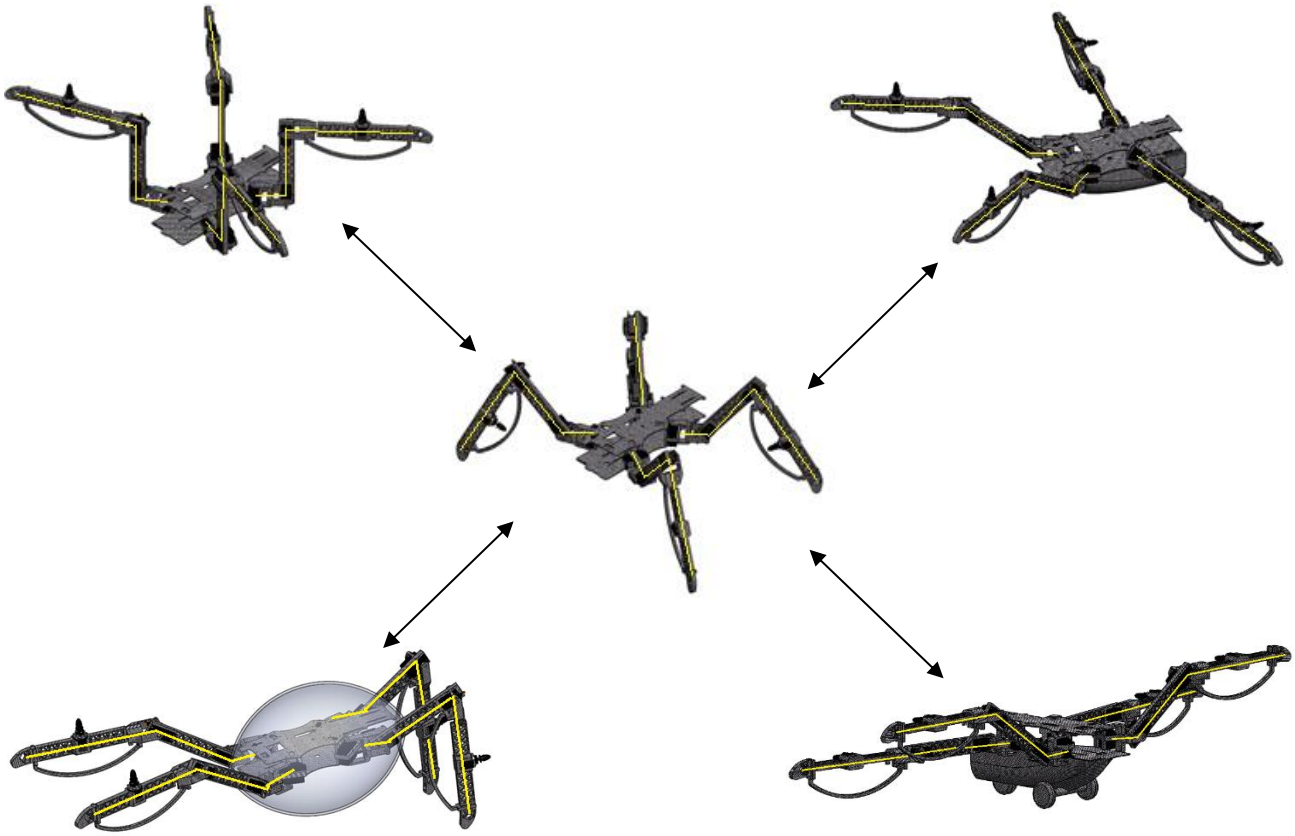
CHALMERS



Dynamic multidimensional platform

Bachelor's thesis in Electrical Engineering

ADAM BAJRASZEWSKI
STEFAN MARKOVIC

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2019

BACHELOR'S THESIS

Dynamic multidimensional platform

ADAM BAJRASZEWSKI
STEFAN MARKOVIC

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Dynamic multidimensional platform
ADAM BAJRASZEWSKI
STEFAN MARKOVIC

Examiner: JONAS DUREGÅRD

Department of Computer Science and Engineering
Chalmers University of Technology / University of Gothenburg
SE-412 96 Gothenburg
Sweden
Telephone: +46 (0)31-772 1000

Cover:
Image of the 3D designed platform with transformation phases.

Department of Computer Science and Engineering
Gothenburg, 2019

ABSTRACT

The project is to be carried out to reach places such as caves, mountain peaks, or other places without exposing a driver or pilot to danger. The purpose is to try to make a platform that, intelligently, moves around in these environments. This is an area where there is potential for improvement and therefore became the choice of this project.

The goals for the project were implemented successfully into the platform. The platform's design was successfully made to transform into the three phases and have implemented ground-borne movements. The three phases are Ground-, air, and water-borne. The design also had implemented symbolic protection shields for the propellers. Wireless communication was implemented.

The project mainly focused on the ground-borne phase and movements, while the rest of the phases symbolizes the purpose of this project, which intelligently, move in any terrain or direction. This can be done by allowing the platform to transform into the most efficient of the three phases, corresponding to the challenge it will face. The platform's potential is more significant than just the three previously mentioned phases.

Keywords: Dynamic platform, multidimensional platform

PREFACE

This is a thesis project done at the department of Computer Science and Engineering at Chalmers. The project is the last segment of our Bachelor of Science degree. We want to thank our supervisor and friend, Sakib Sistek at Chalmers University, as this project would not have been made possible without him. With his support and guidelines, there was no obstacle too challenging to overcome.

Adam Bajraszewski & Stefan Markovic, Gothenburg, May 2019

Table of Contents

Terminology

CW – Clockwise

CCW – Counterclockwise

PWM – Pulse Width Modulation

ESC – Electronic Speed Controller

RPM - Revolutions per Minute

BLDC – Brushless DC Motor

DC – Direct Current

IDE – Integrated Development Environment

RC – Radio Control

CAD – Computer Aided Design

GUI – Graphical User Interface

1. Introduction

In this chapter, the project's background and the main idea is presented, followed by this project's purpose, goals, and limitations.

1.1 Background

It is not surprising that the human body becomes exposed to great dangers at high altitudes. Especially at extreme altitudes as on the next mountains; Mt Everest, K2, and Cho Oyu. Many of the strength-testing climbers die because of the extremely challenging environmental conditions at these mountains. Whether it is from extreme cold, avalanches, or oxygen limitation, it is well known that many people have done their last climb at some of these mountains. Fatal incidents could occur, not only climbers but also people working in rescue organizations around these. Without flat surfaces for helicopters to land or weather-conditions to even allow secure flights, there are limitations to rescues there.

Despite this apparent danger, there are currently, among others, dedicated rescue-groups in the area around Mt Everest in Nepal. These groups attempt rescues in case of any reported emergency. In 2009 experienced Swizz mountain rescuers started a private organization where they educate people in medical aid and piloting to support Nepal expeditions at Mt Everest. This project has expanded since 2009, and today, Swizz and Nepal are working closely together to specialize their pilots to gain the ability to fly in even more extreme conditions. [1]

Problems with the helicopter's size, landing ability and the exposure of human pilots and crews, suggest that the vehicle used in specific missions should be updated. Comparing these helicopters with today's technology, perhaps some of these rescue missions could be possible with a quadcopter.

Quadcopters can quickly be started up and instantly ready to locate a person in an emergency. By modifying the quadcopter to be ground-borne, to move through uneven surfaces, or water-borne, to move through water, they could get even closer to its destination.

Other studies have noted the advantages of implementing the ability for movement in more than one possible dimension. Among these are the two-phased robot project developed by four students at the King Mongkut's University of Technology Thonburi in Thailand. In that project, issues regarding the rescue-availability in collapsed buildings were addressed and solved by implementing two movement-phases on a robot. This robot was designed to be driven in two different modes: rolling and walking. Rolling movement, for moving across flat surfaces, and walking mode, for climbing small obstacles. [2]

Designing a remote-controlled platform that can explore places that are near impossible to reach comes with problems in the availability of repairs. Therefore, it is required that the platform is maintenance-free. Since the platform's battery has a finite amount of energy, it must be charged with energy provided by the surrounding environment. Solar panels could be used to charge the platforms batteries. Deployable

shells for the propellers and protection for the solar panels could be implemented to protect the platform's vital parts.

1.2 Main idea

The main idea of having a dynamic multidimensional platform is to perform specific tasks such as rescue missions, observation, or information gathering. All this without risking the life of a pilot or passengers and reaching places current vehicles can not easily reach. This platform can reach specific places by transforming itself into different types of vehicles. To know what the most efficient path and vehicle type is, information is needed. This information can be gathered in different ways; one of them is by deploying a drone. In figure 1.1, there is an illustration of this.

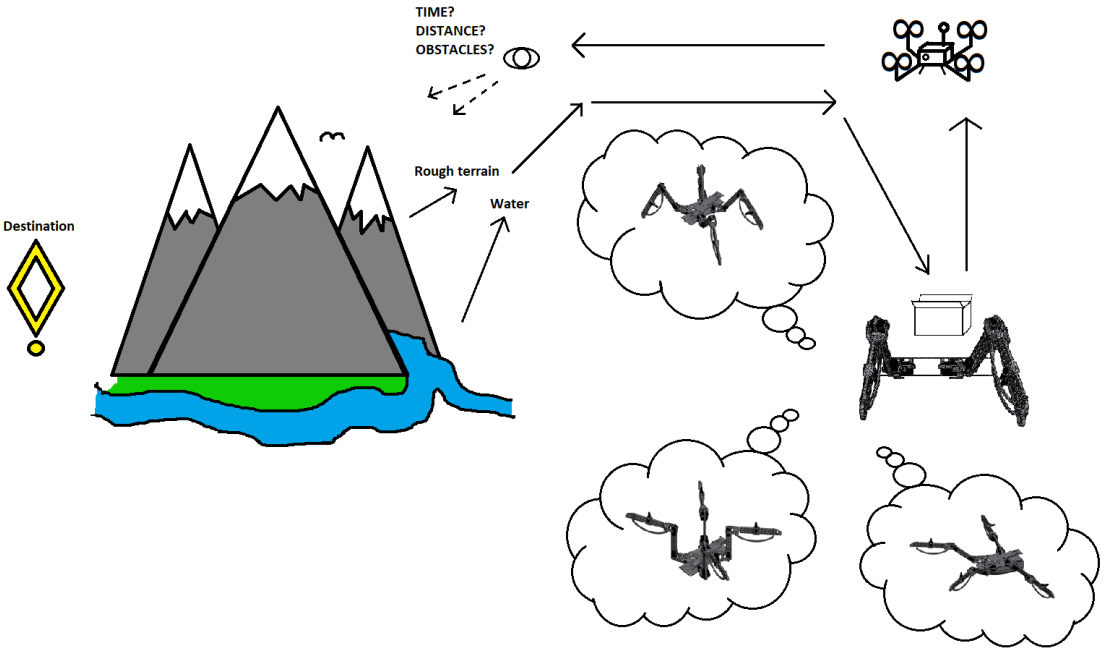


Figure 1.1: *Illustrates the implementation of a drone that can scout the upcoming terrain and return with valuable information.*

The drone on the platform is for acquiring information about upcoming terrain and its possible obstacles. This drone is enclosed inside the platform and is released upon the need for new information. Upon release, the drone takes to the air and starts the information gathering immediately with time traveled, distance, and visualization of obstacles the platform would encounter. However, this drone must have some model-based predicate control implemented as it encounters obstacles. "Model-based predictive control (MPC) is an approach that allows the incorporation of constraints on the control objective; thus, it has the capability to meet the avoiding collisions objective." [3] MPC is a system that allows for predictions based on the models in a library. The models can be anything, for example, rocks, trees, or cliffs that can be a hindrance. The drone uses this system to determine whether the obstacle it is facing is a hinder or not. After the information is gathered, the drone returns to the platform.

With the information gathered by the drone, the platform can now decide its route based on what path is the most energy conservative. If the energy consumption of the platform for its different phases is known, these can be used in an algorithm to

determine which is the best viable route and transformation to sustain stored energy. With tests on energy consumption in different scenarios such as hard wind, deep snow, or rain, the platform could also consider these factors. In theory, this will result in an increased lifespan. The platform's three different phases are designed to perform different movements depending on the environment. If the platform is facing obstacles too hard to overcome in one phase, for example, a big rock in the ground-borne phase, it has the option to transform into either the air-borne or water-borne phase. The platform can optimize itself depending on obstacles on its path to its destination.

The ideal version of the platform is battery powered and has solar panels where it could be charged with a renewal energy source, solar energy. Wasting energy is not beneficial for the battery and in the long run, the environment, because of charging many battery cycles.

With the possible phases, several vehicles can fit into one platform. This means that the material and components for all these vehicles can be reduced since it is one platform.

The ethical application of the platform is to save lives or assist during natural disasters such as forest fires. Not to be used for harmful purposes.

In this project, only the design and functions of transforming the platform into the three different phases and ground-borne movement will be done.

1.2 Purpose

The project's purpose is to design and construct a prototype platform that could reach dangerous places without exposing a driver or person to danger. The platform should be able to transform into three movement phases, making it multidimensional. These phases are for ground-borne, air-borne, and water-borne movements.

1.3 Goal

The goals for the project are to implement these following criteria:

- Transform the platform into a ground-borne position.
- Transform into an air-borne position.
- Transform into a water-borne position.
- Incorporate ground-borne movements.
- Implement a design that protects the hardware equipment.
- Implement a wireless controller for the transformation phases and movements.

1.4 Limitations

The project's limitations are the following:

- The platform will not be tested in actual dangerous environments.
- The platform will not have implemented movements for air and water.
- There will be no sensors or GPS-modules implemented in this project; however, implementing these afterward on a working platform should not be a problem.

2. Theory

In this chapter, the project's theory is presented. First is the theory about ground movement followed by air movement. After these two sections, a theory about the hardware components is presented.

2.1 Ground movement

Many of the places and environments the platform is meant to explore are mostly already inhabited by lifeforms. Lifeforms such as animals, insects, and others. These creatures are exposed to and can handle the same kind of environment as the platform is going to be exposed to. Therefore the inspiration for the platform's ground-borne movements is taken from arthropods, more specifically, spiders. This arthropod can climb on vertical walls, running through uneven surfaces without hurting its fragile body and can do all this with great agility and speed. This is the reason for replicating a spider's movement for this project's ground-borne movement.

The spider belongs to a class of joint-legged invertebrate animals called arachnids. Apart from many other insects, the spider has two body parts instead of three. It has eight legs, which are all connected to the body and could be referred to as four pairs of legs. The spider's leg is divided into eight parts and are called; coxa, trochanter, femur, patella, tibia, metatarsus, tarsus, and claws. This project will only refer to the three main parts of the spider's leg, the coxa, femur, and tibia. [4] These parts can be seen in figure 2.1.

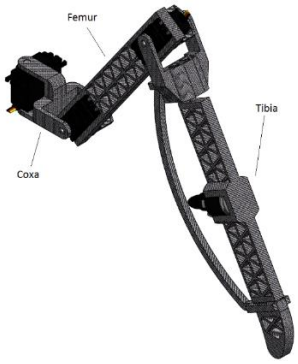


Figure 2.1: *Model of the leg, with Coxa, Femur, and Tibia labeled.*

Each part will be controlled by servo motors for the platform to perform the ground-borne movement. Some limitations to the rotations of the parts will be applied. The limitations can be seen in figure 2.2. Coxa-, Femur, and Tibia-parts will have a limited rotation of roughly 125, 210, and 270 degrees. The limitations of these angles must be taken into consideration in the software where the servo motors will be programmed and controlled.

Figure 2.2: *Maximum rotational degree from left to right, Coxa – 125°, Femur – 125° and Tibia – 270°*

During ground movement, there are certain areas in which the leg will be moving. These working areas represent a limited area of movement of which each leg can operate. If one of the legs would operate outside this area, there's a possibility of overlapping parts. This would result in damaged hardware and a malfunctioning platform. These working areas can be seen in figure 2.3, where number 1 represents the back pair and number 2 the corresponding front pair. This also must be considered in the software development process.

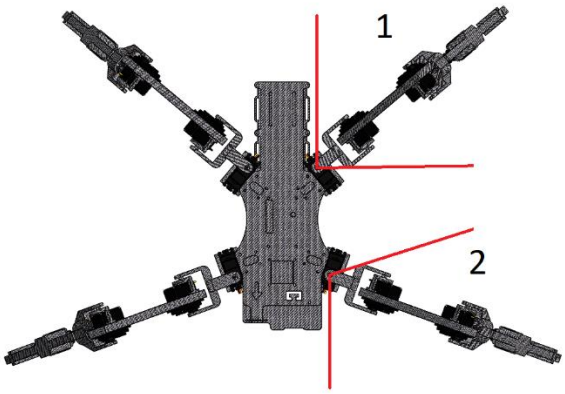


Figure 2.3: *1 and 2 represent the working areas for the ground movement.*

As seen in figure 2.4, each row of legs from the front to the back are called sets. This means there is a total of four sets. Each of these sets works as human legs if viewed independently, where only one leg moves at a time. As the first set moves, the second one works in counter to the first one. The third set copies the first set's movement, and the fourth respectively copies the second ones.

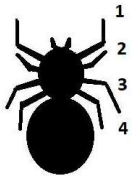


Figure 2.4: *1, 2, 3, and 4 represent the leg sets.*

In this project, there are only four legs, which is two sets. These sets move in the same way the spider's two sets are. The walking sequence referred to as figure 2.5 is 1, 4, 2, 3, and repeat.

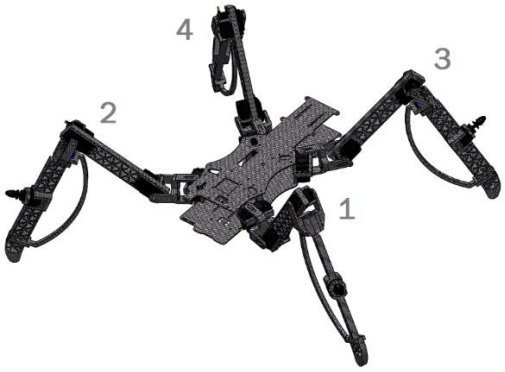


Figure 2.5: *Platform's legs numbered from 1-4 to achieve the spider's walking sequence.*

2.2 Air movement

To get an understanding of computer-aided flight-control systems, one must get familiar with the history of aircraft development.

In the year 1922, the first quadrotor aircraft was created. This was named "The Wright Field" This was the first piloted quadcopter and could reach an altitude of 5 meters. Achieving a hovering function was done by creating a lift force caused by its four rotors. Compared to the working principle of the aircraft that relies on passing air, the quadrotor aircraft relied on rotating rotors to stay in the air. This means the quadrotor aircraft did not need to be constantly moving to stay air-borne. In fact, at the time, the lateral movement for a quadrotor aircraft was impossible for a pilot to handle manually, and this would result in a temporary stop of the quadcopter's development. The development was not re-engaged until the invention of computer-assisted flight-control systems. [5].

It was not until 40 years after the invention of the Wright Field that the Osprey aircraft was designed, an aircraft that also had horizontal rotors. The Osprey looks and flies like a plane, using its wings to sore through the air, but with the hybrid touch to it. With the help of two adjustable rotors on each wing, the Osprey merged together with the functionalities of a rotor-based aircraft with the lateral movement ability of a plane. It both landed and started vertically, like the Wright Field. This is what makes this aircraft especially interesting and relevant to this project due to its transforming ability. During those 40 years, computer-assisted flight-control systems were invented and developed. These systems allowed air-borne vehicles to implement more complex flight controls and instructions and did not require them to be manually controlled. Instructions include balancing a quadrotor aircraft and at the same time gain lateral movements by tilting the aircraft slightly without losing this fine balance.[5]

The quadcopter used today, just as the quadrotor aircraft invented 100 years ago, gain a lift force with the help of its four rotational motors. They rotate at very high speed, and the propellers create air pressure and gain altitude. Some of the motors are set to rotate in two different directions. Clockwise and counterclockwise. Which motor is rotating in what direction can be seen in figure 2.6. Here the motors are labeled clockwise (CW) or counterclockwise (CCW).

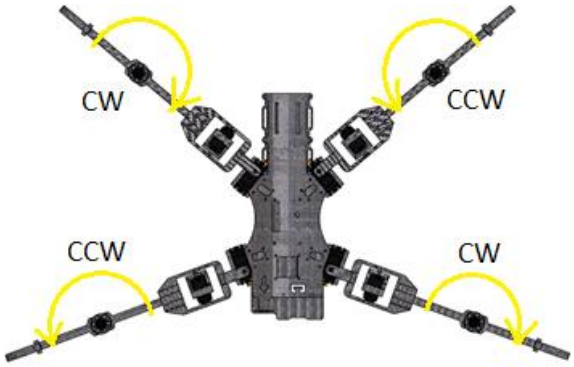


Figure 2.6: *Direction of the motors is labeled CW or CCW.*

By rotating the propellers accordingly, CW and CCW, it makes sure that the quadcopter doesn't self-rotate. By increasing the motors Revolutions per Minute (RPM), the quadcopter will gain altitude and respectively lose altitude when the RPM is decreased. Each of its motors RPM can be controlled separately, which is very useful when weather conditions or other phenomena might affect the four motors differently.

In this project, the communication with the platform will be wireless. It is done with a transmitter and a receiver at the 2.4GHz radiofrequency. The signals sent to control the motors are done through Pulse Width Modulation (PWM).

The direction of the quadcopter is decided by how its axes that can be seen in figure 2.7 are rotated. The roll axis goes from the center of gravity to the nose of the aircraft and decides the rotation of the roll axis by moving the wings up respective down. The pitch axis is located across the wings and decides the vertical angle of the aircraft. The yaw axis is the axis that decides if the nose points either left or right.

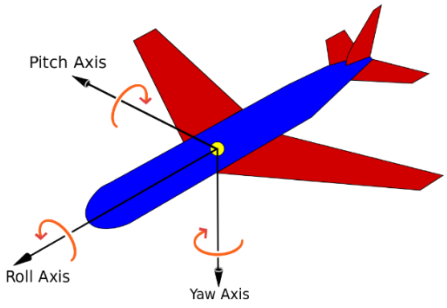


Figure 2.7: *An airplane with the Pitch-, Roll- and Yaw axis labeled. [6]*

2.3 Brushless DC-motor

The Brushless Direct Current motor (BLDC) is a non-frictional motor, and therefore, its name "brushless." The BLDC can be divided into two parts, the stator, and the rotor. The stator and rotor can vary between consisting of either inductors or magnets. These can also vary in the number of magnets and inductors. As the names suggest, the rotor is rotating around the stationary stator with the help of magnetic forces. [7]

Explaining the principles of the BLDC, it is easier to limit the amounts of magnets in the rotor to four and inductors in the stator to three. Usually, a BLDC contains more than four magnets and three inductors, but this will prove more difficult to illustrate and explain.

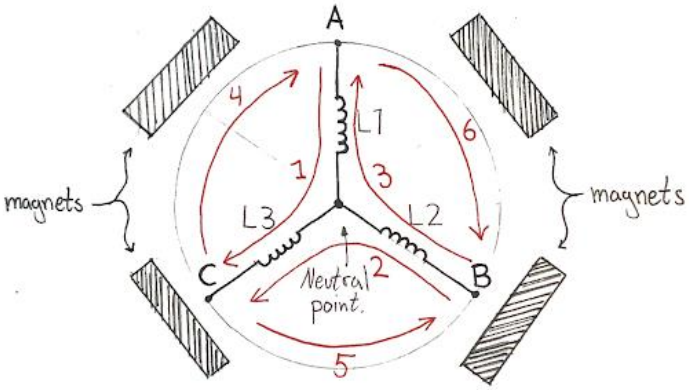


Figure 2.8: BLDC Motor

Step	1	2	3	4	5	6
High	A	B	B	C	C	A
Low	C	C	A	A	B	B

Table 1: *Step, High, Low of the BLDC motor.*

In figure 2.8, three inductors are each connected to respective points A, B, and C, which is surrounded by the four magnets in the rotor. These three points have the working function of a three-phase voltage and are connected to a single face DC power source via 6 MOSFETs. The red arrows labeled 1-6 are directions in which the current flows through the inductors. In the current direction 1, as seen in figure 2.8, the current will flow from point A to point C, and in the current direction two, it will flow from point B to point C. The same goes for the current direction 3 to 6 but with variations in from which point the current to flow to what point. This is done by letting one of the phases be grounded, while one is applied a higher voltage, and the last one left to be natural. These mentioned current directions are what make the BLDC step, or more specifically rotate a notch. The table above, table 1, shows which point has a high or low voltage potential during what corresponding step.

When current flows through each of the three inductors, magnetic fields are created. These fields will attract or repel the magnets in the rotor, making the rotor turn towards a specific position. Therefore, it is called a step. With every new direction and path of the current, the rotor rotates into a new position. By rapidly change which of the points, A, B or C, should be conducting and which of them should have high or low voltage, the rotor will gain a continuous rotation around the stator.

Even though the BLDC is, as the name suggests, a DC-driven motor, it functions with a three-phase voltage input. The three-phased voltage input usually comes from a single phased DC power source that via MOSFETs gets converted to three-phase. By controlling the MOSFETs, the BLDC can be controlled.

2.4 Electric Speed Controller

The Electric Speed Controller's (ESC) job is to control the BLDC motor with the desired speed decided by the input signal. The input signal consists of a PWM signal that determines the speed depending on the duty cycle. The ESC also converts a single-phase DC input to a three-phase DC output.

The ESC used in this project consists of N-channel MOSFETs which are connected to each phase through a half-bridge, 3 phase; therefore, three half-bridges. These work in such a way that when current direction 1, as seen in figure 2.9 and previously mentioned in the BLDC chapter above, is active the MOSFETs open in such a way that current will flow from point A to point C inside the BLDC. This makes A high, C grounded and B natural.

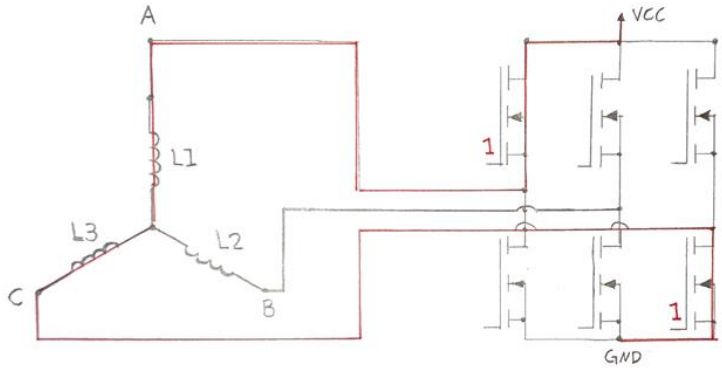


Figure 2.9: *ESC Circuit and connected BLDC motor, current direction 1.*

By opening and closing different MOSFETs, all steps can be created. As seen in figure 2.10, the number on the left side of the MOSFET indicates on what step the BLDC will take once they are open on both sides of the half-bridge.

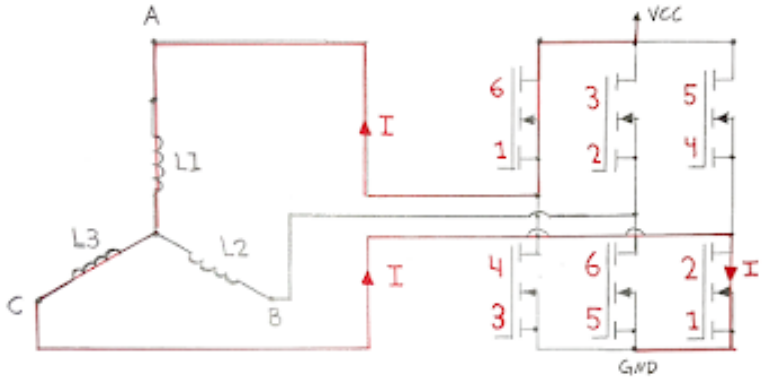


Figure 2.10: *ESC Circuit, with all gates numbered.*

2.5 Pulse-Width-Modulation (PWM)

Pulse Width Modulation (PWM) is modulated pulses, usually in the shape of square waves. "In PWM, which is also called pulse duration modulation (PDM), sample values of the analog waveform are used to determine the width of the pulse signal." [8] This means that these pulses consist of digital values of either one or zero. PWM can be

used to control a BLDC motor, a servo motor, and other analog remote control applications. [9] To control these hardware components, the signal has to be modulated in such a way that analog signals are being sent as voltage pulses. Different duty cycles can be seen in figure 2.11.

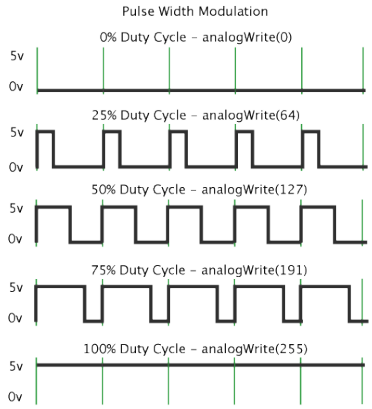


Figure 2.11: *Figure illustrating different duty cycle sizes.* [10]

These can be set to different sizes depending on wanted attributes. The longer the duty cycle is, the longer the pulses will be. Usually, these pulses only control the time of which power supply is active. The pulses are sent to hardware, which determines if the power supply should be active or not. In an active state voltage, from the power supply, will be applied to the specific component controlled by the PWM. A longer ON-time will result in more voltage applied to the component. By regulating the duty cycle of the pulse, for example, the speed of a motor can be controlled. This technique will be applied in the project for both our BLDC motors and the servo motors. The Arduino™ will be creating the desired PWM for the project's application.

2.6 Servo motor

The servo motor is an electrical device that can rotate an object around its axis. This is done with decent precision and high torque and can be fitted into a small, lightweight package. Because of these factors, the servomotor is highly represented in small electric machines such as Lego cars and other robotics. A servo motor's strength is rated in kg/cm, which is a measurement of how much weight the motor can lift at a specified distance.

The servo motor's working principle can be explained by explaining its three signals. The first is the input signal. This signal is a user-controlled PWM signal that decides the requested rotation speed, rotation direction, and final angle of the servo. With the usage of a potentiometer on the shaft of the servo motor, its current position can be measured, which will generate our second signal. This signal is referred to as the output signal and is a real-time measurement of the servos angle. By comparing these two signals, the input signal, and the output signal, a third signal is generated. This third signal is based on the error between the first two signals. The differences between what the user demands the servo angle to be and the actual angle of the servo. The third signal, the error signal, is amplified and applied as the input of the motor, making it rotate. Once the comparator detects no difference between the input and the output signal, the error signal will be off, and the motor will stop. [11]

From these PWM signals, the motor will turn depending on the size of the duty cycle. These sizes in the PWM's duty cycle also determine which direction the motor turns, either clockwise or counterclockwise. As an example, if there's a 1,5ms pulse, the servo motor will turn clockwise into a 90-degree position. If the pulse were to increase its pulse size to 2ms, the servo motor would turn clockwise into a 180-degree position. However, if there's a pulse size that's less than 1ms, then the servo motor would turn counterclockwise from its 90-degree position into a 0-degree position. [11].

2.7 Hardware

2.7.1 Arduino™

Arduino™ MEGA 2560 REV3, which can be seen in figure 2.12, is a 16-bit microcontroller based on the microprocessor ATmega2560. This development board is widely known for its use in projects where prototyping and the first stages of development are needed. Thanks to its IDE (Integrated Development Environment) it is easy to compile and upload code since the Arduino™'s language is a simplified version of C/C++. The microcontroller board is compatible with both analog- and digital signals. This is the main reason why this project uses an Arduino™ MEGA 2560. [12]

Figure 2.12: *Arduino™ MEGA 2560 REV3. [12]*

2.7.2 Adafruit™ 1411 Servo shield

The Arduino™ is capable of controlling servo motors on its own. However, the Arduino™ MEGA 2560 REV3 has its limitation of pins and a limitation of Arduino™ processing power. The project consists of at least 12 servo motors. Each one of these requires a PWM- and a powerful signal to work. This is where a servo shield will be of considerate help. A servo shield from Adafruit™ is chosen to control the servo motors. The Adafruit™ is a 16-Channel 12-bit PWM/Servo Driver Shield and would probably work as it covers more than 12 servo slots. The servo shield can be seen in figure 2.13.

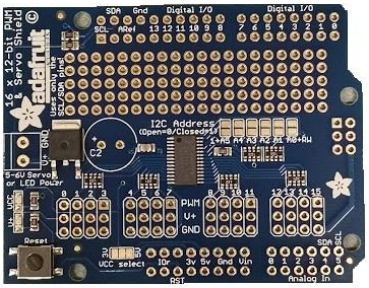


Figure 2.13: *Adafruit™ 1411 Servo shield.*

2.7.3 Servo motor DSS-M15S
The implemented model of servo motor in this project was the DSS-M15S servo motor. This servo motor is a metal-geared 270-degree servo motor and is a robust servo motor with more than enough torque for the project. It has a pulse range of 500-2500 microseconds with an operating frequency range of 50 to 330 Hz. The voltage and current it operates at is 6V respectively 80mA, which is within the compatibility range for the rest of the hardware components. DSS-M15S can be seen in figure 2.14.

Figure 2.14: *Servo motor DSS-M15S.*

2.7.4 Brushless DC Motor MT221611
The BLDC motors used in this project comes from a company named Tiger. These MT221611 has 12 nodes and 14 poles and is in this project powered by an 11.1V battery.

The size of the propeller used with these motors will vary in size and because of this vary in lift force, efficiency and power needed.

2.7.5 Electronic Speed Controller HobbyKing™ 30A
The ESC used in this project is the HobbyKing™ 30A ESC. These ESCs have a few programmable options for frequency, timing, start-up speed, brake, and precise throttle responses.

The three available start-up modes are the following: normal, soft, and super-soft. [13] The used one in this project will be the normal one. It's recommended to use normal for fixed-wing aircraft, while the other two for helicopters. By using the super-soft start-up mode, the start-up becomes less rough and puts less demand for the platform design.

It also has three protection features. These features are low voltage cut-off protection, over-heat protection, and throttle signal loss protection. All these are useful when it comes to aircraft. Not having a throttle signal loss protection in an air-borne vehicle 30m in the air would result in a hard crash if the signal were to disappear.

2.7.6 FlySky™ FS-T6 Controller and Receiver
In order to transmit and receive signals, the FlySky™ FS-T6 controller and its receiver were chosen. These components operate at the 2,4GHz frequency and have good reach with stability, which was also tested. The amount of independent received signals is up to 6 channels. The image of the controller and receiver can be seen in figure 2.15.

Figure 2.15: *FlySky™ FS-T6 Controller and a receiver module.*

2.7.7 Gens Ace™ 3S1P Li-Po Battery

The battery supplying the BLDC motors can be seen in figure 2.16. This is an 11,1V Li-Po battery that's specifically used within the RC vehicles because of its high discharging capacity. This battery is, however, only 2200mAh but will be more than enough to supply our test runs and developments.

Figure 2.16: *Gens Ace™ 3S1P Li-Po Battery 2200mAh.*

2.8 Software

2.8.1 C Programming

The C language is one of the most commonly used languages within programming. The language is powerful and can be used in all levels of programming. Meaning it does not matter if it is used for hardware development or operating systems. The language is fast and reliable for hardware development, which makes it a natural choice of language in this project. [14]

2.8.2 Arduino™ IDE

The Arduino™ Integrated Development Environment is used in the project to develop, compile, debug, and upload code to the Arduino™ MEGA 2560 REV3. The IDE offers great tools for testing the program, such as the serial monitor.

2.8.3 SolidWorks

SolidWorks is a 3D-CAD Software used for designing and creating parts. It is easy to use with its friendly GUI. It also gives the possibility to create single parts first by specifying dimensions and then assemble these parts into an assembly. This is great for designing and modeling before practically making the parts in real life. By assembling these parts into a complete assembly, there is also the possibility to see if certain joints are connected correctly and then make decisions from there.

The software offers integrated simulation tools to stress the physics, temperature, dynamics, etc. of the model. It can also calculate the weight of the model with a specified material. It is also possible to make realistic photo renderings with lighting, shadows, etc. with the software.

The main reason for using SolidWorks as the 3D CAD Software is because of having years of previous experience of this program and its versatility to convert the designed parts into STL-files which is the file-type that is compatible with 3D-Printing machines.

3. Method

In this chapter, the project's method is presented – first, the planning of the project following the working process.

3.1 Planning of the project

Following is the agenda of which the project was based on:

- Study about the hardware function
- Design the platform v0.1
- Hardware assembly
- Software tests for the functions
- Design the platform v0.2
- Hardware implementation
- Software test and implementation

3.2 Working process

This project's working process began with brainstorming. During the brainstorming process, the main goal was to find solutions for the problems this project was facing. Problems such as how to implement the three different platform phases. Also, how to transform between these and what demands the transformations would have on the design that was to be constructed. Another problem that was addressed was how to protect the platform's propellers and how it would function. All these were discussed and planned during this process.

After visualizing the ground laying design of the platform, the hardware to make the platform function were identified. Once the platform's components were acquired and understood, the platform's design was practically visualized using Computer-aided design (CAD) programs.

The design got printed using a 3D-printer that was capable of reading STL-files from the CAD program. Once the printing process was done, the parts were assembled and chosen hardware implemented. The design and hardware were constantly tested and upgraded if problems occurred.

Throughout this project, software tests on the hardware were done to gain knowledge of how to control it. The knowledge acquired from the tests was used to implement movements and transformations to the platform.

4 Design process

In this chapter, the project's design process is presented – first, the versions following the transformation design.

4.1 Platform v0.1

Designing the platform was done using SolidWorks CAD Software. The design was originally created based on the dimensions of the SG90-Servo and the BLDC motor MT221611. These are the two motor models that initially were used in the project. During the first tests of assembling the 3D printed model of the early stages of the v0.1, major flaws were detected. Flaws such as the model being too thin and therefore broke in half after a mild stress test. Without the implementation of truss in the model, its weight would quickly add up to one impossible to be carried by only 4 BLDC.

After the first 3D printed part, flaws such as too thin alignments had to be taken into consideration wherever slim parts were needed. The parts dimensions needed to have a thickness of a minimum of 2-3mm. Before the platform v0.1, as seen in figure 4.1, was printed notation on the SG90 servo motors was made. Because of changes in servo motors, there were new specifications that this current platform would not achieve. Therefore, the platform had to be redesigned to fit the new, bigger servo motors.

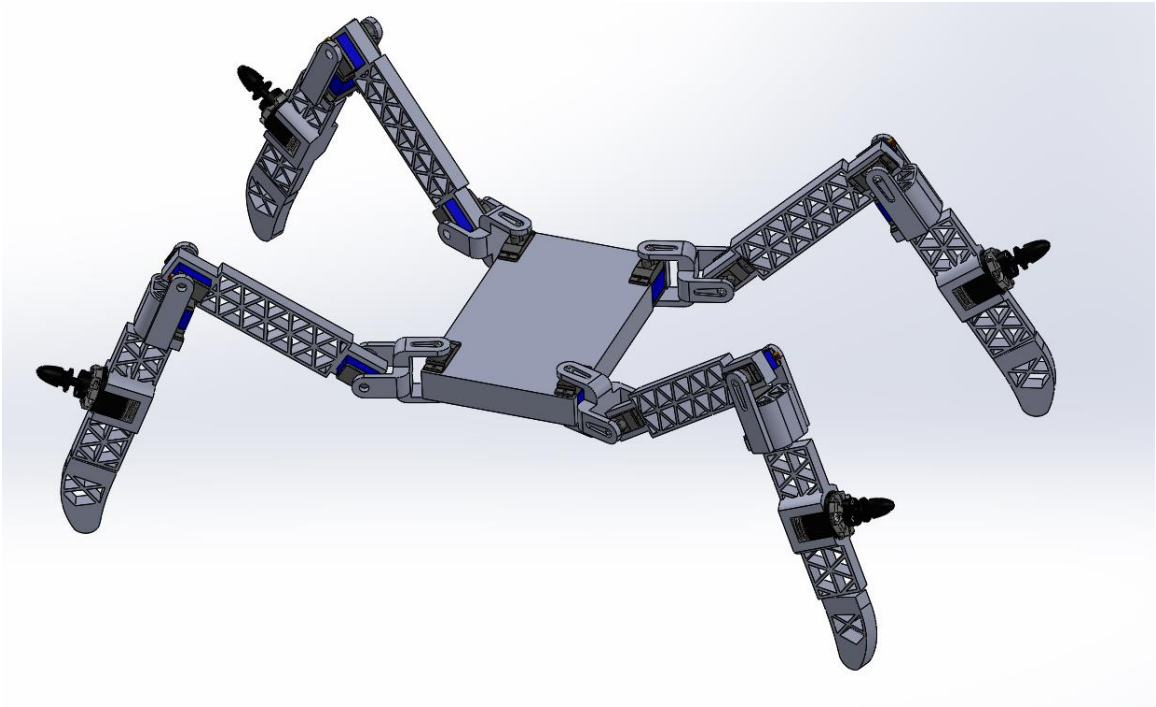


Figure 4.1: *First version of the 3D designed platform.*

4.2 Platform v0.2

A redesigned version of the platform had to be a lot bigger to fit the new DSS-M15S servo motors. Because of this, some parts had to be extended and not only enlarged, parts such as the Femur, Tibia, and the Femur-to-Tibia-part. The DSS-M15S servos also got the option to rotate the Femur-to-Tibia-part 270 degrees, and not only 180 as the old ones, allowing the robot to have the option to do a turnover if flipped. In order to avoid a malfunctioning platform, these new extended rotation limits had to be taken into consideration while designing it.

Making the platform bigger means the project's upcoming functions and ideas instantly become harder to implement. An increment in size led to more mass, and bigger body parts that had to fit in the 3D printer. The limit of the 3D printer was to print pieces of maximal 200mm x 200mm. The redesigned platform can be seen in figure 4.2, and this became the platform's prototype model for this project.

Figure 4.2: *The second version of the 3D designed platform.*

4.3 Platform parts

The model design of the leg consisted of eight pieces. These pieces are called the following and can be seen in figure 4.3: Body-holder, Coxa, Femur, Femur-To-Tibia, Tibia, Claw, Attachment, and a propeller shield for the propeller-blades.

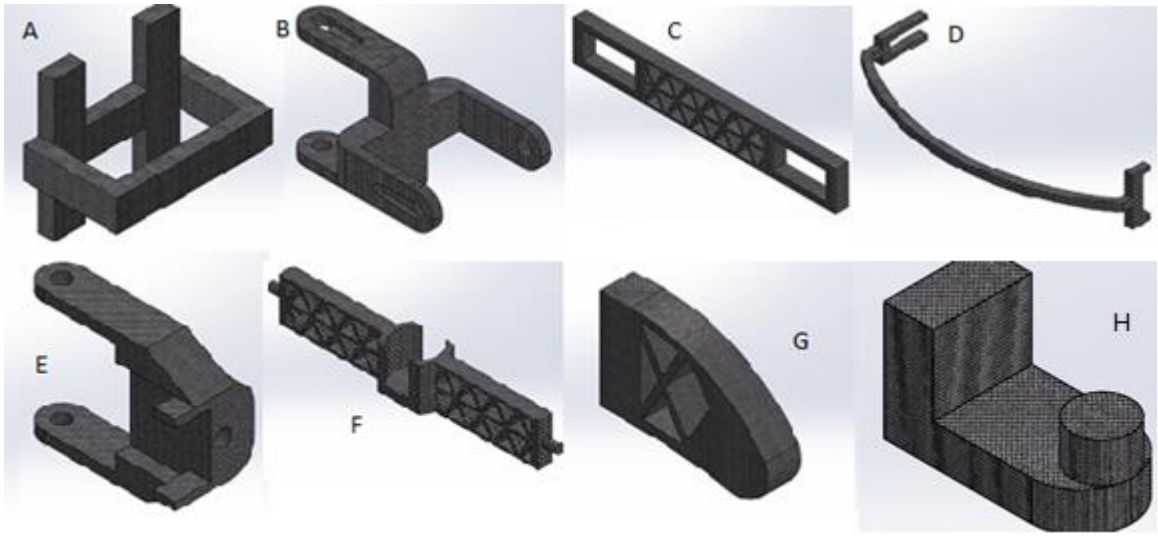


Figure 4.3: *The eight pieces that a full leg consists of, from A – H.*

A. Body-holder
This part is connected to the chassis and connects with the complete leg.

B. Coxa
The Coxa-part is bound to the Body-holder where one of the DSS-M15S servo motors is located and the Femur-part. The Coxa-part has a total of 2 servo motors connected to it.

C. Femur
The Femur-part is what lifts the leg, and the main function is the height adjustments of the platform. It is connected to one of the servo motors from the Coxa-part, and the other one is connected to the Femur-To-Tibia-part.

D. Protection frame
The protection frame-part is built to implicate a shell where the propeller blades could be positioned and protected, when not used. This part is connected to the Femur-To-Tibia and the Claw-parts.

E. Femur-To-Tibia
This part is designed in a way that it performs rotations with an SG90 servo motor located underneath. Implemented as safety protection for the motor and the propeller blade. This is achieved by the servo that turns the Tibia-part. This part is connected to the Femur and the Tibia.

F. Tibia
The Tibia-part holds the BLDC motor in place as mentioned above and rotates 180-degrees when not used, to protect the propeller blades.

G. Claw
This part is Claw-part. The part has the most ground contact out of all the parts because this is what the platform will walk on.

H. Attachment
The attachment holds the Coxa-part and the rest of the legs stable. It has been drilled and mounted on the platform frame to make this function.

4.4 Transformation phases
The idea of the transformation process is to achieve the following phases: ground-, air- and water-borne.

4.4.1 Ground-borne transformation
Once the initialization process is initiated, the platform will be set into a ground-borne stance. This stance consists of positioning the Femur-parts at approximately a 40- to 50-degree position. During this stance, the BLDC motors must be turned into the protective shell. The ground-borne stance can be seen in figure 4.4.

Figure 4.4: *Platform transformed into a ground-borne phase.*

4.4.2 Air-borne transformation
The first step of the air-borne transformation is to raise the Femur-parts into the air, and the Tibia-parts 90-degree angled from the Femur-parts. The next step of this transformation phase is to engage the motors and propeller blades in its position. This is done by rotating the Tibia-part 180-degrees out of the protective shell. The air-borne stance can be seen in figure 4.5.

Figure 4.5: *Platform transformed into an air-borne phase.*

4.4.3 Water-borne transformation
In the water-borne mode, the positioning of the front paired legs is set with a 20-degree forward offset from the initial Coxa position. The Femur- and Tibia-parts have slightly moved into a position that can be seen in figure x. The Tibia-parts are angled in an upward position keeping the balance of the platform. The positioning of the back paired legs and their Tibia-parts are set in a way that it can push the water backward making

the platform move forward. This can be done either with the propellers in the water or the air. The water-borne stance can be seen in figure 4.6.

.

Figure 4.6: *Platform transformed into a water-borne phase.*

5 Hardware assembly

In this chapter, the project's hardware assembly is presented.

5.1 Demo assembly

The 3D printed demo model was a spider design found on the internet. This design can be seen in figure 5.1 below. This model was based on the smaller SG-90 servos. This design did not have transformable legs, but with this, the goal was to learn about robotic movements using servo motors. Unfortunately, this design failed after detecting flaws with the SG-90 servos used in this demo model. Flaws such as lacking quality with curved and odd cogwheels. Other problems with this servo were also detected. Namely, if more than one servo were connected to the motor-shield at once, a significant voltage drop appeared, and the servos slowed down until they entirely stopped.

Figure 5.1 to the left, Figure 5.2 to the right: *Legs from the 3D demo model with SG90 servos. First printed test leg to the right.*

After the unsuccessful first 3D printed demo model, the own designed model was planned, sketched and executed, the platform v0.1. This version had legs with BLDC motor slots for airborne movements seen in the figure. Already after 3D printing one leg, a flaw in the design was detected. It was way too heavy and quickly broke. Due to the designed leg being made out of solid plastic, without air gaps or truss to decrease the weight. It was also too thin in the middle where the BLDC should have been positioned. The thin tibia can be seen in figure 5.2. The combination of the heavyweight and its weak spots lead to a broken leg within minutes of assembling.

A newer version, the v0.2, was designed with implemented truss and enough structural strength to support its weight. It was also designed around new servo motors, the DSS-M15S.

For the width and length, the servo motor had to be deconstructed and reassembled to fit on the femur. The slots also had to be ground for the servo motor to fit. Eventually, all the femurs were assembled with both servo motors and ESC connected, as seen in figure 5.3.

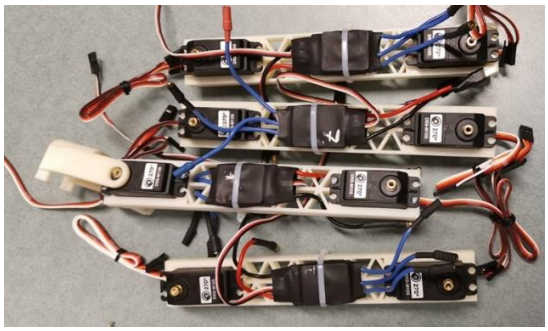


Figure 5.3: *Fully assembled femur with servo motors and ESC.*

The DSS-M15S servo motors could be fixed on the Femur-part. However, the Coxa- and Femur-to-Tibia-parts were too small. These parts were ground so it would fit the DSS-M15S, as seen in figure 5.4. However, this part got too thin and would easily break. After several attempts of grinding, the small leg parts got redesigned and reprinted. This time with enough room for the DSS-M15S servo motor and these parts could now be assembled to the femur.

Figure 5.4: *Grinding of the coxa*

Once the Coxa- and Femur-parts were assembled, they had to be connected to the Tibia-part. This part is seen in figure 5.5 below.

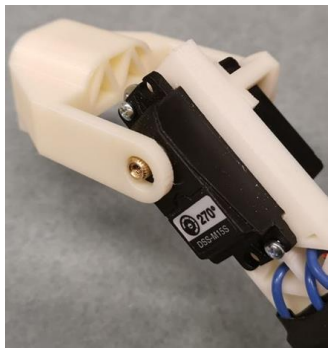


Figure 5.5: *Femur-to-Tibia-part.*

The Tibia-part was designed to hold a BLDC motor. This motor produces rotational force and therefore needed to be mounted in place. After mounting each BLDC to the Tibia-part, these were connected to the rest of the leg, as seen in figure 5.6.

Figure 5.6: The Tibia-parts with respective BLDC motors.

With the legs fully assembled, these were ready to be mounted to the frame. The frame, which was previously used as a quadcopter body, had to be slightly modified by drilling new holes were the Body-Holder-parts and legs were connected. This can be seen in figure 5.7 below.

Figure 5.7: *Body with drilled holes for the legs.*

6 Software tests

In this chapter, the project's software tests will be presented. The first part consists of decoding the transmitter following the tests of the ESC and BLDC motors.

6.1 Decoding the transmitter signals

The Fly SkyFS-6 Flight controller has a bind key that allows the receiver to be paired with it. While connected to the Arduino™, the LED on the receiver is lit red. This implied that the two components were successfully paired, and the transmission test could be started. During the hardware test of the transmitter and the receiver, the PWM signal was measured in three different cases. The first case, when the throttle was at its minimum position. The second case, when the throttle was at a neutral position. The third case, when the throttle was at its maximum position. In figure 6.1, the corresponding PWM signal of these cases can be seen.

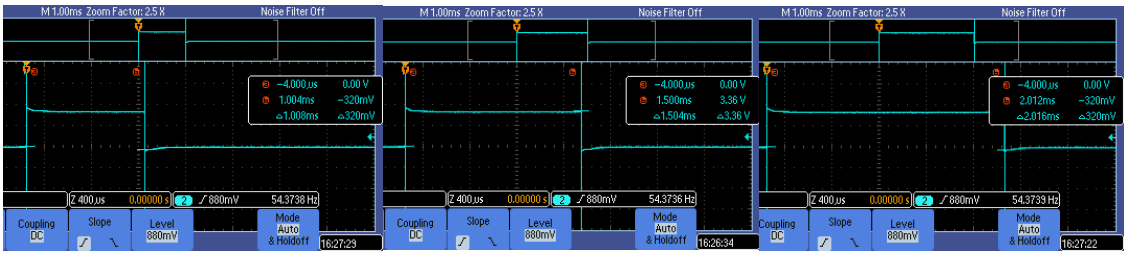


Figure 6.1: *Oscilloscope measured the positions from left to the right: min, neutral, and max.*

These limit values were crucial for successfully decoding the transmitter signals and programming the software to translate the PWM signals into values which later could be used. This was done by the implementation of an interrupt routine that would trigger each time the signal changes. In other words, it would trigger on either the rising edge or falling edge. By then entering the interrupt routine, the returned value would return as a calculation value of the pulse in microseconds.

The code can be seen in figure 6.2 below.

```
#define INTERRUPT_SIGNAL 0                      //The interrupt signal 0 is digital pin 2 in Arduino.
#define THROTTLE_SIGNAL 2                       //Pin 2 is used for the Channel 1 receiver signal which is the throttle's signal.

#define NEUTRAL_POS 1500                        //this is the duration in microseconds of neutral throttle on an electric RC Car.

volatile int newThrottleSignalValue = NEUTRAL_POS;      //This signal is set as volatile because we use Interrupt in our loop therefore it must be declared as vol
volatile unsigned long startPeriod = 0;                 //Has to be unsigned long for the micros() function.
volatile boolean newSignalIndicator = false;            //Defined as boolean; true or false.

void setup(){                                   //Setup for the Hardware interrupt, In the CHANGE-mode handled by the interruptRoutine.
  attachInterrupt(INTERRUPT_SIGNAL,interruptRoutine,CHANGE);
  Serial.begin(9600);
}

void loop() {
 if(newSignalIndicator){                        //When the duration of the signal has been calculated, we print this value using the serial function.
   Serial.println(newThrottleSignalValue);
   // set this back to false when we have finished
   // with newThrottleSignal, while true, calcInput will not update
   // nThrottleIn
   newSignalIndicator = false;                  //We reset the boolean of the newSignalIndicator to not be stuck in the loop.
 }
}

void interruptRoutine(){
  if (digitalRead(THROTTLE_SIGNAL) == HIGH){    //When pin 2 is HIGH, we start the Interrupt.
    startPeriod = micros();                     //Used to return the number of microseconds at this time the Arduino begins to run the program.
  }
  else{
    if(startPeriod && (newSignalIndicator == false)){   //When the pulse is 0 on it's falling edge AND there's no new throttle signal, we calculate the pulse.
      newThrottleSignalValue = (int)(micros() - startPeriod);//This is done by subtracting the startPeriod with the current time read from the micros() function.
      startPeriod = 0;                          //Reset the startPeriod so it can count from 0 when called again on rising edge.
      newSignalIndicator = true;
    }
  }
```

Figure 6.2: *Decoding the transmitter signals.*

Running the software program and putting the stick of the throttle in the min-, neutral- and max position printed out the following values in figure 6.3.

```
17:29:52.422 -> 1004   17:29:10.178 -> 1500   17:30:20.940 -> 2000
17:29:52.422 -> 1004   17:29:10.178 -> 1500   17:30:20.940 -> 2000
17:29:52.462 -> 1004   17:29:10.218 -> 1500   17:30:20.982 -> 2000
17:29:52.462 -> 1004   17:29:10.218 -> 1500   17:30:20.982 -> 2000
17:29:52.502 -> 1000   17:29:10.258 -> 1500   17:30:21.022 -> 2000
17:29:52.502 -> 1000   17:29:10.258 -> 1500   17:30:21.022 -> 2000
17:29:52.502 -> 1000   17:29:10.298 -> 1500   17:30:21.062 -> 2000
17:29:52.542 -> 1000   17:29:10.298 -> 1500   17:30:21.062 -> 1996
17:29:52.542 -> 1004   17:29:10.338 -> 1500   17:30:21.062 -> 2000
17:29:52.582 -> 1004   17:29:10.338 -> 1500   17:30:21.102 -> 1996
17:29:52.582 -> 1004   17:29:10.338 -> 1500   17:30:21.102 -> 1996
17:29:52.622 -> 1004   17:29:10.378 -> 1500   17:30:21.142 -> 2000
17:29:52.622 -> 1000   17:29:10.378 -> 1500   17:30:21.142 -> 2000
17:29:52.662 -> 1000   17:29:10.418 -> 1500   17:30:21.182 -> 2000
17:29:52.662 -> 1000   17:29:10.418 -> 1500   17:30:21.182 -> 2000
17:29:52.702 -> 1000   17:29:10.458 -> 1504   17:30:21.222 -> 2000
17:29:52.702 -> 1004   17:29:10.458 -> 1500   17:30:21.222 -> 2000
17:29:52.742 -> 1004   17:29:10.498 -> 1500   17:30:21.262 -> 1996
17:29:52.742 -> 1004   17:29:10.498 -> 1500   17:30:21.262 -> 2000
```

Figure 6.3: *Output from left to right, Min-, neutral- and max position of the received throttle position.*

These numbers printed out in the Arduino™'s serial monitor, can immediately be related to the numbers in the min-, neutral- and max values from the oscilloscope measurements. The digital representation of the throttle's position values was 1ms, 1,5ms, and 2ms. These values are the same with the only difference that it is in microseconds in the output from the serial monitor.

6.2 Test run of the ESC and BLDC motors

Understanding how the BLDC motors will perform, a test program written. It can be seen in figure 6.4.

```
#include <Servo.h>
#define START_SPEED 30    //Start speed in order to arm the ESC.
#define SPEED 60          //Desired speed
Servo escSignal;


void setup() {
  escSignal.attach(12);
  escSignal.write(START_SPEED);
  delay(3000);            //Delay for arming the ESC
}

void loop() {
  escSignal.write(SPEED);
  delay(20);
}
```

Figure 6.4: *Test program of the ESC and BLDC motor. Attachment of the signal to the output pin on the Arduino™ board. The ESC gets armed, and a defined speed value will be sent to rotate the BLDC motor.*

To run the BLDC motor, they had to be armed through the ESC. This was done by defining a START_SPEED to a starting value, in this case, a value of 30, which would arm the ESC. After the initialization process, which was done with approximately a 3-second delay, the motor began to spin with the desired speed. Using the servo class was done because it has an easy to use PWM implementation that had more than enough functionalities for the BLDC motors purpose in this project.

After the first tests were successful, and different SPEED values were tested advancement to trying to control all 4 of the ESC and BLDC Motors were done. The code can be seen in figure 6.5.

```
#include <Servo.h>
#define START_SPEED 30    //Start speed in order to arm the ESC.
#define SPEED 60          //Desired speed.
Servo escSignal1;         //Representing signals send to each ESC.
Servo escSignal2;
Servo escSignal3;
Servo escSignal4;


void setup() {
  escSignal1.attach(12);  //Attaching the signal to the correspondi
  escSignal2.attach(11);
  escSignal3.attach(10);
  escSignal4.attach(9);

  escSignal1.write(START_SPEED);
  escSignal2.write(START_SPEED);
  escSignal3.write(START_SPEED);
  escSignal4.write(START_SPEED);
  delay(3000);            //Delay for arming the ESC.
}

void loop() {
  escSignal1.write(SPEED);
  escSignal2.write(SPEED);
  escSignal3.write(SPEED);
  escSignal4.write(SPEED);
  delay(20);
}
```

Figure 6.5: *Test program of all the ESCs and BLDC motors. Attachment of the signals to the output pins on the Arduino™ board. The ESC gets armed, and a defined speed value will be sent to rotate all the BLDC motors.*

7 Results

In this chapter, the project's results will be presented. The first part consists of the design results following the software results.

7.1 3D-Printed design

7.1.1 Platform V0.1

The platform v0.1 had implemented servo slots, BLDC slots, and enough room to fit the Arduino™ and a battery pack. With the truss technique implemented, the platform v0.1 might have been able to perform ground-borne movements if not for its malfunctioning servo motors. The SG90 motors were unreliable and problematic to control and were the main reason why the platform v0.2 was developed.

Platform v0.1 never got fully printed due to issues with the SG90, but it was the first big step towards a functioning multidimensional platform. The issues with the v0.1 and solutions to it would make the upcoming version, version v0.2, better.

7.1.2 Platform V0.2

The v0.2 version became the final version and the demo for this project. This version was designed based on solving the problems in version v0.1. Instead of designing the platform with the troublesome SG90 servo motors the version v0.2 was based on the Parallax Standard version servo motors and did not have similar problems as the SG90 with faulty motors. This platform version also had implemented propeller shields that were designed to protect the propellers when the platform did not use them. The propeller shields were symbolic and would not withstand heavy blows, but they were functioning as designed.

Once this version was assembled, software needed to be tested. The servomotors needed to be fully working and doing so without hurting its frame or fragile hardware. After having tested it in the own crafted test rig, the platform could successfully perform ground movements. It could move forward, backward and turning sideways by sequentially controlling the servos using applied PWM. Notations about its low friction claws and how it could hinder the platform from moving on certain surfaces were made and a plan to fix this initiated.

The solution to low-friction claws was to be shrinking rubber. By covering the claws in a sock of shrinking rubber, and applying heat to it, made the rubber ensnare and surround the claw in rubber with significantly higher friction than what the 3D printed plastic had.

Air movements were not possible with the hardware used. The positioning, or also referred to as phase, for the flight was implemented and tested, but it never took to the air. This mainly because of its weight in combination with weak BLDC motors. Due to expensive hardware and limited time, implementing air movement was left for future development.

The same goes for the water-borne movements. This was possible to simulate in the test rig by positioning the legs in certain ways pretending to be floating on water.

7.2 Software implementation

7.2.1 Initializing the servos into a neutral reference point

In order to control the servo motors to move to a specific degree, there was a reference program written. This program sets each of the servos into a 90-degree position. After the 90-degree position is initialized, the platform's parts were locked into place. This meant that every servo movement had to be calculated from a 90-degree point. The test rig can be seen to the right side of figure 7.1.

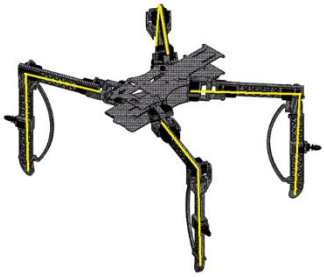
```
#include <Servo.h>

Servo servo[4][3];

const int servoPin[4][3] = {{2, 3, 4}, {5, 6, 7}, {8, 9, 10}, {11, 12, 13}};

void setup()
{
  for (int i = 0; i < 4; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      servo[i][j].attach(servoPin[i][j]);
      delay(20);
    }
  }
}

void loop(void)
{
  for (int i = 0; i < 4; i++)
  {
    for (int j = 0; j < 3; j++)
    {
      servo[i][j].write(90);
      delay(20);
    }
  }
  delay(1000);
}
```



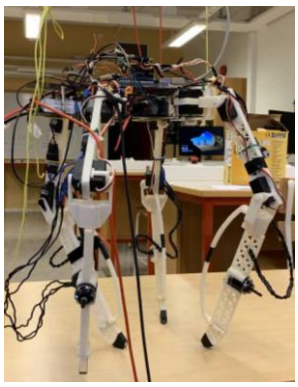


Figure 7.1: *Code and a representation of initializing each servo into a 90-degree position. Image of the test rig on the right side.*

7.2.2 Transform the platform into a ground-borne position

The process of transforming the platform into the ground-borne phase was done by implementing a ground-borne function, where every servo was positioned so that the desired shape appeared. Upon calling this function, the servo motors would always be set to the desired degrees so that this desired shape reappeared. This was called the ground-borne phase. Can be seen in figure 7.2

```
void groundborne()
{
    servo[0][0].write(90);
    delay(20);
    servo[1][0].write(90);
    delay(20);
    servo[2][0].write(90);
    delay(20);
    servo[3][0].write(90);
    delay(20);
    servo[0][1].write(45);
    delay(20);
    servo[1][1].write(135);
    delay(20);
    servo[2][1].write(135);
    delay(20);
    servo[3][1].write(45);
    delay(20);
}
```

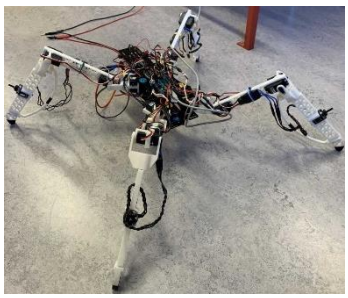


Figure 7.2: *Code and a representation of servos in ground-borne position.*

7.2.3 Transform into an air-borne position

As mentioned in the ground-borne transformation, servos got instructions to transform the platform into a specific shape. During the air-borne phase, the servos got instructed to transform the platform into a shape that could fly. Can be seen in figure 7.3

```
void airborne(){
    servo[0][0].write(90);
    delay(20);
    servo[1][0].write(90);
    delay(20);
    servo[2][0].write(90);
    delay(20);
    servo[3][0].write(90);
    delay(20);
    servo[0][1].write(20);
    delay(20);
    servo[1][1].write(160);
    delay(20);
    servo[2][1].write(165);
    delay(20);
    servo[3][1].write(10);
    delay(20);
    delay(2000);
}
```

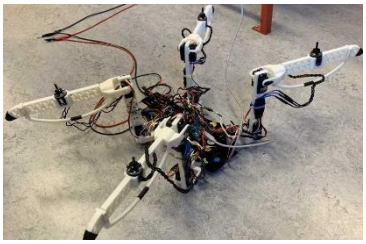
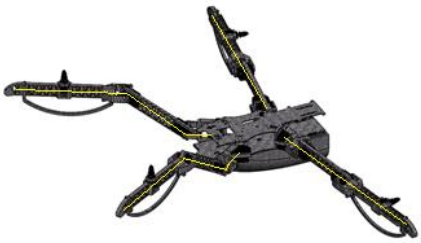


Figure 7.3: *Code and a representation of servos in air-borne position.*

7.2.4 Transform into a water-borne position

As mentioned in the previous transformations, servos got instructions to transform the platform into a water-borne position. Can be seen in figure 7.4

```
void waterborne(){
    servo[0][0].write(70);
    delay(20);
    servo[1][0].write(110);
    delay(20);
    servo[2][0].write(70);
    delay(20);
    servo[3][0].write(110);
    delay(20);
    servo[0][2].write(125);
    delay(20);
    servo[1][2].write(55);
    delay(20);
    servo[2][1].write(100);
    delay(20);
    servo[3][1].write(80);
    delay(20);
    servo[2][2].write(25);
    delay(20);
    servo[3][2].write(155);
    delay(20);
    delay(2000);
}
```



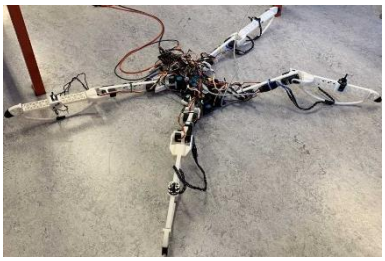


Figure 7.4: *Code and a representation of servos in water-borne position.*

7.2.5 Incorporate ground-borne movements

The forward motion function, as seen in figure 7.5, is programmed in a leg sequence referred to as the theory of a spider's movement. This sequence is as follows 1, 4, 2, and 3. After the first leg and before the last leg there's an implemented function that's called "bodyForward". This function lounges the platform forward by quickly setting each servo that's connected to the Coxas to its maximal back position. Measurements were made at a 3 meters distance and an estimated time for travel was roughly 25 seconds. This result varied based on the claw's friction to the ground.

```
void moveForward(unsigned int step){
  while (step-- > 0){
        servo[0][1].write(15);  //Leg 1
        delay(200);
        servo[0][0].write(50);
        delay(200);
        servo[0][1].write(75);
        delay(200);
        bodyForward();          //Body forward
        delay(200);
        servo[3][1].write(15);  //Leg 4
        delay(200);
        servo[3][0].write(110);
        delay(200);
        servo[3][1].write(45);
        delay(200);
        servo[1][1].write(165); //Leg 2
        delay(200);
        servo[1][0].write(130);
        delay(200);
        servo[1][1].write(135);
        delay(200);
        bodyForward();          //Body forward
        delay(200);
        servo[2][1].write(165); //Leg 3
        delay(200);
        servo[2][0].write(50);
        delay(200);
        servo[2][1].write(135);
        delay(200);
    }
}


void bodyForward(){
  servo[0][0].write(110);
  servo[1][0].write(50);
  servo[2][0].write(100);
  servo[3][0].write(60);
}
```

Figure 7.5: *Code parts of the ground-borne movement.*

7.2.6 Implement a wireless controller for the transformation phases and movements.

The implementation of the wireless controller was successful. Phase transformations were connected to one of the channels, giving the option to transform when switched on. The wireless range was tested on approximately 50 meters. This was done through multiples of walls and structures.

8 Discussion

In this chapter, the discussion of the project's results will be presented. The first part consists of discussion towards the project's results following the discussion of future implementation.

8.1 Discussion towards the project's results

Due to issues with some of the hardware used during the project, the expected order of which the project was supposed to be executed got changed. Some of the unexpected issues occurred in the development of the ground-borne movements for the smaller SG-90 demo spider. This demo can be seen in chapter 5, figure 5.1 above.

With the second version, v0.2, of the own designed platform; there were problems fitting some of the hardware in the printed model. By enlarging and reprinting some of the parts in this platform, this issue was resolved.

All and all the platform performed movement and transformation with 12 servomotors. Each of these servomotors had three active wires connected to them. The wires, if not managed correctly, could create a magnetic field that, in turn, could cause hardware malfunction and signal disturbance. In this project, this was never a problem that was detected, but despite that, precautions were made and the wires correctly managed.

The platform had issues with low friction between the rounded plastic claw, and the ground it stood on. This created difficulties in performing ground-borne movements without slipping. A rubber film, to increase the friction on the low-friction claws, was applied but it was despite that not optimal. By increasing the contact area of the claw or make it sharp and pointy could give a more desired result.

The transformation between the ground-borne, the air-borne, and the water-borne phase was successfully implemented. This project mainly focused on the ground-borne phase with implemented ground-borne movements. However, the air-borne and water-borne phases were created without implemented movements. These phases were created to symbolize the possibility for the platform to transform into other phases and perform various movements corresponding to what challenge it was facing. The implementation of the wireless controller allowed for the platform to perform movements without an onboard pilot and therefore without risking that person's life.

8.2 Future implementations

The platform can unlock even greater potential with further development and modification of it. Below are ideas for future implementation listed and explained.

8.2.1 Ground-borne phase with wheels

By adding wheels to the platform would give it the ability to move faster while in a ground-borne phase. By positioning the legs in a way that its propellers are creating a forward force instead of an upward force lateral movement could be achieved. By also adding a waterproof hull, where the wheels could be attached, the platform can cross shallow water or puddles with this movement technique See figure 8.1.

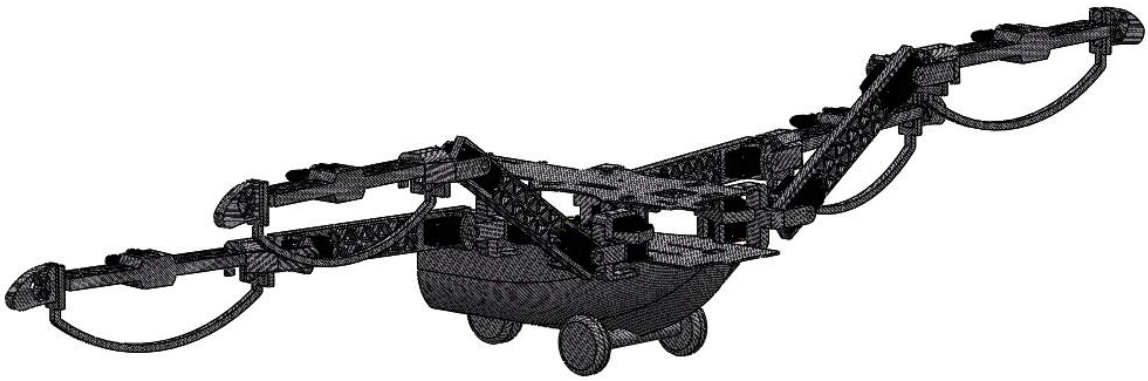


Figure 8.1: *Concept of wheels and hull implemented for a faster ground-borne movement.*

8.2.2 Submerged phase

By implementing a waterproof shell that protects the core of the fragile platform and gives it a more hydrodynamic shape, it could attain a submerged phase. This submerged phase would allow for the platform to gain the characteristics of a submarine and transport itself underwater. Positioning both the front legs and back legs closer together to reduce hydrodynamic forces on the platform. The front legs BLDC motors would work as active stabilizers keeping the platform in the upright position. The BLDC motors on the back legs would, in turn, act as rudders to turn the platform in the desired direction and at the same time create its forward force. See figure 8.2.

Figure 8.2: *Concept of the waterproof platform in submerged phase*

8.2.3 Propeller shield

The current version of the propeller shield in this project is only a symbolic cover. This could instead be a fully closing shield that wraps around the hardware itself. Implementing this would make it more enduring and significantly reducing the risk of getting a broken propeller or damaged BLDC motor. This would elevate the platform in the direction of becoming completely maintenance-free.

8.2.4 Self-sustained platform

Making the platform energy self-sufficient is also a necessity to make the platform maintenance free. An implemented solar panel could provide energy self-sufficiency. The solar panel can have various shapes and appearances. The two solar panel designs this project touched were a leaf shutter lens design and a folding fan design. The leaf shutter lens design has its design from a shutter lens that is mainly used in camera objectives. This design would provide a much more sufficient and durable protection but would require more space to implement and also less area for solar panel stripes to fit.

By comparing the leaf shutter lens design with the folding fan design, the latter should be more efficient due to the larger area for solar panels but also more fragile. Another option is to incorporate the solar panels with the previously mentioned idea of adding a shell that encloses the core of the platform. This would allow the platform to charge even though the sun is sparsely distributed or when the platform is moving on uneven surfaces due to the solar panel covering multiple angles.

8.2.5 The mini-drone implementation

Implementing a small drone on the platform that, by utilizing a raspberry pi and a small camera, could scout upcoming terrain. After its scouting mission, it returns to the platform with useful information. Based on the information gathered from video images and the data on traveling time and distance, a decision could be made on which direction and type of movement are best suited for the path ahead. This to minimize energy consumption and to avoid upcoming dangers. See figure 8.3.

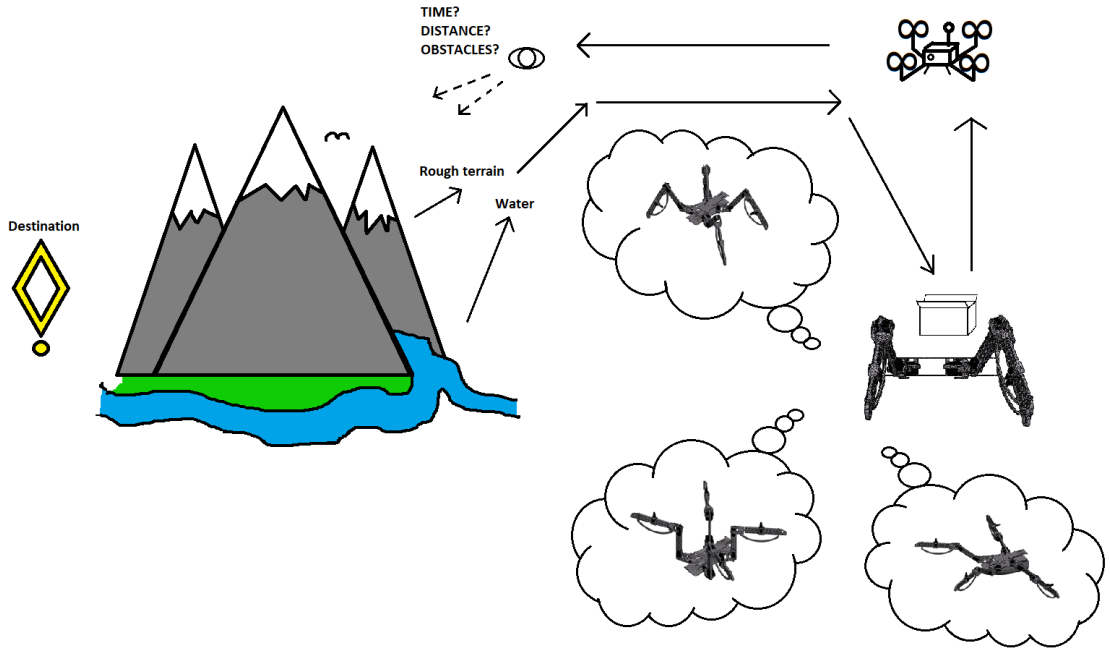


Figure 8.3: *Illustrates the main idea of platform usage.*

9 References

[1] Brodmann Maeder MM, Basnyat B, Stuart Harris N, “Wilderness & Environmental Medicine Volume 25, Issue 2: Empowering Rescuers and Improving Medical Care in Nepal”, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1080603214000040 [Accessed 14 05 2019].

[2]
Sun T, Xiang X, Su W, Wu H, Song Y, “A transformable wheel-legged mobile robot: Design, analysis and experiment”, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889017303081 [Accessed 28 05 2019].

[3]
Mendoza-Soto JL, Alvarez-Icaza L, Rodríguez-Cortés H, “Constrained generalized predictive control for obstacle avoidance in a quadcopter”, 2018. [Online]. Available: https://www.cambridge.org/core/services/aop-cambridge-core/content/view/AE33117272E2C270363B05C1FFA0EC55/S026357471800036Xa.pdf/constrained_generalized_predictive_control_for_obstacle_avoidance_in_a_quadcopter.pdf [Accessed 28 05 2019].

[4] Parry DA, ”Spider Leg-muscles and the Autotomy Mechanism”, Volume s3-98 [Online]. Available: http://jcs.biologists.org/content/joces/s3-98/43/331.full.pdf [Accessed 28 05 2019].

[5] Donald N,“Build Your Own Quadcopter: Power Up Your Designs With The Parallax Elev-8”, 2014. [Online]. Available: https://www-accessengineeringlibrary-com.proxy.lib.chalmers.se/browse/build-your-own-quadcopter-power-up-your-designs-with-the-parallax-elev-8#fullDetails [Accessed 25 05 2019].

[6] Auawise, Jrvz, “Yaw Axis”, 2010. [Online] Available: https://en.wikipedia.org/wiki/Aircraft_principal_axes#/media/File:Yaw_Axis_Corrected.svg [Accessed 28 05 2019].

[7] Khelifa M, Mordjaoui M, Medoued A, “An inverse problem methodology for design and optimization of an interior permanent magnetic BLDC motor” [Online] Available: https://www.sciencedirect.com/science/article/pii/S0360319917304342 [Accessed 28 05 2019]

[8] Couch II LW, Digital and Analog Communication Systems. 7th ed. p.213
ISBN NR: 0131424920

[9] Glover IA, Grant PM, Digital Communications 2nd ed. p.164
ISBN NR: 0130893994

[10] Arduino™, ”PWM”, Arduino™. 2019. [Online] Available: https://www.arduino.cc/en/Tutorial/PWM [Accessed 28 05 2019]

[11] Kankaria R, “Servo Motor: types and working principles explained”, 2018. [Online]. Available: https://engineering.eckovation.com/servo-motor-types-working-principle-explained/

[12] Arduino™, ”Arduino™ MEGA 2650 REV3”, Arduino™. 2019. [Online] Available: https://store.arduino.cc/mega-2560-r3 [Accessed 28 05 2019]

[13] HobbyKing™, “Manual of Sensorless Brushless Speed Controller”, HobbyKing™, 2013. [Online]. Available: https://p11.secure.hostingprod.com/@hobbypartz.com/ssl/ibuyrc/manual/07E-FLYFUN.pdf [Accessed 05 05 2019]

[14] Horton I, Beginning C. 5th ed. ISBN: 1430248815

A. Appendix

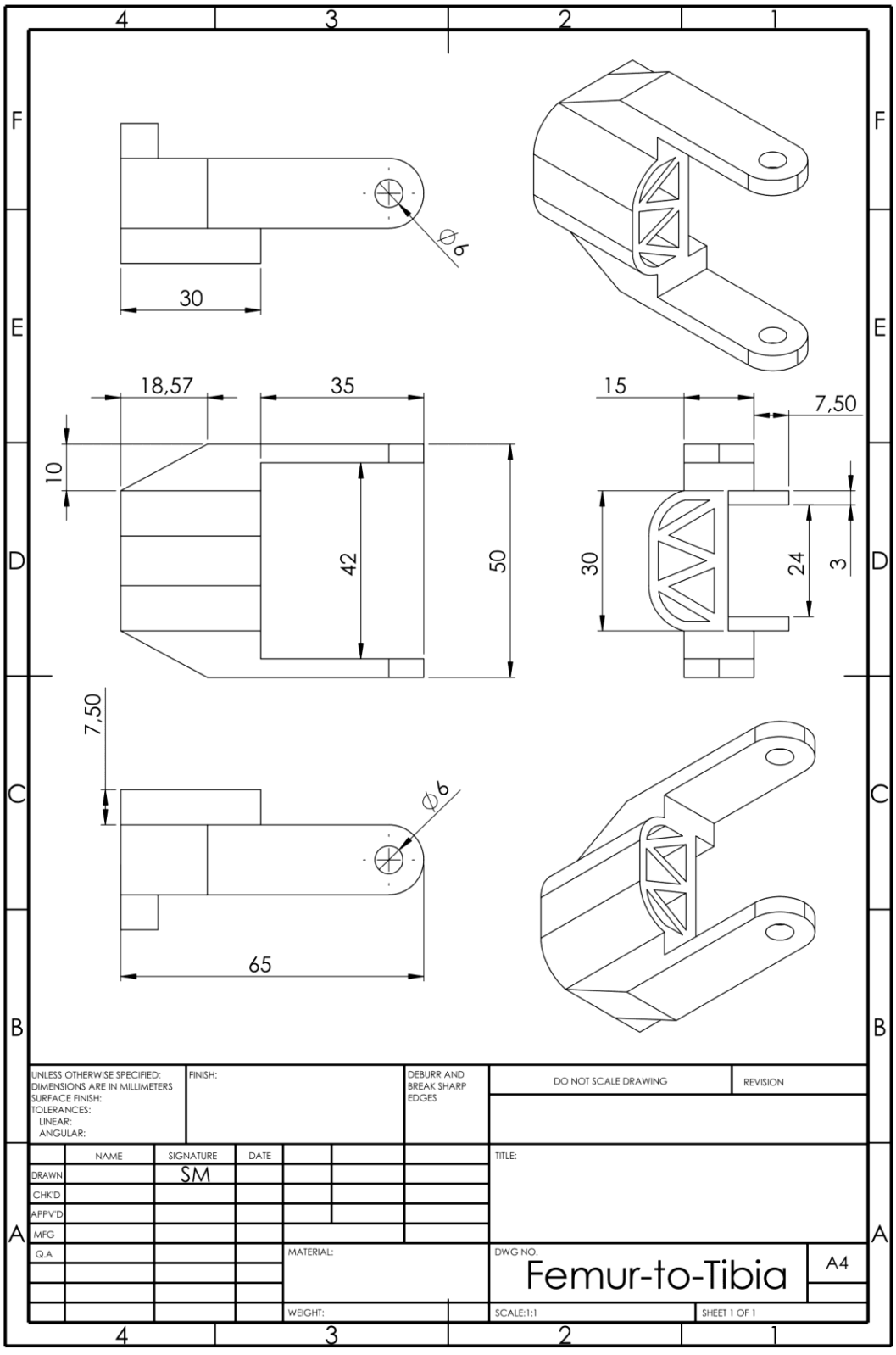


Figure A.1*: Drawing of Femur-to-Tibia.*

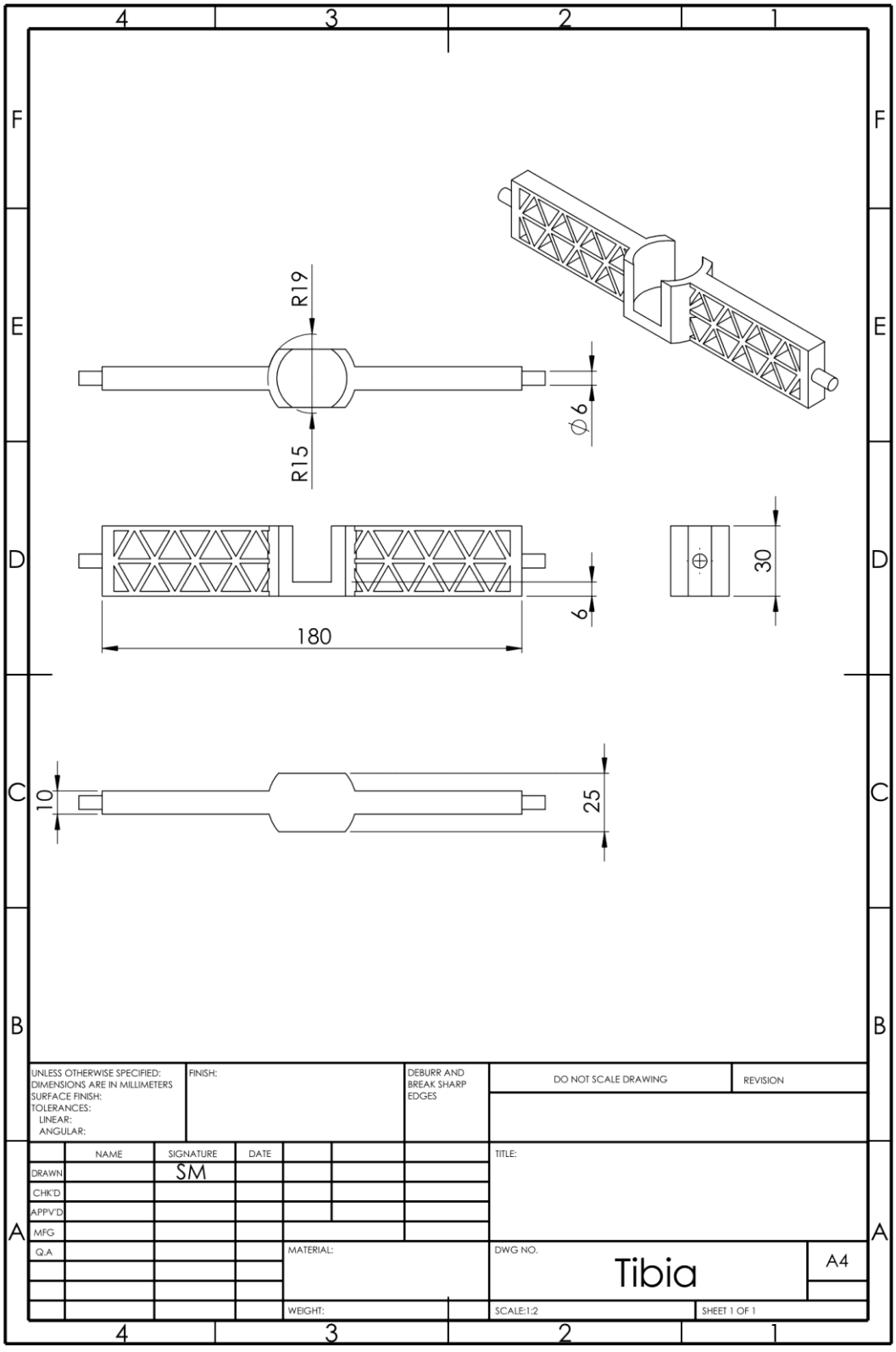


Figure A.2: *Drawing of Tibia.*

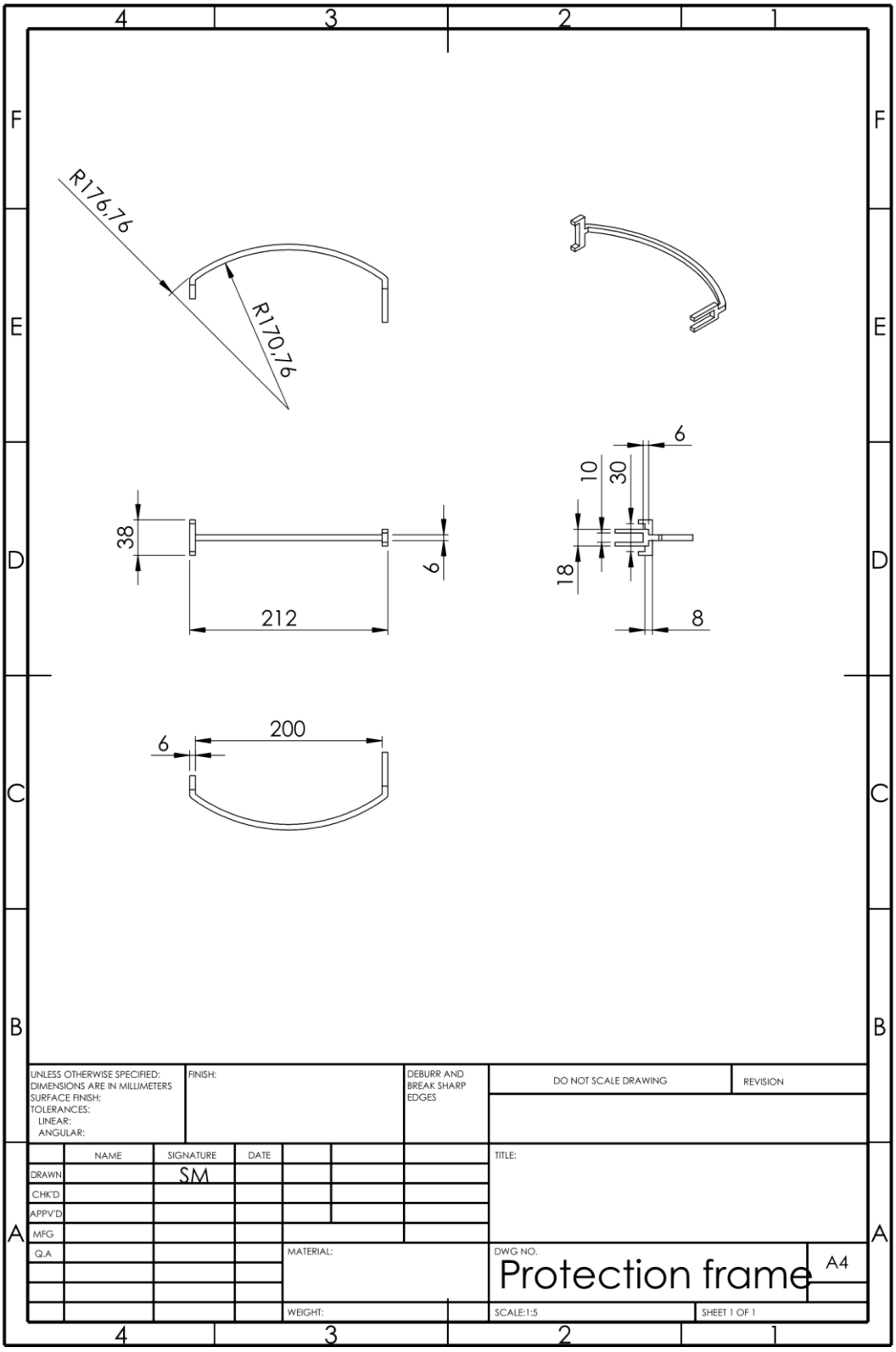


Figure A.3: *Drawing of Protection frame.*

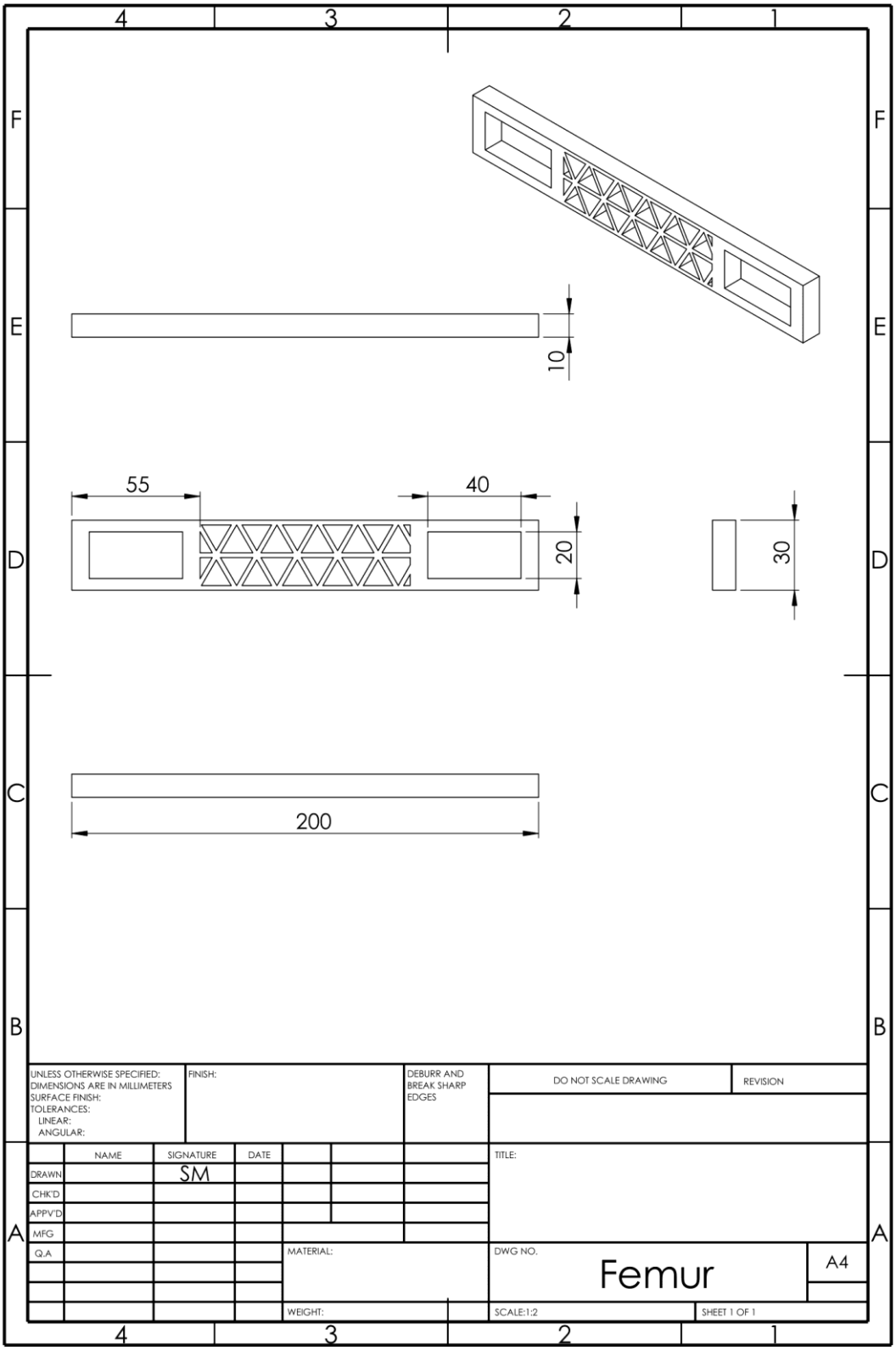


Figure A.4: *Drawing of Femur.*

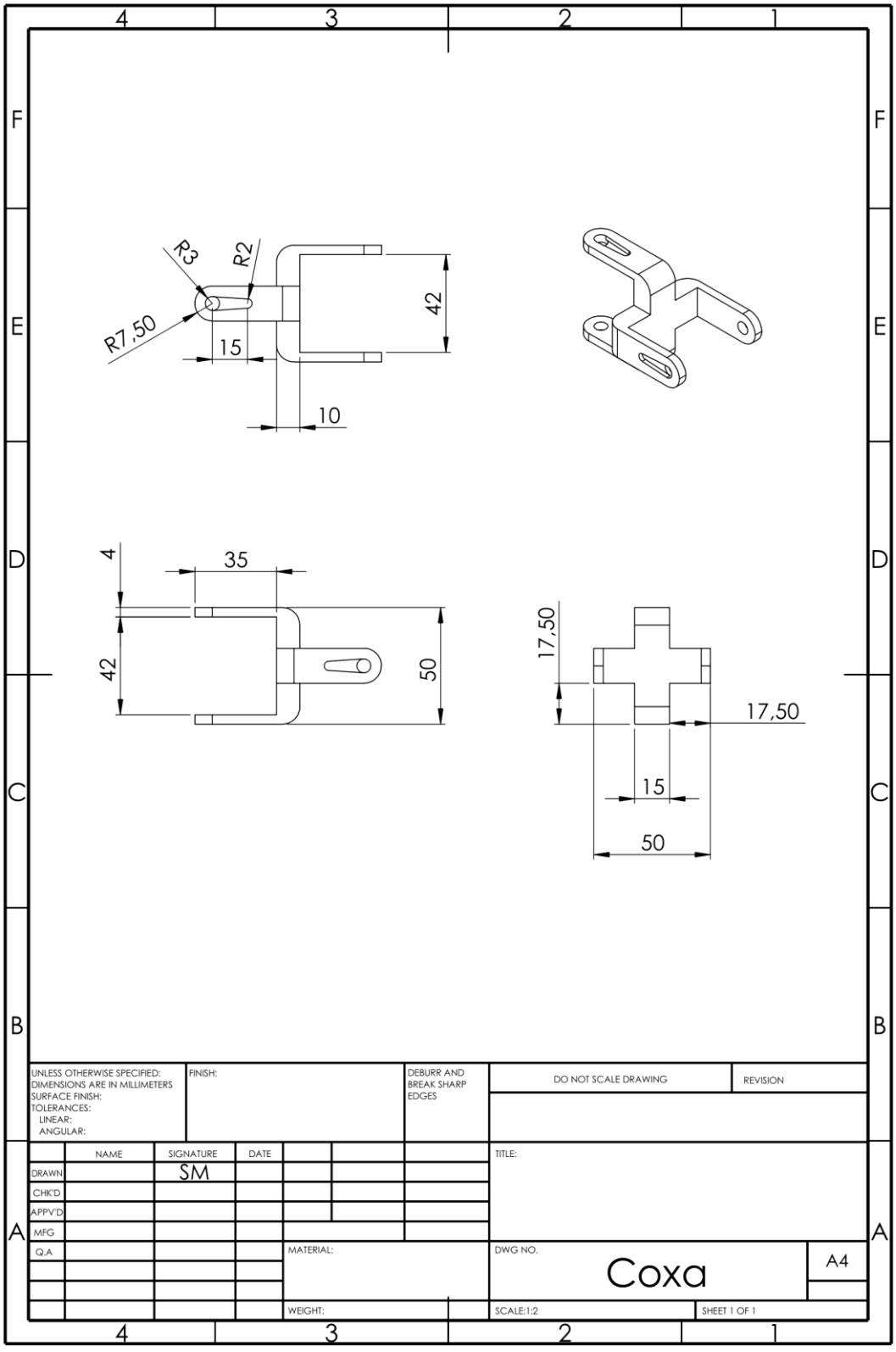


Figure A.5: *Drawing of Coxa.*

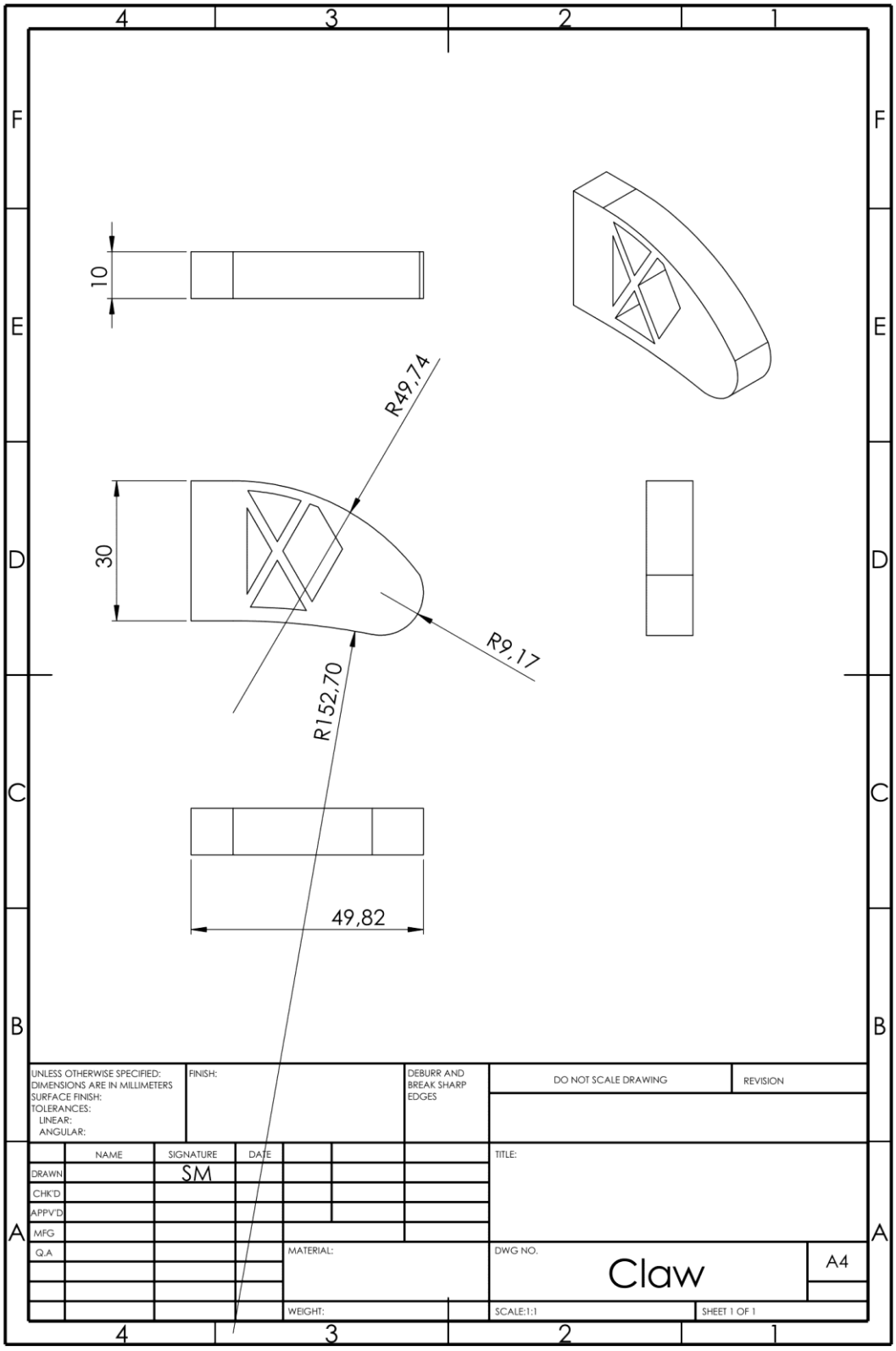


Figure A.6: *Drawing of Claw.*

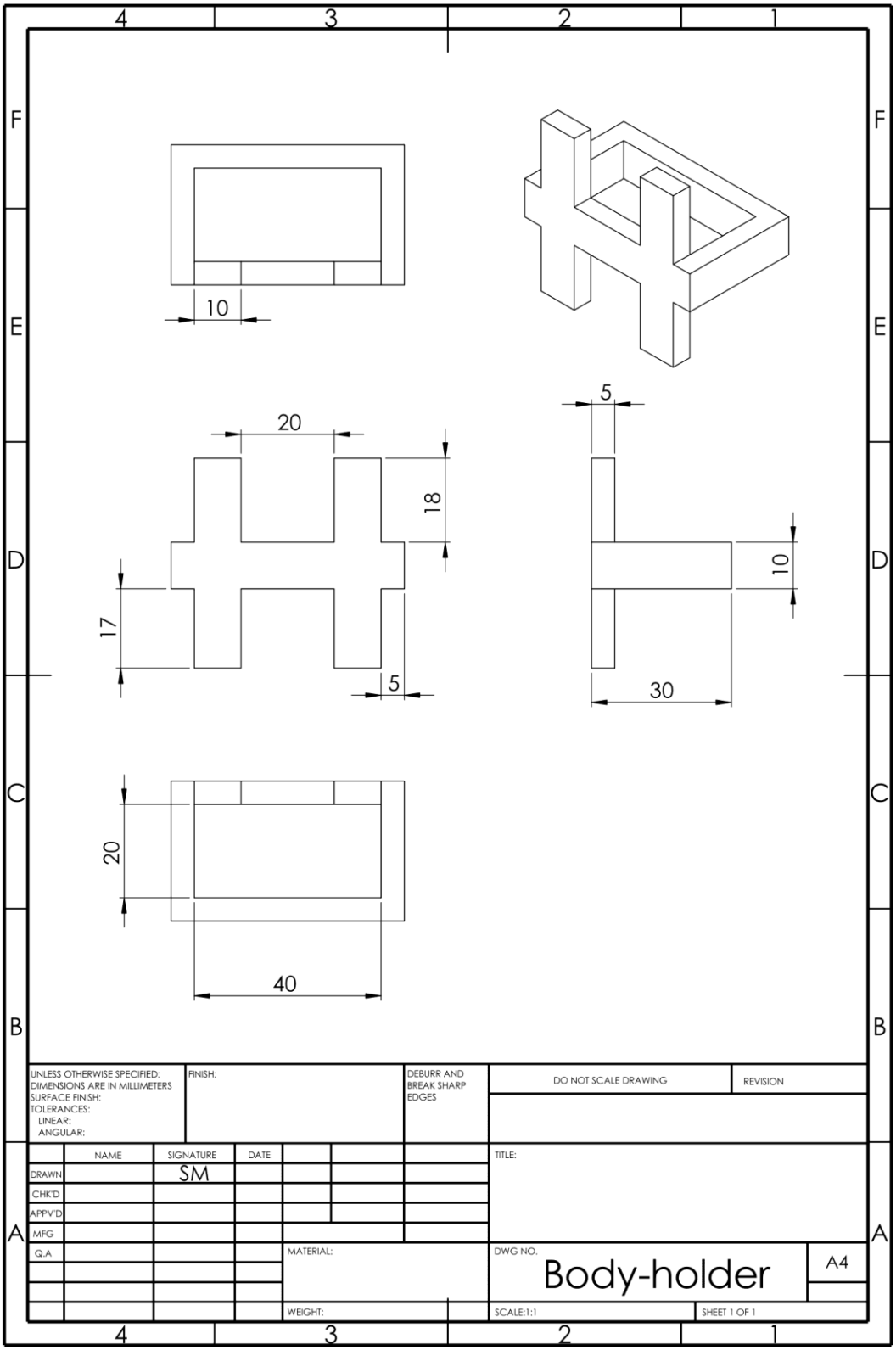


Figure A.7: *Drawing of Body-holder.*

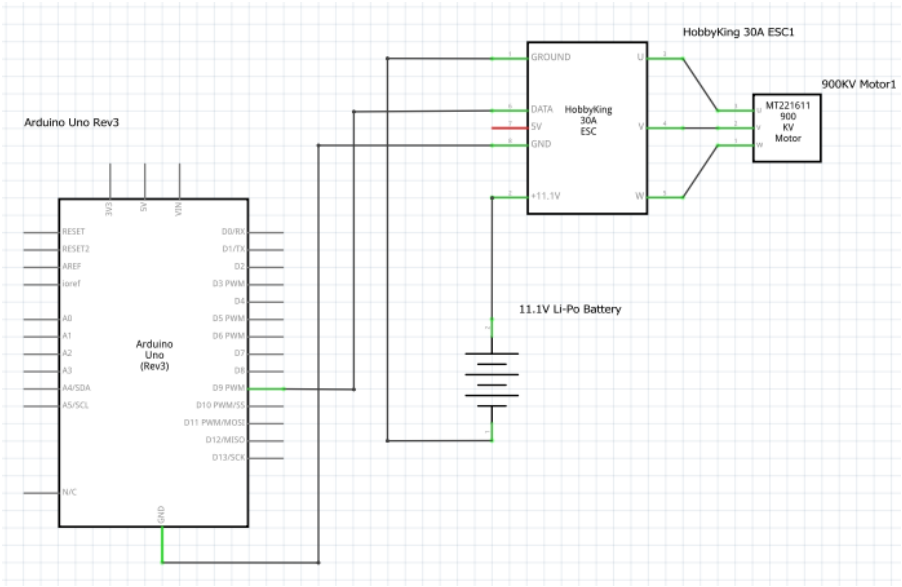


Figure A.8: *Schematics of BLDC-test.*

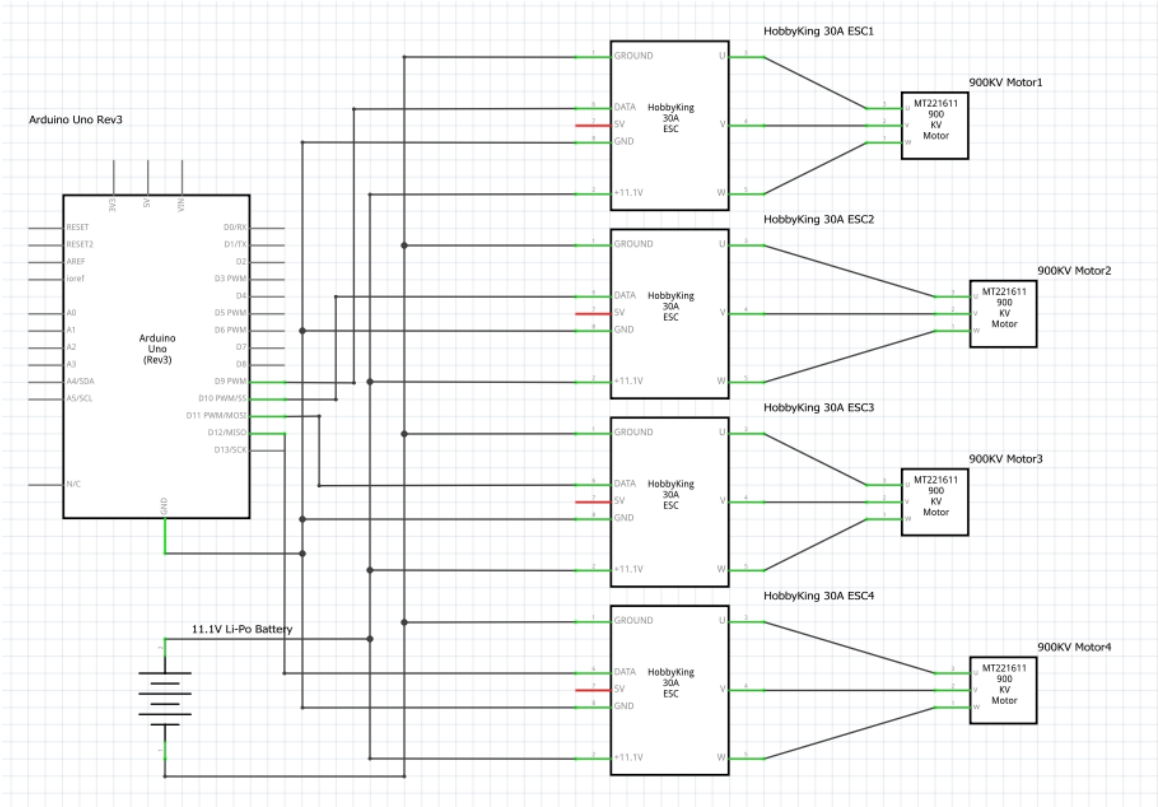


Figure A.9: *Schematics of all four BLDCs.*

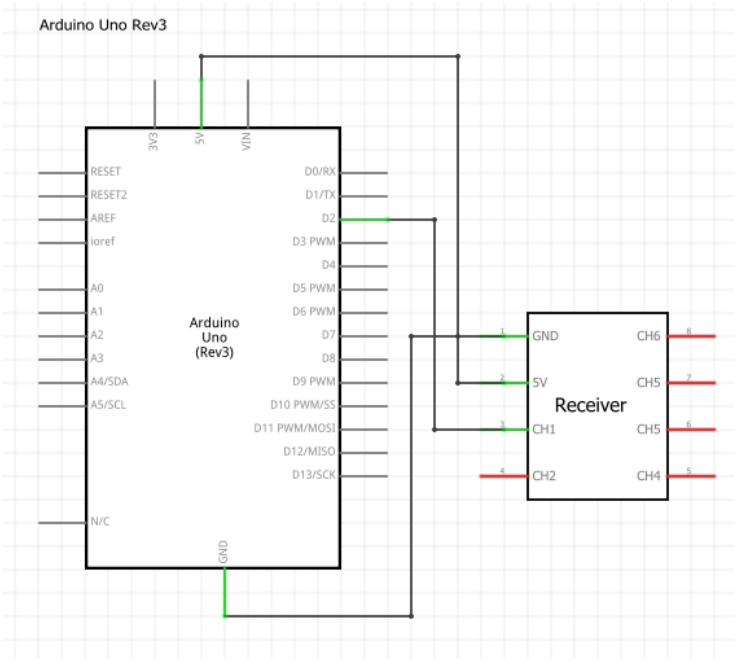


Figure A.10: *Schematics of receiver-test.*