with transmitter
5.8GHz transmitter					
ft952	flyingtech	1	480		
2.4GHz receiver					
orangerx r617xl	hobbyking	1	167	167	DSM2/DSMx and cPPM
orangerx r616xn	flying robot	1	328	328	DSM2/DSMx and cPPM
micro dsmx receiver	flying robot	1	299	299	DSM2/DSMx and SBUS
orangerx r618xl	flying robot	1	201	201	DSM2/DSMx and cPPM
2.4GHz Transmitter					
spektrum dx8 transmitter	ubuy	1	5594	5594	6 channel
flysky fs i6x transmitter	flying robot	1	1149	1149	6 channel
522 flysky fs gt3c	RC King	1	1095	1095	3 channel
WLtoys V911S	banggood	1	315.17	315.17	4 channel
FlySky FS i6	banggood	1	608	608	6 channel, comes with receiver
HSP	banggood	1	404.5	404.5	3 channel, comes with receiver
Lights					
acm star power led white	communica	4	45	180	5W each
3121 high bright led	netram	4	45	180	5W each
3120 high bright led	netram	4	50	200	10w each

10.6.1 2.4GHz Receiver Selection

The selection of the 2.4GHz receiver/transmitter pair required some thought about the various protocols and types of receivers involved. The exact transmitter/ receiver pair must be compatible. Certain

protocols are universal, while others are proprietary, and you need a specific brand to be compatible with a device using that protocol. These types are [25]:

RX Protocols

- PWM (universal)
- PPM (universal)
- PCM (universal)
- SBUS (Futaba, Frsky)
- IBUS (Flysky)
- XBUS (JR)
- MSP (Multiwii)
- SUMD (Graupner)
- SUMH (Graupner)
- CRSF – Crossfire (TBS)
- FPort (Frsky)
- SPI_RX (universal)

TX Protocols

- D8 (Frsky)
- D16 (Frsky)
- LR12 (Frsky)
- DSM (Spektrum)
- DSM2 (Spektrum)
- DSMX (Spektrum)
- AFHDS (Flysky)
- AFHDS 2A (Flysky)
- A-FHSS (Hitec)
- FASST (Futaba)
- Hi-Sky (Deviation)

It was decided that whatever transmitter receiver pair were selected, they must use PPM because this would be easiest to implement, requiring a single signal wire. It is also one of the most common protocols amongst receivers [25]. PWM is also very common, but requires a signal wire for every channel, which is not desirable.

Various transmitters and receivers were considered, as well as a Spektrum DX8 transmitter and Orange RX R617xl receiver which were already owned. Refer to Section 10.6 of the Appendix for a detailed analysis of these various options. Due to the price of transmitters and receivers, and the lead time required for delivery for the best options, it was decided to use the transmitter and receiver already owned. There were however alternatives selected, and these are included in the Bill of Materials, found section 10.4.

10.7 SOFTWARE

The full software for this project was created using Atollic TrueStudio and CubeMX. The project files for this are available at the Di-Wheel Google Drive. Some excerpts are included here for reference.

10.7.1 Interrupt Handler Function for PPM Recording

```
void EXTI4_15_IRQHandler(void)
{
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_10);

  uint32_t currentTime= recordTime();

  if(flags[0]==0){//set CH0 to current value if zero (meaning the start of the message)
          // CH0 is approximately 13.2ms
          CH[0] = currentTime;
          flags[0]=1;
  }
  else if(flags[1]==0){//set CH1 to current value if zero, and previous channels not zero
          CH[1] = currentTime-CH[0];
          CH[0] = currentTime;// reset CH0 for next channel input
          flags[1]=1;
          Synch(CH[1]);
  }
  else if(flags[2]==0){//set CH2 to current value if zero, and previous channels not zero
          CH[2] = currentTime-CH[0];
          CH[0] = currentTime;// reset CH0 for next channel input
          flags[2]=1;
          Synch(CH[2]);
```

```
        if(flags[2]==1){//if it didn't synch
                //invert action of channel 2
                CH[2] = PPMMAX-CH[2]+PPMMIN;
        }
        //setTestPWM(CH[2]-PPMMIN);//uncomment to see pwm change according to CH2
  }
  else if(flags[3]==0){//set CH3 to current value if zero, and previous channels not zero
        CH[3] = currentTime-CH[0];
        CH[0] = currentTime;// reset CH0 for next channel input
        flags[3]=1;
        Synch(CH[3]);
  }
  else if(flags[4]==0){//set CH4 to current value if zero, and previous channels not zero
        CH[4] = currentTime-CH[0];
        CH[0] = currentTime;// reset CH0 for next channel input
        flags[4]=1;
        Synch(CH[4]);
  }
  else if(flags[5]==0){//set CH5 to current value if zero, and previous channels not zero
        CH[5] = currentTime-CH[0];
        CH[0] = currentTime;// reset CH0 for next channel input
        flags[5]=1;
        Synch(CH[5]);
  }
  else if(flags[6]==0){//set CH6 to current value if zero, and previous channels not zero
        CH[6] = currentTime-CH[0];
        CH[0] = currentTime;// reset CH0 for next channel input
        flags[6]=1;
        Synch(CH[6]);

        if(flags[6]==1){//if it didn't synch
                //invert action of channel 2
                CH[6] = PPMMAX-CH[6]+PPMMIN;//invert action of channel 2
                // end of message, reset for next message
                SetFlags(0,0,0,0,0,0,0);
                resetTime();
        }
  }
}
```

10.7.2 Interpret Function

```
void Interpret(void){//expects values from PPMMIN->PPMMAX from each channel
        int *CH= ReadChannels();//pointer to array for channels

        int CHpower = *(CH+CHPOWER);                    //dereference power channel
        int CHturn = *(CH+CHTURN);                      //dereference turn channel
        int CHlight = *(CH+CHLIGHT);                    //dereference light channel

        //Motor control
        int forwardA = 0;//percentage forward power on motorA (-100 reverse, 0 stop, 100 forward)
        int forwardB = 0;//percentage forward power on motorB (-100 reverse, 0 stop, 100 forward)


        //map values from channels to -100 to 100 range
        int power = (CHpower-PPMMID)*100/(PPMMAX-PPMMID);
        int turn  = (CHturn-PPMMID)*100/(PPMMAX-PPMMID);


        //set forward power on each motor, according to acceleration and direction
        if(turn<=DELTA && turn>=-DELTA){                //if inside deadband, don't turn
                forwardA = power;
                forwardB = power;
        }
        else{                                           //if outside of deadband, turn
                if(power>=0){//if power is forwards then proceed as normal
                        forwardA = power+turn;                          //left motor
                        forwardB = power-turn;                          //right motor
                }
                else{//if power is reverse then invert action
                        forwardA = power-turn;                          //left motor
```

```
                        forwardB = power+turn;                          //right motor
                }

        }

        if(forwardA>0){
                forwardA = DCLOWEST+(100-DCLOWEST)*abs(forwardA)/100;
        }
        else{
                forwardA = -(DCLOWEST+(100-DCLOWEST)*abs(forwardA)/100);
        }

        if(forwardB>0){
                forwardB = DCLOWEST+(100-DCLOWEST)*abs(forwardB)/100;
        }
        else{
                forwardB = -(DCLOWEST+(100-DCLOWEST)*abs(forwardB)/100);
        }


        if(abs(forwardA)<DCLOWEST+10){
                forwardA=0;
        }
        if(abs(forwardB)<DCLOWEST+10){
                forwardB=0;
        }

        controller(forwardA, forwardB);


        //Light control
        int light = 100-(CHlight-PPMMIN)*100/(PPMMAX-PPMMIN);//a value from 0-100 depending on
toggle position
        SetLightsDC(light);
        //set the duty cycle of the lights according to channel reading
}
```

10.7.3 Controller Function

```
void controller(float powerA, float powerB)
{
//scale motor powers to match motors in forward and reverse
        if(powerA>0){//forward
                powerA = ARATIOF*powerA;
        }
        else{
                powerA = ARATIOR*powerA;
        }

        if(powerB>0){//reverse
                powerB = BRATIOF*powerB;
        }
        else{
                powerB = BRATIOR*powerB;
        }

        //limit input powers
        if(powerA<-100){
                powerA = -100;
        }
        else if(powerA>100){
                powerA = 100;
        }
        if(powerB<-100){
                powerB = -100;
        }
        else if(powerB>100){
                powerB = 100;
        }

        //calculate next pwm
        pwmA = pwmA + (powerA-pwmA)*K;
```

```
        pwmB = pwmB + (powerB-pwmB)*K;

        //sign detection for forward or reverse drive
        int AStatus = STOP;
        int BStatus = STOP;

        if(pwmA>DELTA){
                AStatus = FORWARD;
        }
        else if(pwmA<-DELTA){
                AStatus = REVERSE;
        }

        if(pwmB>DELTA){
                BStatus = FORWARD;
        }
        else if(pwmB<-DELTA){
                BStatus = REVERSE;
        }

        //set motor controls
        SetMotorDC(pwmA,pwmB);
        SetMotorDir(AStatus, BStatus);
        //HAL_Delay(D*1000);
}

// Used to set the Duty Cycles on the inputs to the H-Bridge
void SetMotorDC(float PowerA,float PowerB)
{
        float ENA = abs(PowerA);//DCLOWEST+(100-DCLOWEST)*abs(PowerA)/100;
        float ENB = abs(PowerB);//DCLOWEST+(100-DCLOWEST)*abs(PowerB)/100;

        //set pwm to calculated value, setting it as positive with absolute
        __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_3, (int)ENA);
        __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_2, (int)ENB);
}

// Used to enable both motors
void SetMotorDir(int DirA, int DirB)//use FORWARD, REVERSE, STOP
{
        if(DirA == 1){//motor A forward
                HAL_GPIO_WritePin(GPIOB, IN1_Pin, GPIO_PIN_SET);
                HAL_GPIO_WritePin(GPIOB, IN2_Pin, GPIO_PIN_RESET);

        }
        else if(DirA == -1){//motor A reverse
                HAL_GPIO_WritePin(GPIOB, IN1_Pin, GPIO_PIN_RESET);
                HAL_GPIO_WritePin(GPIOB, IN2_Pin, GPIO_PIN_SET);

        }
        else{//motor A STOP
                HAL_GPIO_WritePin(GPIOB, IN1_Pin, GPIO_PIN_SET);
                HAL_GPIO_WritePin(GPIOB, IN2_Pin, GPIO_PIN_SET);
        }

        if(DirB == 1){//motor B forward
                HAL_GPIO_WritePin(GPIOB, IN3_Pin, GPIO_PIN_RESET);
                HAL_GPIO_WritePin(GPIOB, IN4_Pin, GPIO_PIN_SET);
        }
        else if(DirB == -1){//motor B reverse
                HAL_GPIO_WritePin(GPIOB, IN3_Pin, GPIO_PIN_SET);
                HAL_GPIO_WritePin(GPIOB, IN4_Pin, GPIO_PIN_RESET);
        }
        else{//motor B STOP
                HAL_GPIO_WritePin(GPIOB, IN3_Pin, GPIO_PIN_SET);
                HAL_GPIO_WritePin(GPIOB, IN4_Pin, GPIO_PIN_SET);
        }
}
```

10.7.4 SetLightsDC Function

```
void SetLightsDC(int LIGHTS)
{
```

```
        //set light between 3 levels of brightness only
        if(LIGHTS>PPMMIDLIGHT){ //maximum brightness
                __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_4, MAXLIGHT);
        }
        else if(LIGHTS>PPMMINLIGHT){ //medium brightness
                __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_4, MIDLIGHT);
        }
        else{//minimum brightness
                __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_4, MINLIGHT);
        }
}
```

10.8 TESTING PROCEDURE TO FIX MOTOR NOISE PROBLEM

The testing procedure used to try and debug the weak-point in the circuit that the motor noise was effecting is described below, as well as the actions taken and results. Unfortunately the problem was not solved due to time constraints, but suggestions are given on how to continue the debugging process.

Table 15: Summary of testing procedure to solve motor noise problem

Test Procedure	Results	Theory as to why	Action taken	Results of action
Measure power supply with and without motors	Some fluctuations with motors turning on	Brownout on supply causing unknown problems	First increased decoupling (on power at h-bridge)	Noise on power lines removed
			Changed 5V supply to a fixed regulator to ensure a stable supply for the H-bridge	No change
			Used larger power supply with 6A current	No longer drooping voltage, but problem persists
			Use larger wires to motors	No change
Use test software and observe response and signals from microcontroller	Behaving as expected, except that light flashed occasionaly	Not clear why (found later to be that the board was resetting)		
Trace motor commands from microcontroller to h-bridge	All stable, except when motors turned on. Then enable pins change DC, lights send on off signal	Noise from motors are effecting receiver signals, causing false interrupt triggers	Decouple receiver power and signal	Very small noise, thought not enough to trigger events. Problem persists. Can't reduce noise any further
			Add noise resilience to code	Slight improvement in problem, but still persistent
			Measure frequency of noise spikes on receiver to see if it correlates to another signal.	Seemingly random spikes, unpredictable
			Wire up H-Bridge module instead to check if the h-bridge built has faults	Problem persists, no change.
Test if problem caused by LED driver circuit			Replaced LED driver circuit with upgraded design	LED driver more efficient as it turns fully off,

				but no difference to problem
Test all forms of input to microcontroller	Found noise on reset pin, causing the board to reset	Noise picked up from motors causing spikes, resetting microcontroller repeatedly	Added pull up resistor and decoupling capacitors to reset pin, and all other hanging pins.	Huge improvement on problem. Enable pins now mostly stable, the input pins are now haywire
Confirm that resets are no longer happening,			Turn off lights code. The lights turn on by default on start up, so if it flashes it is from resets	There were no flashes, therefore the problem is no longer from resets.
Possible future test procedures				
Check input pins from debugger, or lines around USB area (the micro sends commands when the USB cable is finished debugging the micro)	Not enough time for this. Will be done for future development.	The debugger is causing issues by trying to communicate.	Remove debugger to check if it is communicating with the micro.	
Check if microcontroller is somehow faulty by using another one.				
Check signals from light, or the power draw as a result, as the motors randomly jump when the lights are on				
Toggle a test pin when an interrupt occurs to check if there is noise causing interrupts				

As seen in the summary above, initially there was no concept as to why the signals could be changing so much. As actions and tests were done, the understanding came that the microcontroller was causing the fluctuations. Using the following expected signals truth table, the signals were tested from the microcontroller with and without motors plugged in.

Table 16: Microcontroller signals truth table

Channels			**Outputs**						
Power	**Turn**	**Lights**	ENA (PWM)	IN1 (1 or 0)	IN2 (1 or 0)	IN3 (1 or 0)	IN4 (1 or 0)	ENB (PWM)	Lights (PWM)
0	**0**	**0**	0	1	1	1	1	0	0
100 (Forward)	**0**	**X**	100	1	0	0	1	100	X

-100 (Reverse)	**0**	**X**	100	0	1	1	0	100	X
0	**-100 (left)**	**X**	100	0	1	0	1	100	X
0	**100 (right)**	**X**	100	1	0	1	0	100	X
100	**-100**	**X**	0	1	1	0	1	100	X
100	**100**	**X**	100	1	0	1	1	0	X
-100	**-100**	**X**	0	1	1	1	0	100	X
-100	**100**	**X**	100	0	1	1	1	0	X
X	**X**	**L≥75**	X	X	X	X	X	X	100
X	**X**	**75>L≥25**	X	X	X	X	X	X	70
X	**X**	**25>L≥0**	X	X	X	X	X	X	50

The signals were all as expected without the motors plugged in, but they were all not as expected when the motors were plugged in. However, it was unlikely (not impossible) that it was the code, as the test software worked when motors were plugged in, driving the motors through various phases. Therefore, it was stable for internal commands, but unstable for external commands. The signals seen are shown below.

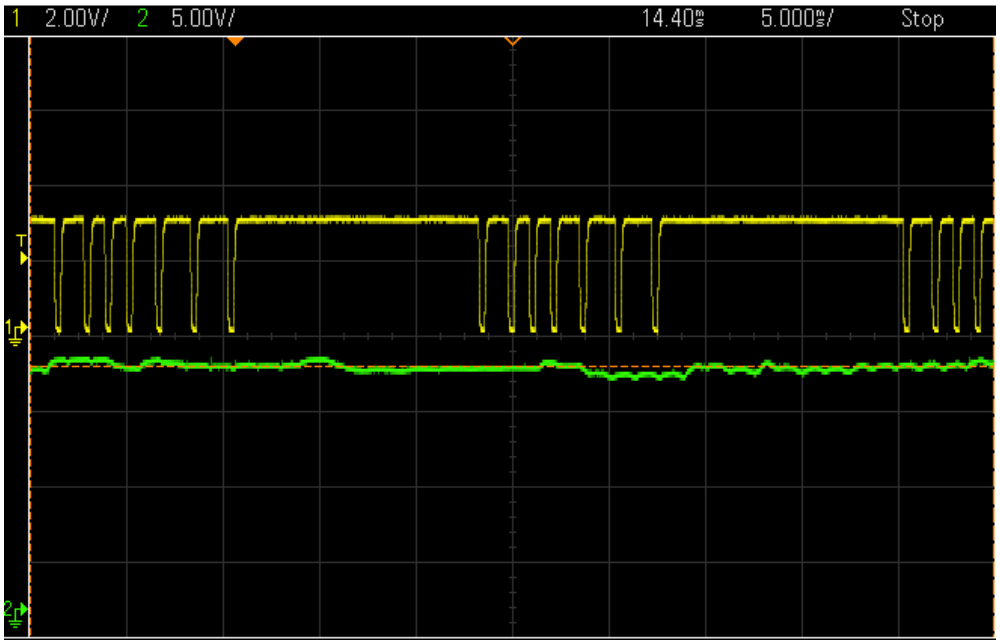


Figure 64: PPM messages and battery voltage when motors are running

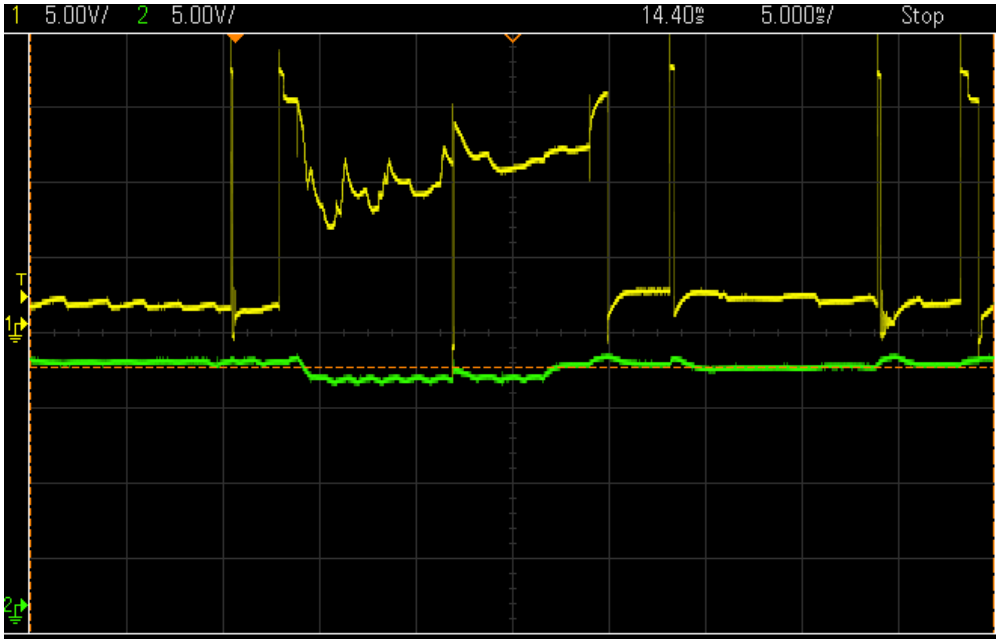


Figure 65: voltage on batteries and battery voltage when motors are running

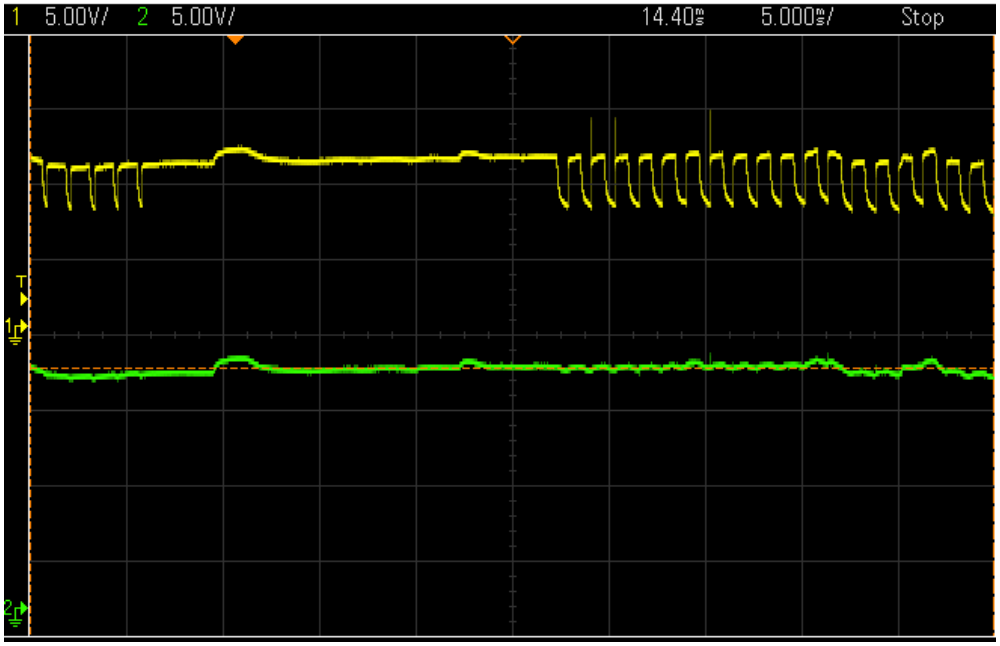


Figure 66: Signal to lights and battery voltage when motors are running

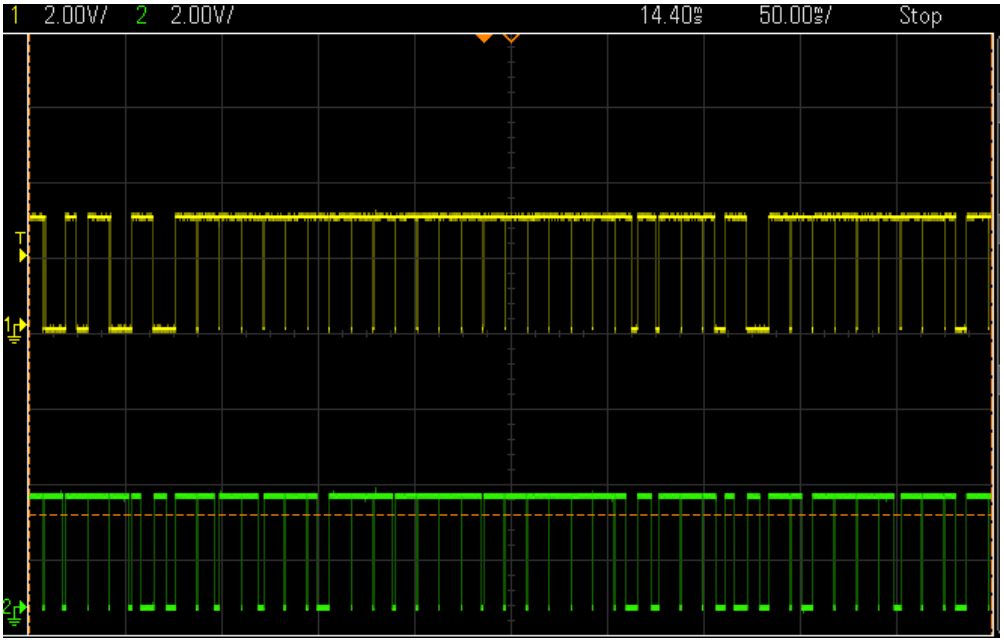


Figure 67: Enable A and B signals and battery voltage when motors are running

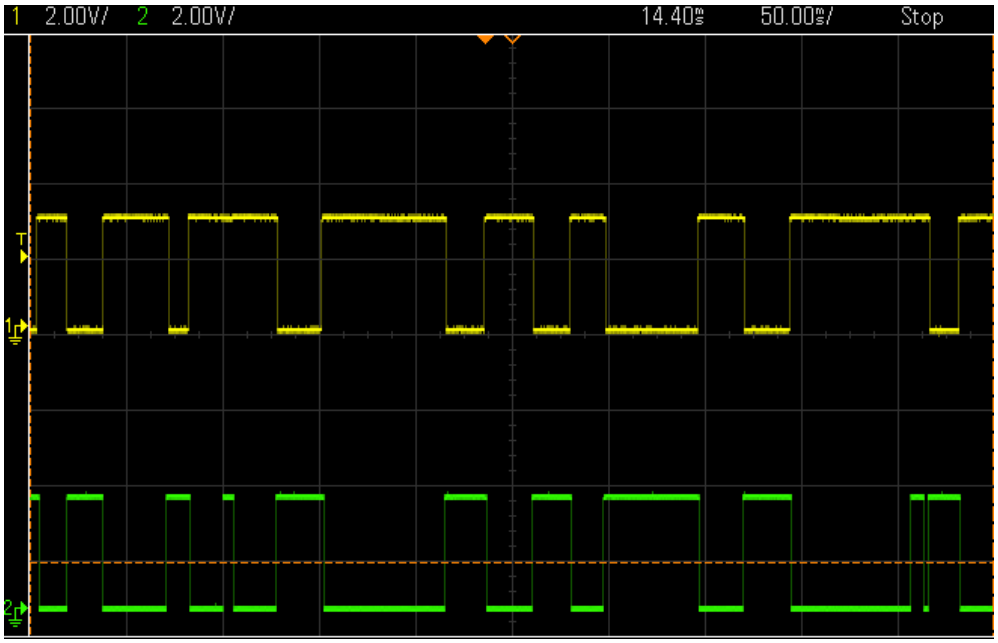


Figure 68: directional inputs and battery voltage when motors are running

After reviewing these signals, it led to the understanding that it was the inputs to the microcontroller that were causing the problems. It was initially believed that it was noise on the receiver line that was causing false interrupts and confusing the software into thinking there was a channel when there wasn't. Adding decoupling capacitors to the receiver and software conditional statements to the interrupt line improved the situation but only slightly.

Only after analysing the signals on each input to the microcontroller did true results begin to occur. The various inputs to the microcontroller are:

- Power pins
- Ground pins
- Receiver PPM signal pin
- Reset pin

- Boot0 pin
- External oscillator pins (input and output)
- T_SWCLK and T_SWDIO for debugger

The power, ground and receiver pins were all decoupled to reduce noise impact, and conditional statements were added in the software to add resilience to false triggering of interrupts for the PPM signal. This all had a small impact but not much.

The reset pin was a big culprit of the problem. The noise on the pin caused continual resetting of the microcontroller. By adding a pull up resistor and decoupling capacitor this stopped occurring. This drastically improved the results, but the problem persisted, although in a different form. Initially it was the enable pins that were fluctuating. Now they only sometimes changed, but it was the directional inputs that were fluctuating.

The Boot0 pin also had noise, for which a decoupling capacitor was added. This cleaned up the noise but had no result on the problem.

Due to time constraints the full debugging of this problem was not possible, but the course of action that would have been taken would have been to check every pin on the board and ground and decouple any that aren't being used. This is just as a precaution. Additionally it would be wise to check the signals on the debugger to make sure that it isn't randomly trying to make a connection with the microcontroller because of noise.

Because all of the testing has been done under the unknown condition of continual resets, it would be wise to repeat all the tests on the signal to the components. The questions that should be asked are: are the signals expected, and if not what is the expected signal? Does it behave any better without the motors connected? It would also be wise to confirm that the reset pin is definitely not resetting still, and if so that it should be decoupled more.

10.8.1 Summary of Motor Noise Debugging

Initially it was not understood what was causing the robot to jitter and be unreliable. After testing it was found that the motor noise was causing the inputs to the microcontroller to trigger undesired results. This was always worse when there was a larger load on the motors, presumably from the higher current draw. The first problem that was solved was that the reset pin was being triggered constantly by noise. After this was solved, there was no time to continue debugging.

The various actions taken to compensate for the noise and current draw are:

- Decoupling on the motors was increased to several thousand μF
- Pull up/ pull down resistors as appropriate on the various input pins of the microcontroller, especially the reset pin. Decoupling capacitors were also included on each of these pins.

These solutions should be included in future designs.

Although the Veroboard was initially considered a waste of time, the lessons learned with it were invaluable. If the design was made on a PCB immediately there would have been less chance to debug the issues that arose. A breadboard set up would not have worked either due to the nature the robot being designed. A more permanent circuit board was needed than a breadboard to resist the driving motion, but a PCB would have been too restricting on the debugging process.

In addition to always checking power first, it is important to analyse for noise on each pin of a microcontroller. Testing for noise where you expect there to be none is often useful in finding unknown problems.

10.8.2 Solving the Motor Noise Problem

The only problem that could not be solved was the noise caused by the motors. When the motors were plugged in, the signals from the microcontroller started going haywire. The debugging process described in Section 5.4 showed that the microcontroller was performing as expected and that the chance of the problem being on the software side was slim. The noise from the motors was corrupting the signal from the receiver before it entered the microcontroller, and thus the software was interpreting the wrong data. This would best be fixed by cleaning up the noise.

Possible solutions to this problem are a complete redesign, or an extensive debugging process to find where the noise is entering the system.

- Redesign

The redesign option would be to give the motors their own dedicated power supply, with the only inputs to that circuit being the commands from the microcontroller. The body of the robot should be lengthened so that the microcontroller can be placed further away from the motors, with plenty of decoupling capacitors shielding it from the noise. This should go a long way to cleaning up the problem and is in fact a good redesign anyway as it follows better standards. The batteries can be closer to the motors as well which would enable easier climbing too because of a closer centre of mass.

- Extensive debugging

As discussed in Section 5.4, various steps are recommended for debugging the problem to find a solution. These are to:

a) test the pins from the debugger to make sure it is not trying to communicate with the micro controller
b) Replace the microcontroller in case it is faulty as a result of continual resetting (this was an issue that was fixed)
c) Track signals from light. When this was on it started causing jitters too. It is possibly not the motors, but just a current draw problem
d) Toggle a test pin when an interrupt occurs to check if noise is causing false interrupts

These are merely suggested ways forward, and would hopefully shed more light on what the potential solution is.

10.9 Equalised Control of Motors

It was seen in Design Phase 1 and 2 that some form of control was needed to equalise each motor's strength with the other. This was in order to better control the motors and prevent one becoming dominant and flipping the robot onto its side. Refer to Figure 26 in Phase 1 to see an example of this happening. Two methods are proposed below, however as they are not in the scope of this project they are not investigated further.

10.9.1 Current Sensing Control of Motors

Using the inbuilt current sensing pins of the L298N, the current can be read into the microcontroller. A controller can then be designed using this feedback loop to ensure the motors achieve the desired motion specified by the user. This would form an assisted open loop system as the user is still in control of the motors.

Space was left for a resistor to be placed in the H-bridge circuit in the Veroboard design to measure the current through each of the motors. The circuit diagram for this is shown in Figure 69.

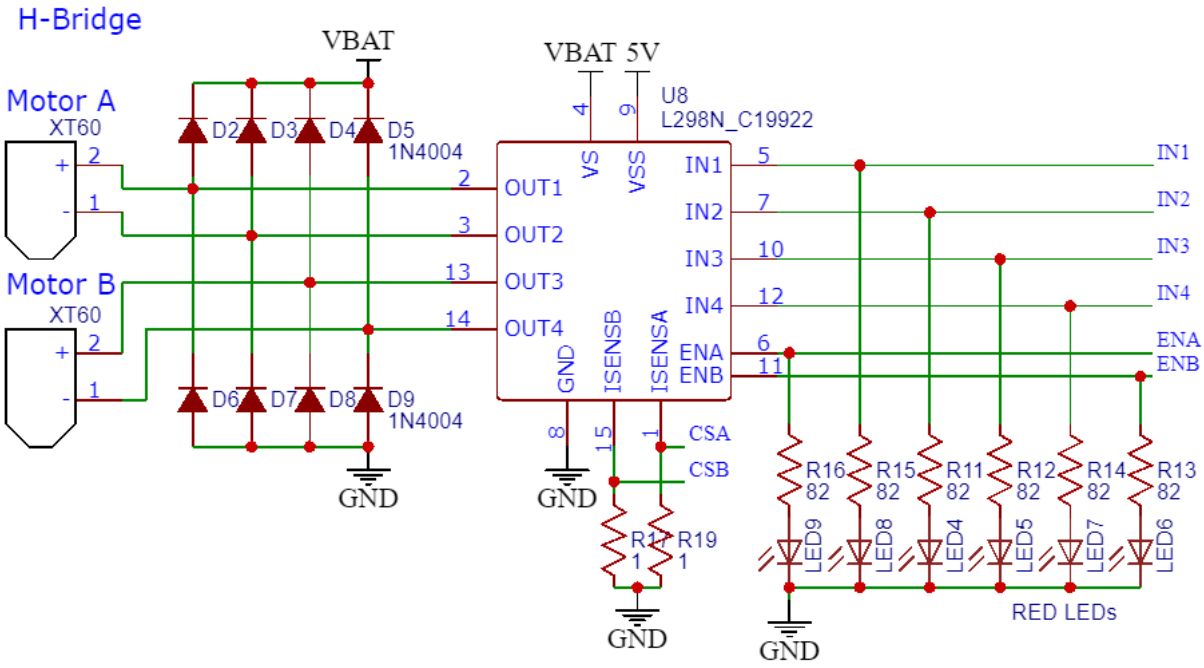


Figure 69: Current sensing implementation in H-Bridge circuit

The outputs CSA (Current Sense A) and CSB go to pins PA7 and PA6 respectively. These were already marked in the original microcontroller circuit in preparation for this possible modification.

With these inputs set up to read analogue inputs using the onboard ADC, the voltage level tells the microcontroller what the current in each motor is at any given time. These resistors must be very carefully matched for this purpose.

From this point onwards, this is a control theory problem. The inputs are the voltage levels from the resistors the microcontroller what the current in each motor is. The outputs are the PWM duty cycles to each motor. There is already a controller for controlling the motor PWM using a setpoint, and it works with each motor individually to achieve differential drive. It should be possible to implement this equalising controller by piggy-backing on the system already in place.

As stated before, this was not attempted as it is outside the scope of this project.

10.9.2 Accelerometer Control of Motors

The current sensing control may not be the best solution. Using an accelerometer may be useful for equalising control on the motors, as it can tell the microcontroller exactly what direction the robot is accelerating in. Using similar control concepts as discussed above, this can be used to perform differential driving on the two motors to equalise the motors according to the orientation of the robot.

10.9.3 Summary of Equalised Control of Motors

These controller features could be implemented separately, or be used together to ensure control action is correct. This would require extensive system identification as well as control theory. This was not investigated further as it was outside the scope of this project, and could in fact be an entire project on its own.

11 BUILDER'S GUIDE

Using this builder's guide, anyone can reproduce the results of this project by building their own Di-Wheel robot. There are laser cutting sheets which must be cut as explained below, as well as shafts that must be manufactured.

11.1 SHAFTS NEEDED FOR ASSEMBLY

The shafts that are needed for the assembly are cut from 6mm aluminium rod, as well as an M5 hexagonal head bolt. Their dimensions are shown at the end of this document, before the laser cutting forms.

11.2 LASER CUTTING

At the end of this document are the laser cutting forms. The first four pages are 3mm hardboard, the rest are 3mm Perspex.

11.3 CIRCUIT BOARD DESIGNS

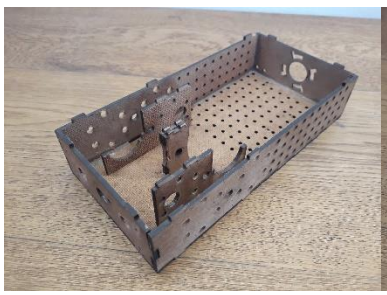
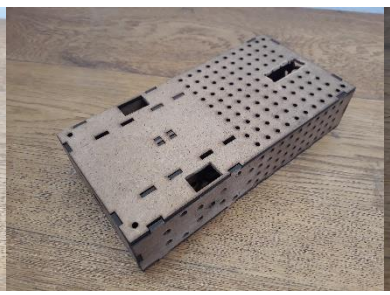
At the end of this document are the designs for the manufacture of the circuit boards, these must be manufactured before continuing.

11.4 BILL OF MATERIALS

Refer to the bill of materials in the main report for the various components to purchase.

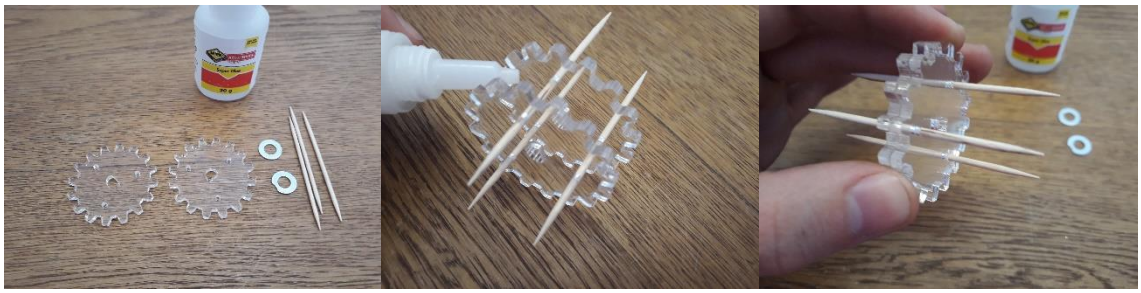
11.5 STEP-BY-STEP ASSEMBLY INSTRUCTIONS

After the lazer cutting is complete, assemble the body and tail as seen below. The hardboard is glued together with wood glue, and the Perspex is glued together with super glue.

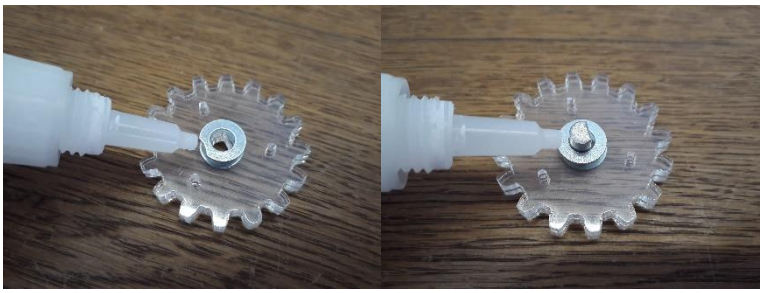




The gears that are cut out of Perspex must be aligned together with toothpicks, and glued together with superglue. You must ensure that the superglue covers as much of the mating surfaces of the gears as possible. Remove the toothpicks before the glue sets so that they are not glued in place.

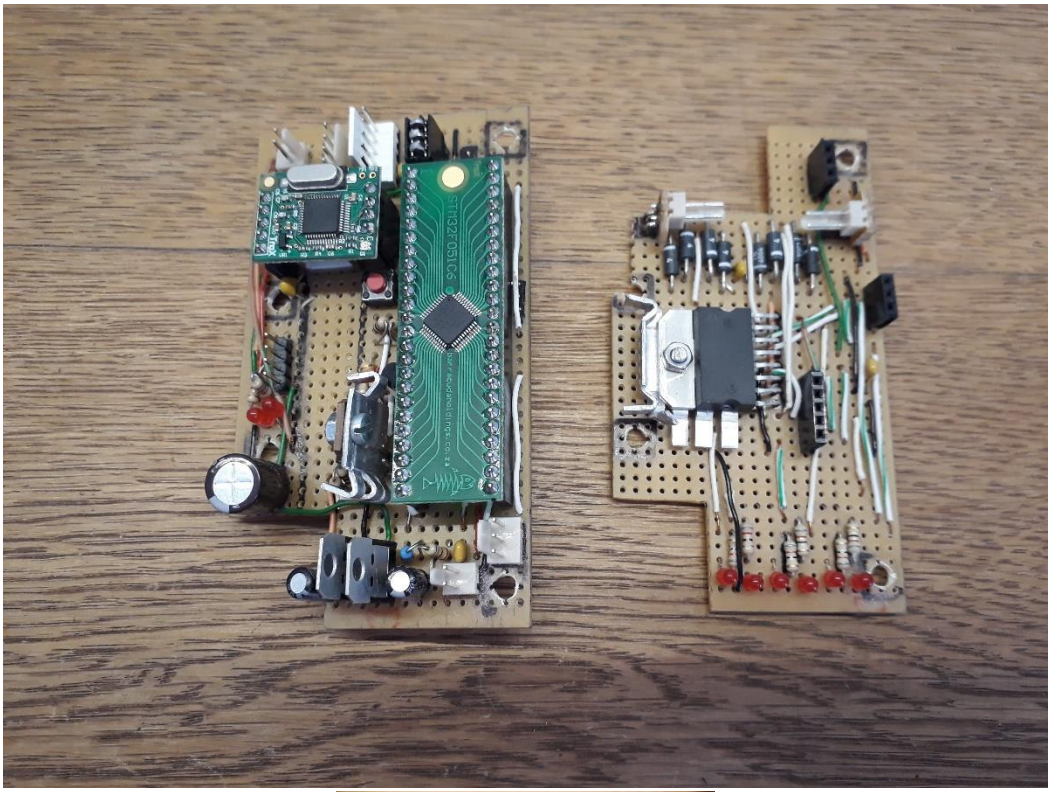
All the Di-Wheel gears (these have smaller teeth) are only two layers thick. All the body gears (these have larger teeth) are four layers thick. You will find that the toothpicks can’t be removed in time for the body gears and so the extending bits should be cut off and sanded down so that the gear is completely smooth.

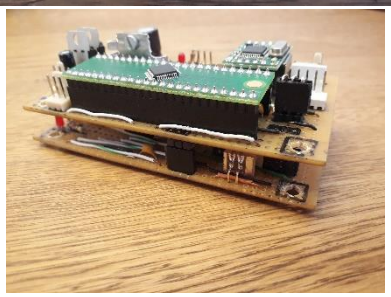


All the Di-Wheel gears require a washer to be glued around the shaft hole on either side. The Idler gear requires the short shaft to be glued in as well (this is the middle sized, small tooth gear)

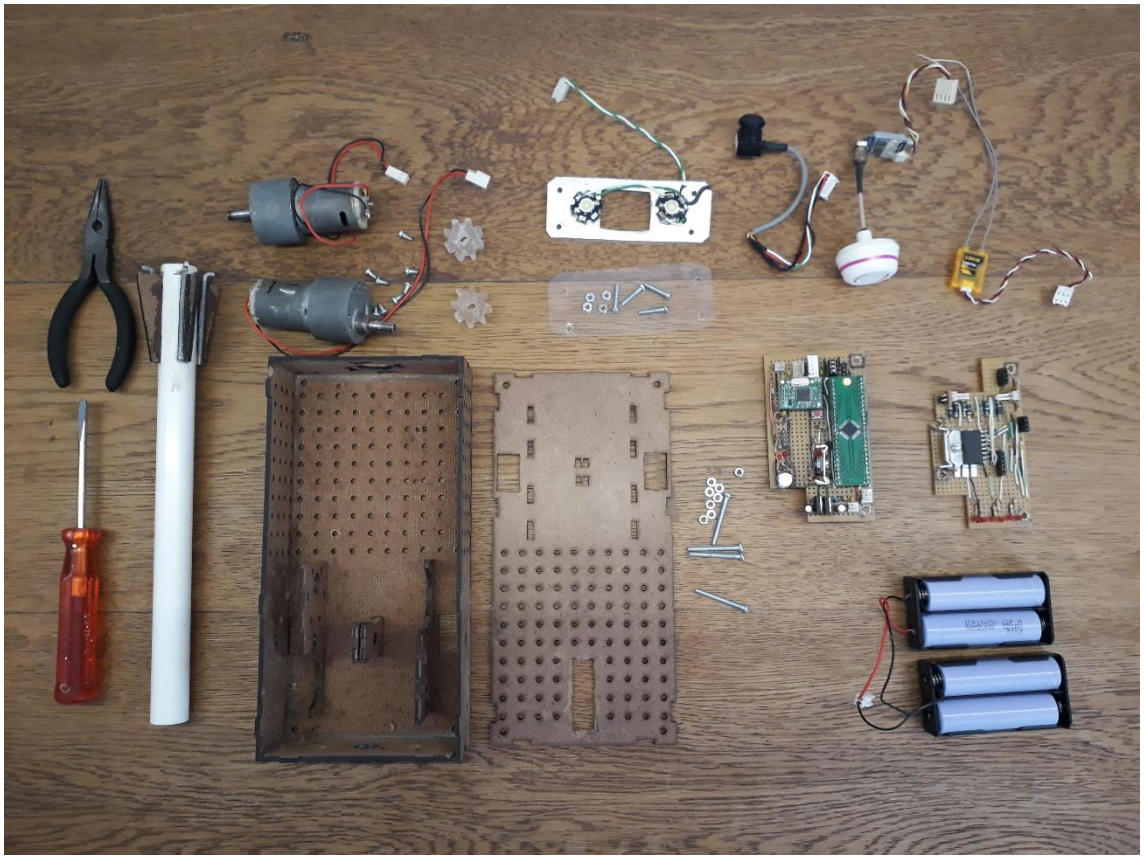


Once the Veroboards have been assembled, make sure that they fit on top of each other, and that the holes line up.

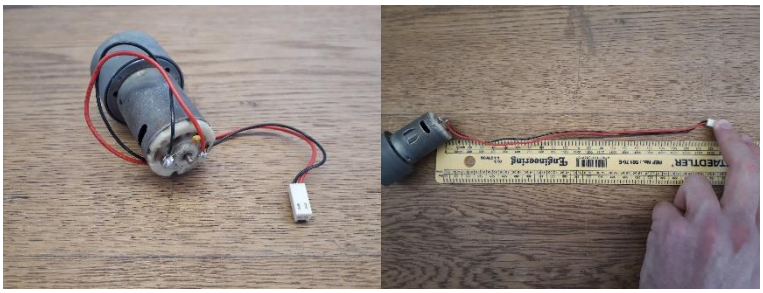




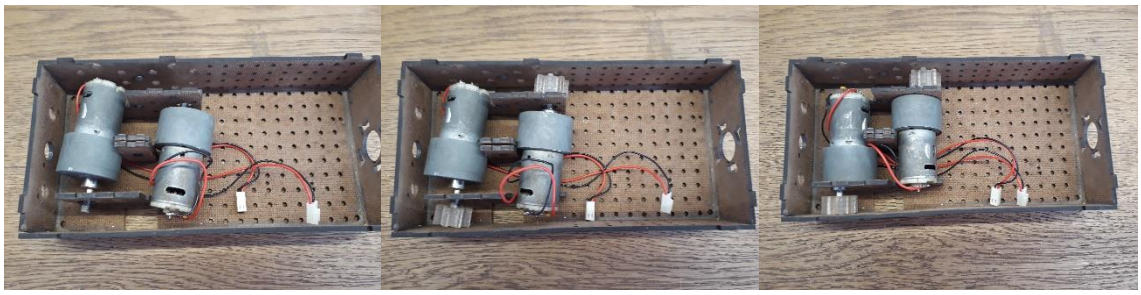
Shown below are all the components required for the body assembly. Included are (from left to right): the tail, motors, power LEDs, camera, 5.8GHz transmitter, 2.4GHz receiver, body and top, M3x30 fasteners, the electronic boards, and the batteries. Make sure that the power LEDs are glued to their bracket, with the 2-pin molex connector ready. Also make sure that the batteries are connected in series, with enough cable length that they can sit on opposite sides of the body. They require a 2-pin molex connector.



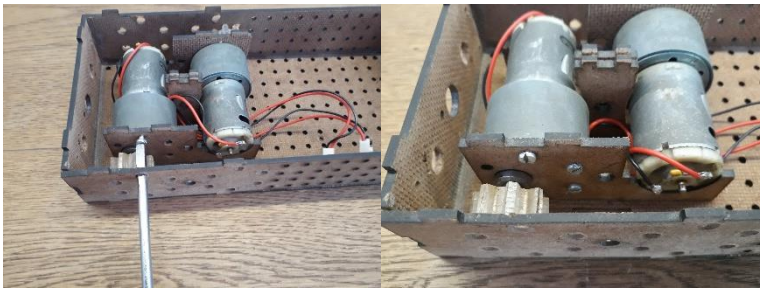
Wire up the motors, each with a cable of about 20cm long. A 2 pin female molex connector should be on the end as shown, as well as a suitable capacitor between the motor connectors. It is very important that the tabs that the wires are soldered onto are folded as flat as possible so that



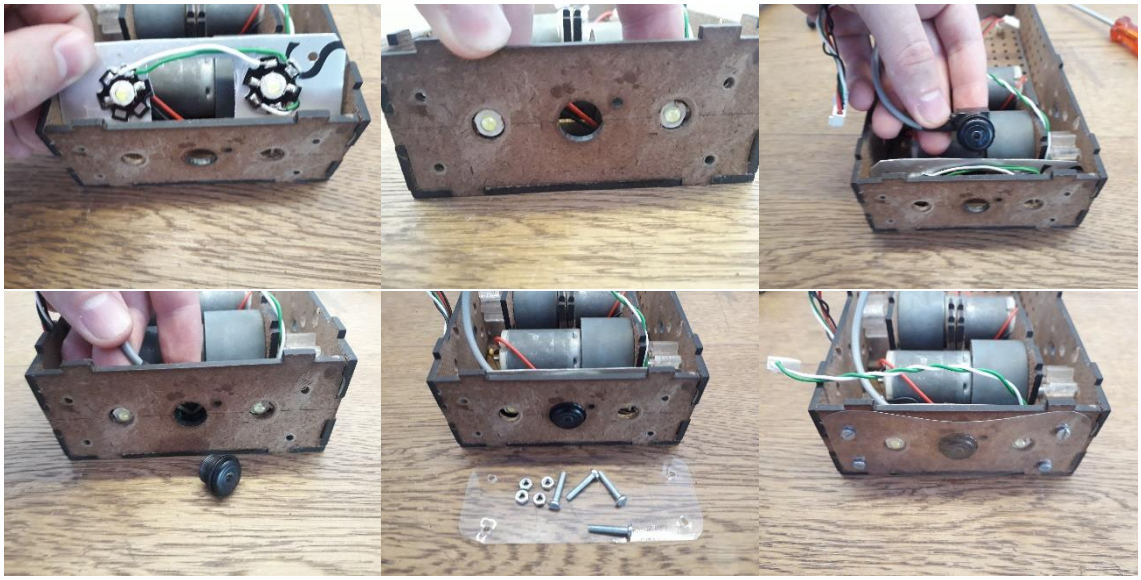
Place the motors through the holes in the supporting brackets, align the gears and finish locating the motors. Be careful to have the wires travelling underneath the motors as shown, away from where the gears will be.



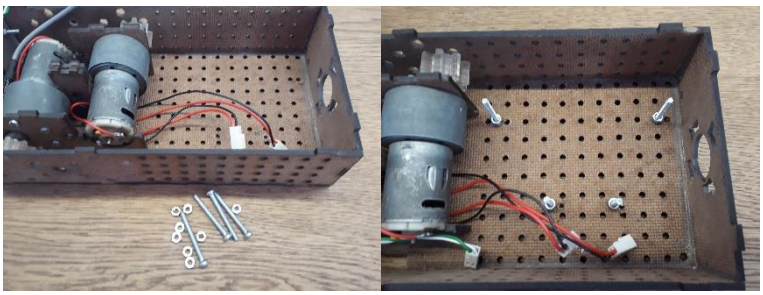
Fasten the motors in place with M3x6 fasteners.



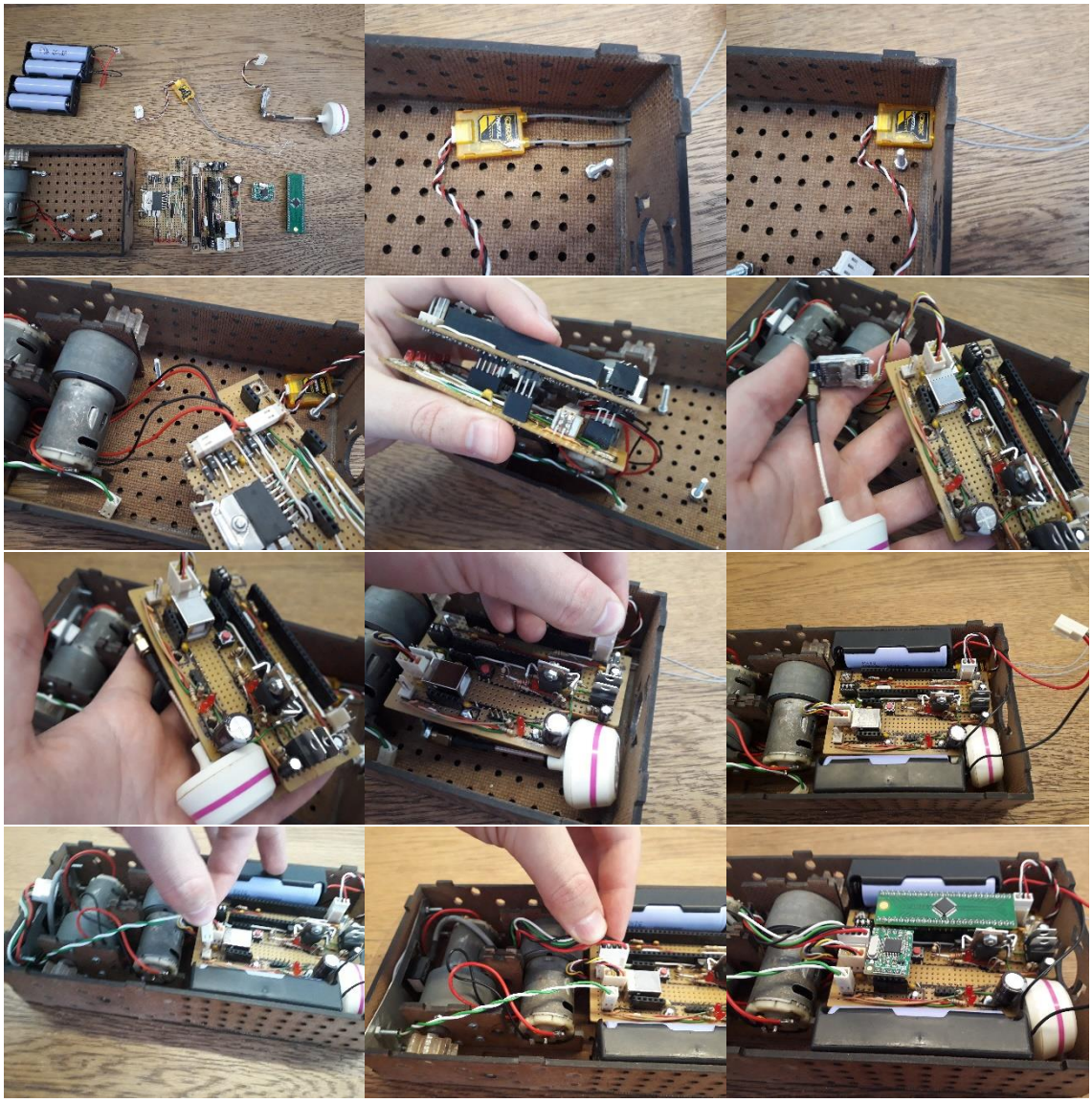
Locate the power LEDs, and screw the camera in place through the LED bracket's hole. Once done, fasten the protective plastic shield to the front of the body, using the same fasteners to secure the LED bracket. This uses M3x16 fasteners.



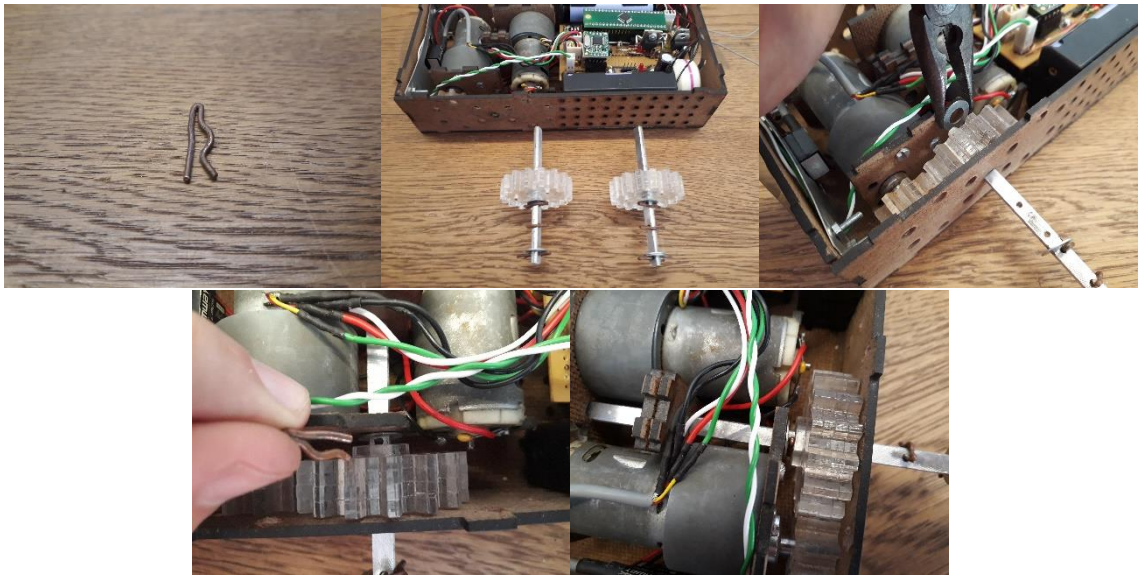
Place the M3x30 fasteners, with two nuts per fastener, in the locations shown. These are going to hold the circuit board in position.



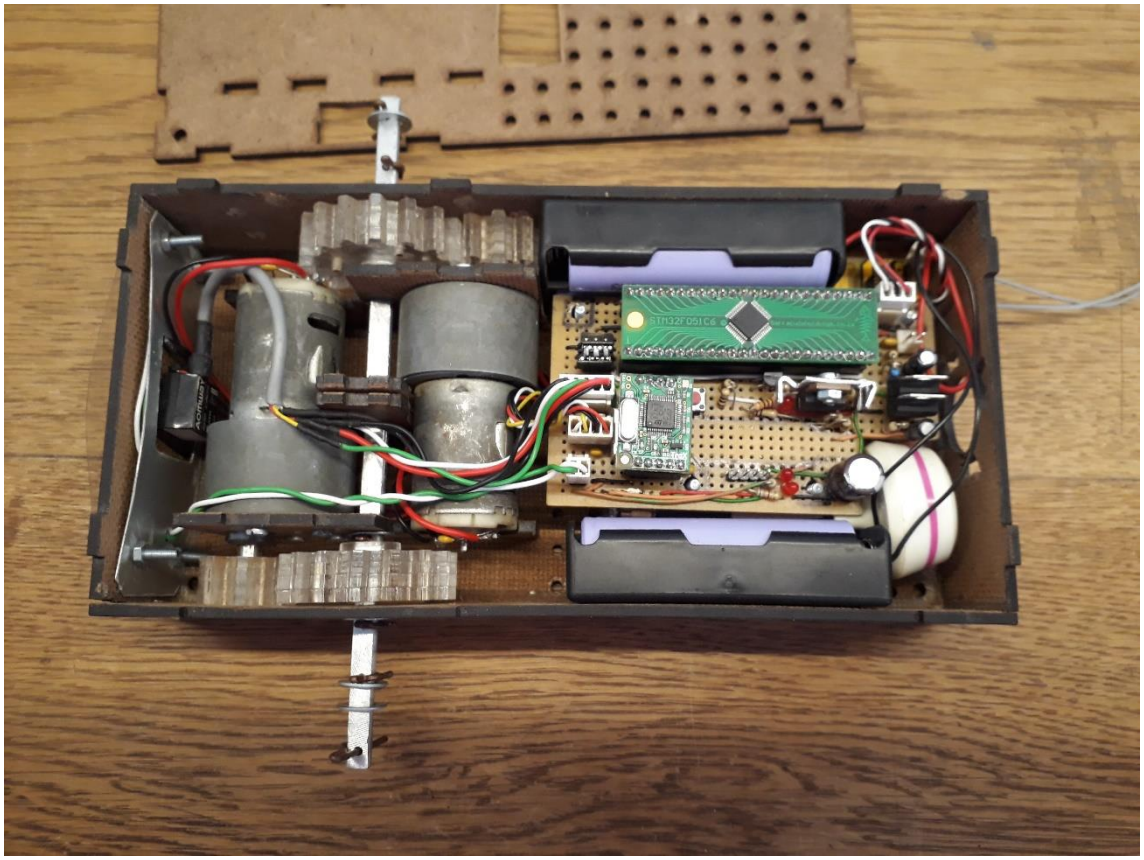
Place the receiver in position, plug in the motor cables to the lower circuit board, and secure the top circuit board. Plug in the transmitter, make sure it is in the position shown and slide the circuit board onto the fasteners. Plug in the receiver. Slide in the batteries and don't plug them in, this must be done last. Plug in the LED cable, and the camera cable. The microcontroller and debugger can be plugged in next

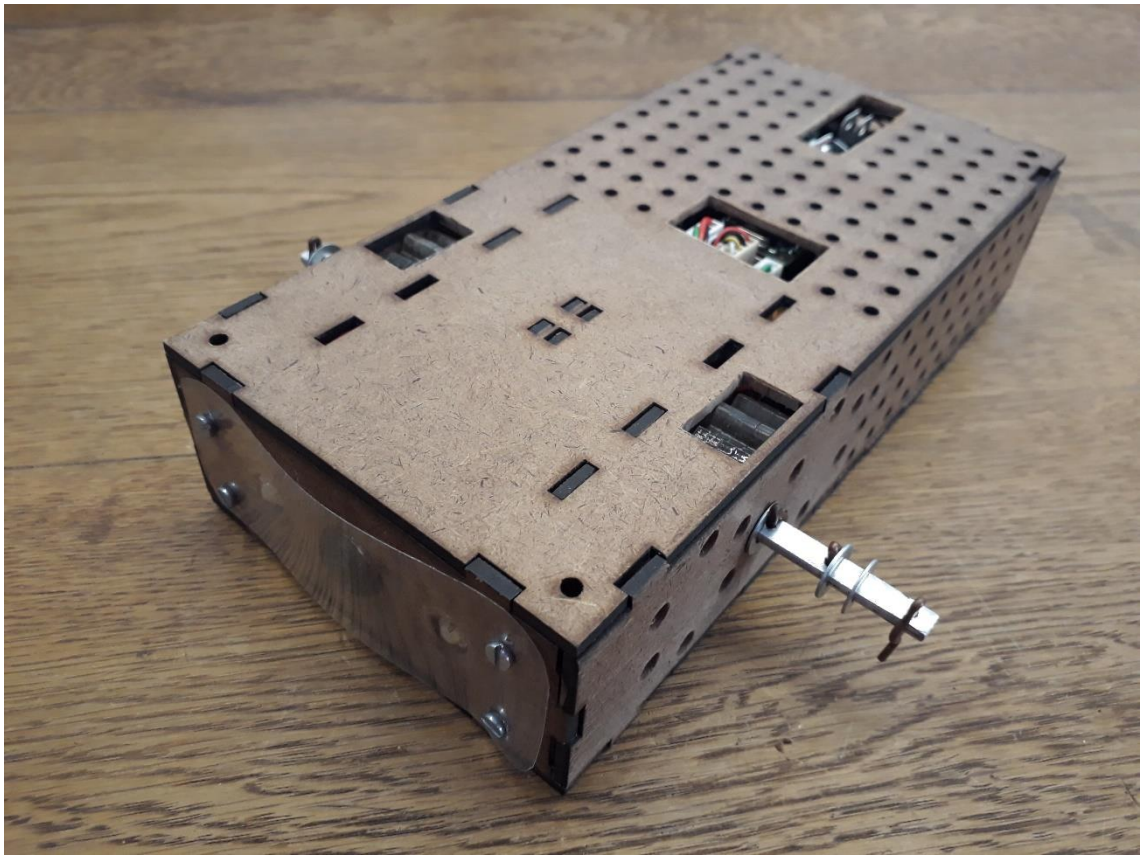


The motor shafts can now be placed in. Hitch pins were manufactured using simple 2mm copper wire, but they can be purchased as well. Remove the inner clip and washer and gear. Slide the shaft through first the wall, then the gear then the washer. Line up the hole for the clip and fasten the clip. Make sure that the cables are out of the way of the gears, and won’t get in the way of the lid closing.

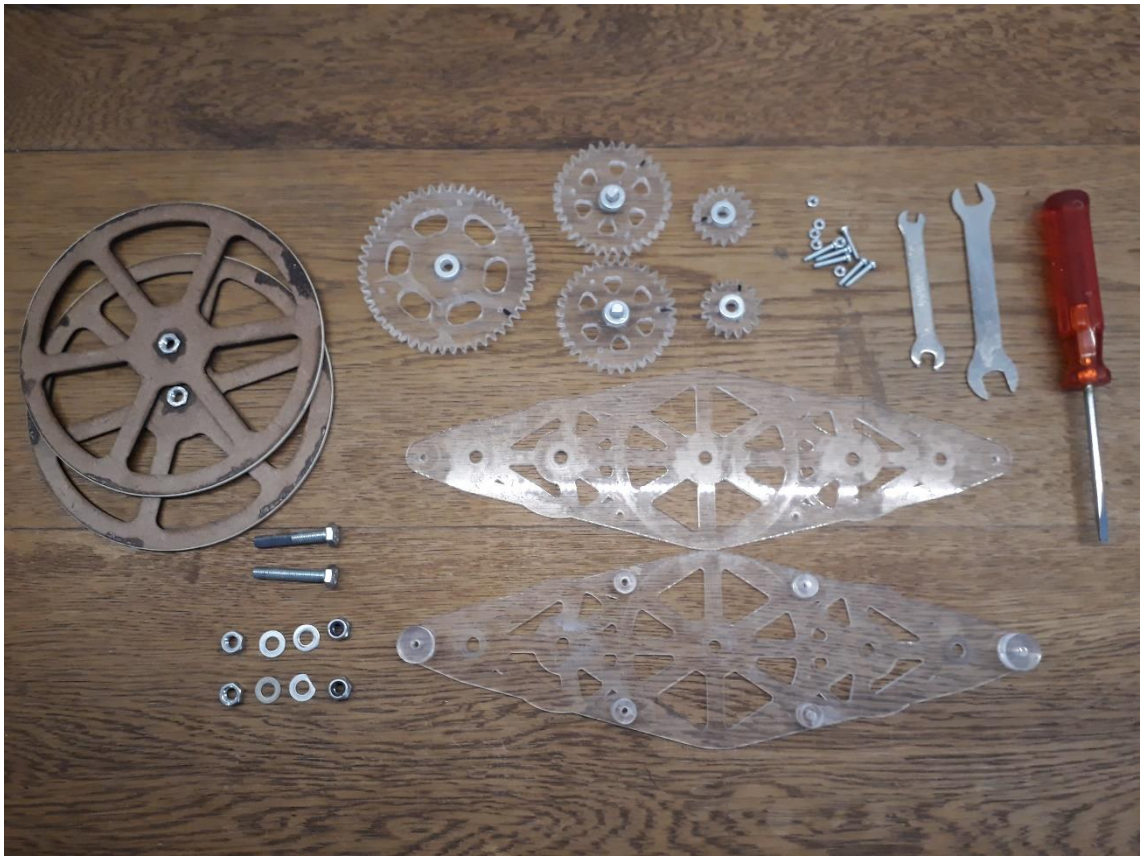


The body is now ready. Check that everything looks the same as shown below.





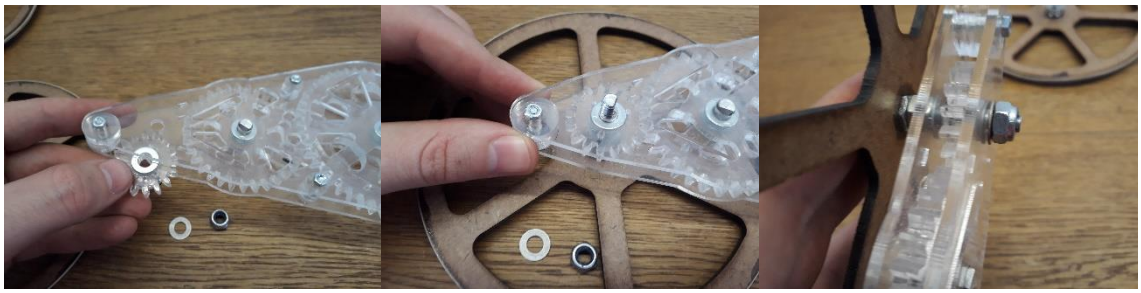
The Di-Wheel components for assembling one Di-Wheel are shown below. Please make sure that the spacers are glued in place on the carrier before beginning, as this makes everything easier. There needs to be 3 washers per fastener hole, as seen below.



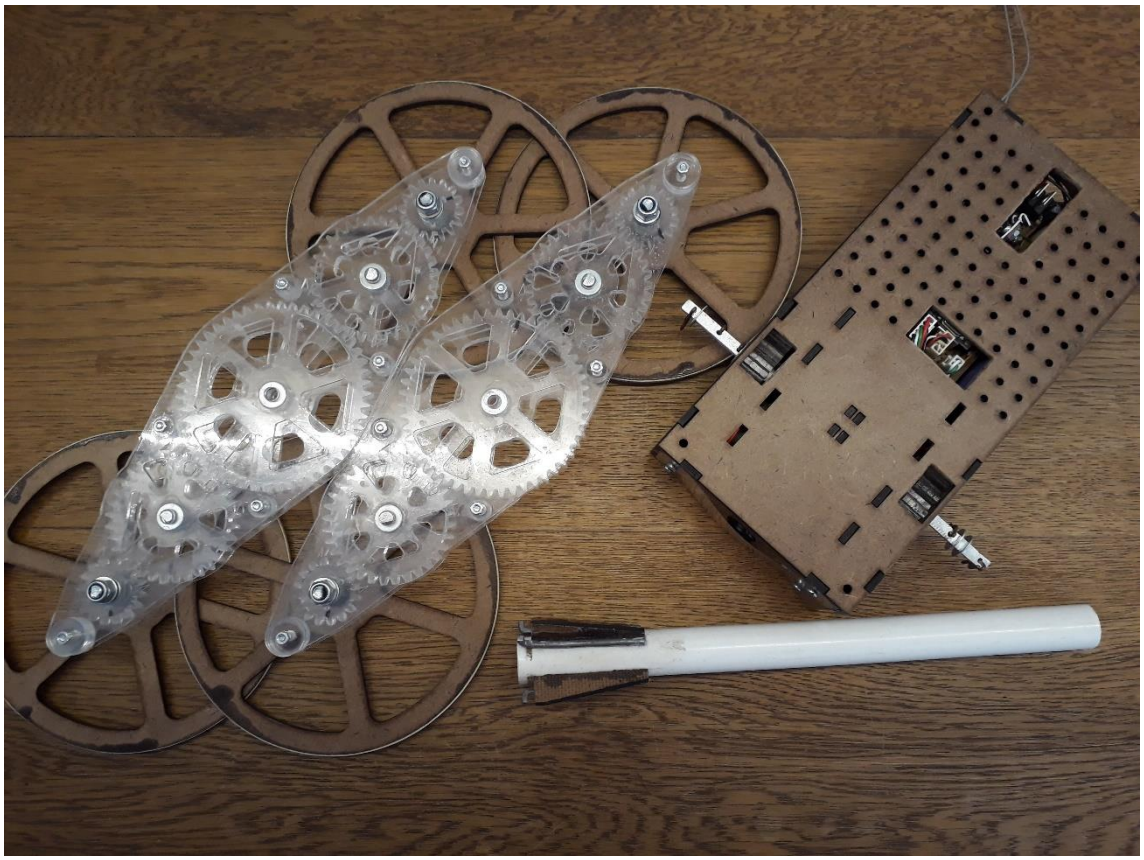
The wheel must have a rubber band glued on the side, as well as a nut glued into the centre hole. The fastener must be tightened on until the nut locks with the fastener head, then place the gear, two washers and a locknut on. Repeat this for each wheel.

The sun and idler gears must be placed on to the carrier, and then the top layer of the carrier can be fastened in place with M3x16 fasteners.

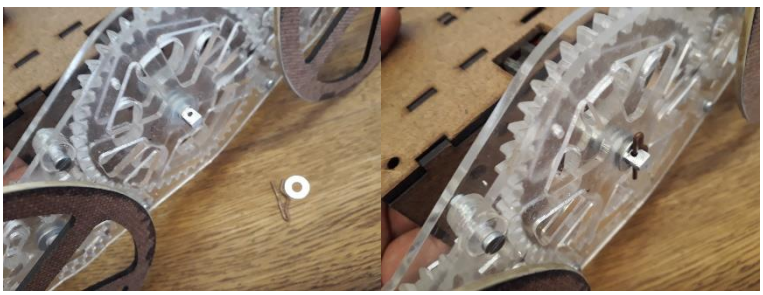
The Di-Wheel is now ready for the wheels to be fastened on. Remove the planet gear from the wheel assembly and slide it into the space shown. Before sliding the wheel shaft through, ensure there is a washer in place on the shaft. After sliding the wheel shaft through the carrier and planet gear, fasten it on the other side with another washer and locknut.



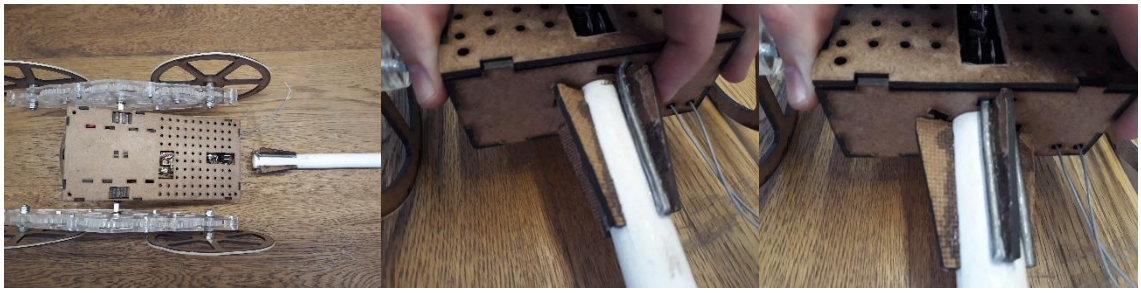
The Di-Wheel robot is now ready to be fully assembled.



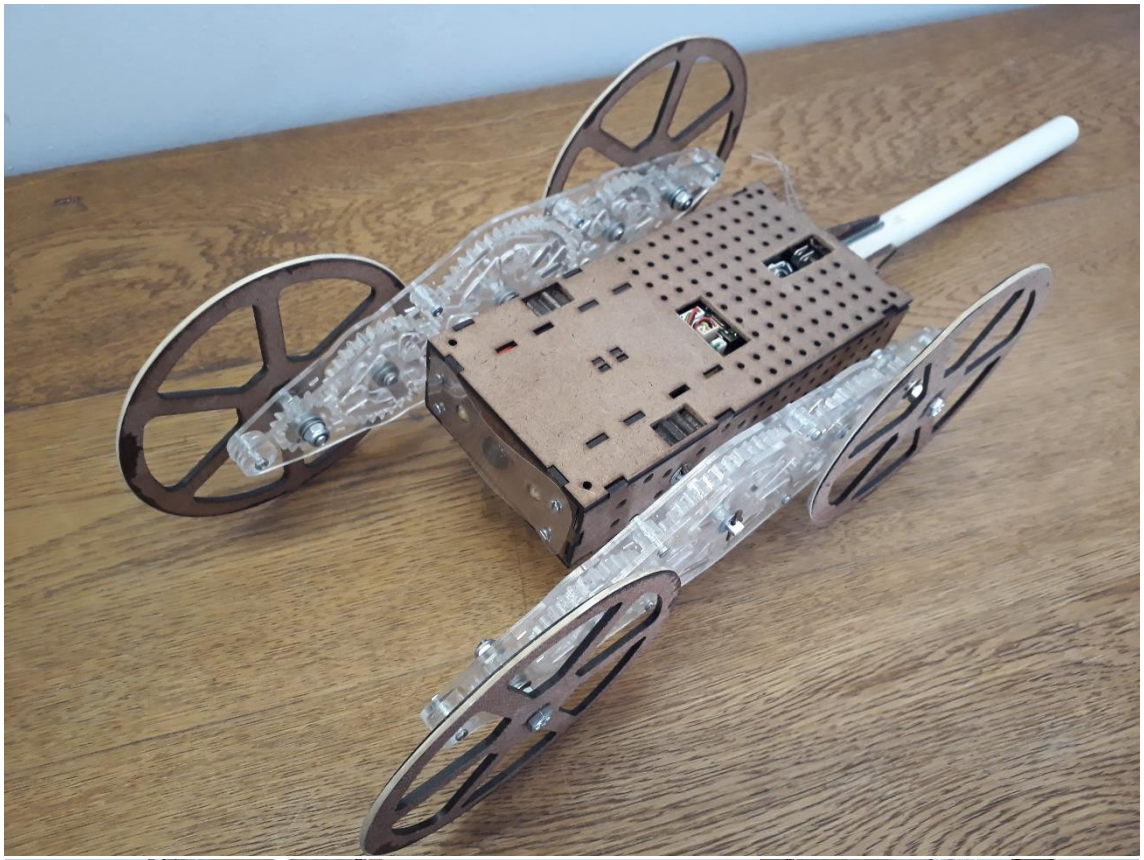
Remove the clip from the main shaft, and one washer. Slide the Di-Wheel in place and replace the washer and clip. Repeat this for both Di-Wheels.



Locate the tail ribs and rotate into place to ensure the tail is secure.



The Di-Wheel robot is now fully assembled and ready to go once the software has been uploaded. This is best done using Atollic TrueStudio.



11.6 VEROBOARD, TECHNICAL DRAWINGS AND LASER CUTTING TEMPLATES

Shown below are the Veroboard designs of the circuit boards (designed with DIYLC). Also shown is the layout of the components and how they fit around the circuit board. Ensure that the holes shown in the corners of the Veroboard can fit an M3 screw. Refer to the Bill of Materials for all the components needed.

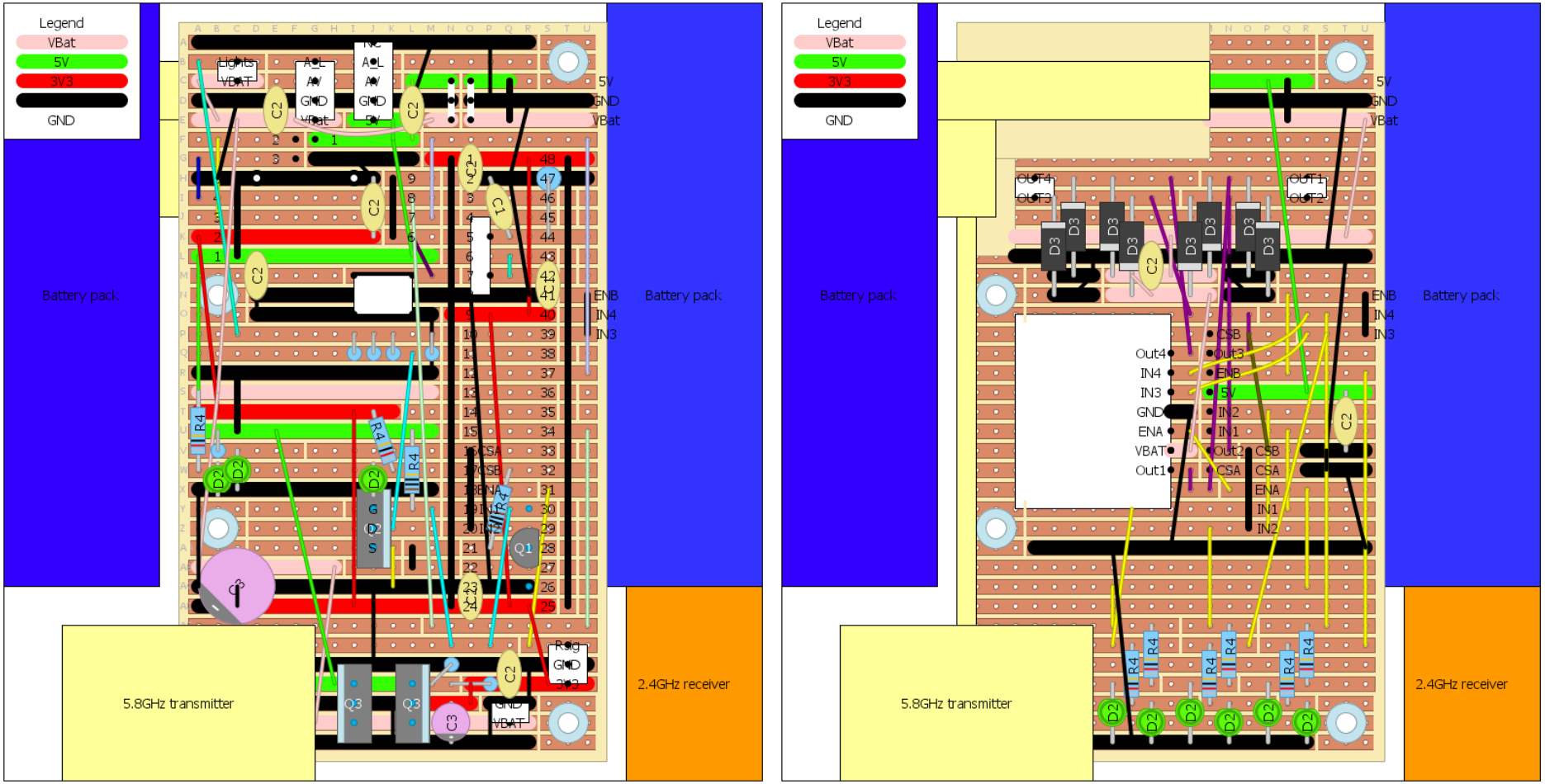
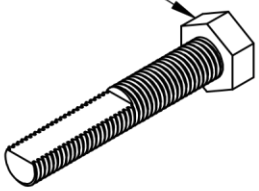
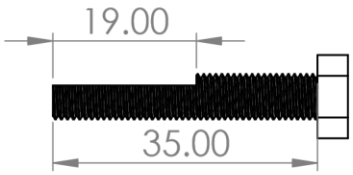


Figure 70: Full scale image of Veroboard design

Following this is the technical drawings for the various shafts in the assembly, and the laser cutting sheets.

M5 hexagonal head bolt





UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	FINISH:	DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING / REVISION

	NAME	SIGNATURE	DATE
DRAWN			
CHK'D			
APPV'D			
MFG			
Q.A			

TITLE:

MATERIAL:

DWG NO. ShaftPlanet

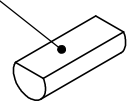
A4

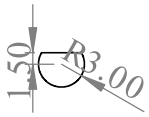
WEIGHT:

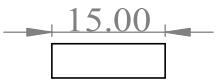
SCALE:1:1

SHEET 1 OF 1

6mm aluminium rod







UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	FINISH :	DEBURR AND BREAK S HARP EDGES	DO NOT SCALE DRAWING / REVISION

	NAME	SIGNATURE	DATE
DRAWN			
CHK'D			
APPV'D			
MFG			
Q.A			

TITLE:

MATERIAL:

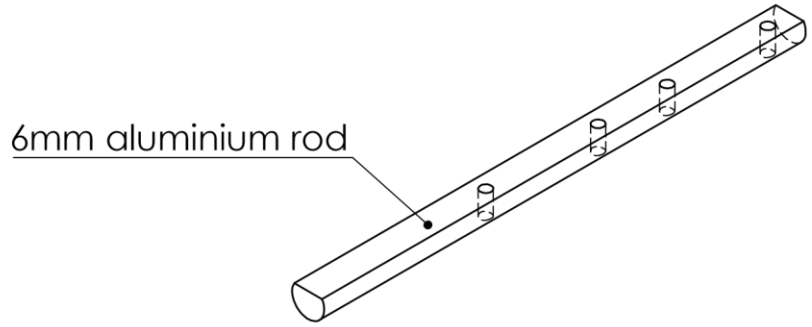
DWG NO. ShaftIdler

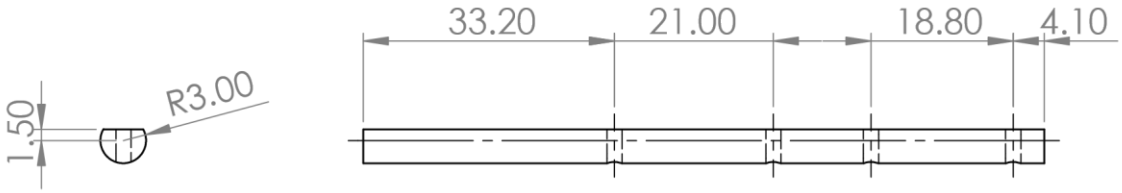
A4

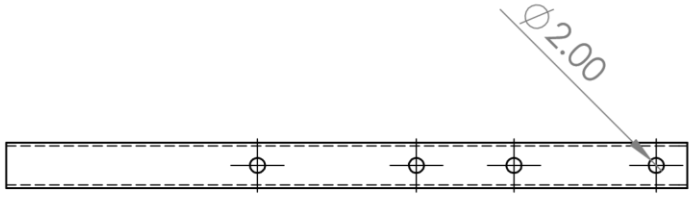
WEIGHT:

SCALE:1:1

SHEET 1 OF 1







UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	FINISH:	DEBURR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING / REVISION

	NAME	SIGNATURE	DATE
DRAWN			
CHK'D			
APPV'D			
MFG			
Q.A			

TITLE:

MATERIAL:

DWG NO. ShaftOut

A4

WEIGHT:

SCALE:1:1

SHEET 1 OF 1