S.T.E.V.E

Specific Terrain Evaluation Verification Equipment

A modular mobile sensor platform facilitating non-specialist assembly

By Joshua Graham Bowen Foulkes

Student number: 10617101

Abstract

This report aims to explore how to design a cost-effective mobile sensor platform for high-risk environments. Existing search and rescue platforms can be high cost and highly specialised, requiring specially trained operators. This report evaluates a robust mobile platform of my own creation that can continue to operate even with certain levels of damage and can be repaired locally. Because of the need for reparability, the large part of the construction technique used modular 3D printed parts and widely available hardware and software. The code was converted into an API, allowing the user to access data and configure the platform with ease. The results led to the conclusion of employing a spider's form for stability, movement, and adaptability. The design has survivability characteristics, capable of being repaired in the field. It can be upgraded and updated.

Key words: cost effective, mobile, sensor, reparability, modular, API, spider, survivability, easy to operate.

Acknowledgements

I would like to thank the following people who have helped me:

- The university of Plymouth for help and financial support
- My project supervisor for all his support.
- To my peers.
- To my parents for the support throughout my degree and putting up with me working from the kitchen table.
- To everyone who had to listen to the 3D printer endlessly going on.

Table of Contents

Introduction

This report presents the research and creation of a proof-of-concept modular all-terrain walking robot that can be used for search and rescue purposes.

The first 72 hours following a disaster are the most crucial to saving lives (Ross, 2019), and it is incredibly rare for people to survive for more than two weeks in rubble (BBC News, 2021). It can be difficult to get specialist equipment and operators to be deployed within that time and help to those in need.

A sensor platform is a robot with sensors that can be used and adapted for a wide range of uses, and additionally can have the ability of locomotion via legs, wheels or by some other means. Sensor uses can be for terrain mapping, search and rescue and EOD.

The investigation of my project starts with search and rescue robots.

- The iRobot 510 PackBot is the product that is currently one of the most used EOD bots and one of the most used search and rescue robots. This robot is strong, robust and can traverse many kinds of terrain.

Figure 1 FLIR PackBot 510 (Teledyne FLIR, 2021)

It includes minor modularity – it uses a standard controller supplied by the company. Although this is good inspiration for the creation of the design specification for my project, this product is only available at high cost and cannot be purchased by an individual. (Army Technology, 2021) (Built In, 2021) (Teledyne FLIR, 2021)

- The design features incorporated into the Elevate project car by Hyundai also offers a good insight.

Figure 2 Elevate project car by Hyundai (Ross, 2019)

This concept is designed to help search and rescue operators move across harsh terrain in the event of an accident or a natural disaster but could also be used for “walking” up to a wheelchair user’s front door.

The concept includes four robotic legs extending from the body, each legs has six degrees of freedom. These legs follow the terrain to keep the body stable and, therefore, the passengers. This project also includes interchangeable body parts that can be swapped depending upon the situation. (Ross, 2019)

My investigation continues into other walking robots.

- One of the most prominent robotics companies at present is Boston Dynamics: Spot robot dog.

Figure 3 Boston dynamics Spot robot dog (Boston Dynamics, 2021)

The Boston Dynamics' dog has a multitude of features: These include a 14Kg carry weight, 360-degree perception to map its environment, all-terrain movement, and navigation. Additionally, unique external hardware can be attached and integrated using mounting rails and payload ports. Unfortunately, due to its high complexity, the base model comes with a price tag of over £74,000. It also cannot be reproduced easily. (Boston Dynamics, 2021)

From these case studies, it can clearly be observed the advantages of using a walking design over regular wheeled design: Improved stability and all-terrain capabilities. Therefore, the design of a spider will be used as it is one of nature's best all-terrain experts. To help with the design, a scientific research article is used which graphs the spider's mechanism of locomotion over different terrain conditions. It also analyses the construction of a spider's leg. (Hao, et al., 2019)

- One of the better attempts at recreating a spider is the product created by Robugtix.

Figure 4 T8X robot spider by Robugtix (Robugtix, 2021)

This spider robot is designed mainly for entertainment purposes, it features eight legs. It uses them to move around, and it also has a camera for user feedback. However, this product is only designed with aesthetics in mind. This is highlighted by it utilising weak plastic servos meaning it is not very durable. It is able to coordinate four pairs of legs allowing for a wide range of movement, and terrain navigation. An application programming interface (API) has been incorporated which a user can modify to meet their needs. It does not allow for much modularity and expandability due to its design being fixed. (Robugtix, 2021) (Savage, 2013)

Each design has different characteristics to be included in the project. These offer solutions which will supply inspiration for the creation of the design specifications.

This research has enabled me to clarify my objective.

From these products available there currently are not many all-terrain sensor platforms in the market which can be created quickly and at a low cost, but also most of the current platforms require specially trained operators.

The purpose of my project will be to design a solution to meet the following criteria: Robust, cost efficient with the ability to be created locally, not requiring difficult manufacturing techniques and to include modularity. The latter will allow for it to be upgradable and modifiable by a user with limited knowledge of the product. There also needs to be an easy-to-understand graphical user interface (GUI) with recognisable controls, so that anyone can use it.

Project Objective and Specification

The project objective is to create a proof-of-concept all-terrain robotic spider base.

Servo driven legs will be used as this gives stability over rough, uneven terrain. The number of legs will need to balance cost, weight, current draw and lifting force. The number of legs affects its survivability as with more legs it is easier for the remaining legs to compensate if one fails.

The project will also need sensors. It will need sensors to detect when the leg is contact with ground. Additional sensors could be incorporated, an example being ultrasonic to stop collisions. The method for attaching the sensors needs to be expandable. An addition will be a camera which will allow for the operator to have visual feedback as well as to reduce collision risk.

The camera's stream and other relevant information needs to be displayed to the user. Therefore, a graphical user interface, GUI is needed. This interface also needs to include a way for the user to send commands and control the robot. The controls for the robot need to be recognisable and easy to understand.

The robot is required to communicate wirelessly with the GUI. Due to the robot being wireless it will be battery powered. The battery will supply power to the legs and to the ICs. There is a necessity for the circuit to be safeguarded to protect the battery and the ICs. The battery's voltage needs to be sent to the GUI; therefore, a voltage sensor is essential.

A requisite in construction is that it is modular. This will mean standardising the connections between the different components which will also allow them to be removed and replaced for a alternative component should there be a requirement for repairs or upgrade.

The construction material of the body and legs needs to be strong and robust to make the project survivable. It also needs to be low cost and easily manufactured. The chassis will have to be made from a material with similar properties, but it also will need to be lightweight to reduce the stress on the servos.

The servos selected should offer a balance between cost and torque, due to the large amount of servos. They should be at a reasonable cost but will have enough torque to lift and drive the body.

The code demands that it is easy to understand and well-structured, meaning separated files for some functions and classes. The main coding language to be used will be C++ as it is one of the fastest languages.

Design Philosophy

The research of existing solutions and the investigation of spider movement has formed the basis of these core design objectives and philosophy:

- Modularity

  The product should be fully modular allowing the user or manufacturer to change it to meet their needs.

- Simplicity

  The code should be simple to navigate, this can be achieved by making it into an API and having it fully commented. The design needs to be simple to understand so that it can be upgraded or repaired by someone with a limited knowledge of the product.

- Durability

  It is imperative that it is durable and survivable. Survivability is the ability to remain alive or continue to exist. This can be achieved by using durable materials for construction. This also means that the product can still function even with some of its components broken, such as being able to operate on six legs instead of eight.

- Ease of use but allowing for customisation by user

  The project may be used by someone with a limited knowledge of the product, an example of how this can be done is by using controls that are recognisable. A desirable feature will be that the product is upgradable by the user.

- Upgradability.

  The product should have capability so that it can be upgraded by user or the manufacturer later. This can be achieved by leaving some port space for additional servos or using I2C to control the sensors for simplicity.

- Easy to build and package

  Another desirable requirement is that not only is the product easy to build but also easy to package for transit to wherever it might need to be exploited. This will enable it to be built on site. The aforementioned modularity of the build contributes to this, as well as allowing for pieces to be manufactured and packaged rapidly. It can then be assembled using basic tools on location.

Such approaches would provide a product well suited for its desired purpose.

Actual design

My project has eight legs as this not only gives a large amount of stability but also means that if one leg is damaged due to extreme conditions in search and rescue, the others can compensate for it. This octopod makes it easier to navigate and allows stability on rough, uneven terrain. There was the option of only having six legs, and there were indeed some advantages to this. However, this took away from the aesthetics, and meant that if one leg breaks it would be more difficult for the others to compensate.

Sensors have been added to the bottom of the feet to detect for terrain contact. These sensors are simple buttons that can be upgraded for weight sensors in the future in order that stability of the footing can be more precisely measured. Additional sensors such as an ultrasonic could be added to stop object collision. The sensors are attached via SPI and/or I2C, meaning that more sensors can be added very easily by simply connecting them to power and the data lines. A USB web camera is an addition to the build for user feedback. The stream is sent over Wi-Fi, so a Raspberry Pi has been added for this purpose.

The GUI has been created using a software called Processing. This software has a host of libraries that can help with the creation of a GUI. It has basic data display, the stream from the USB webcam and a method for inputting commands for the robot to execute. An Xbox One controller has been introduced to control the robot, as it is recognisable and has libraries available to interface with it. The controller is also easily replaceable if it breaks. Feedback from the controller is displayed to the user via the GUI.

Bluetooth has satisfactory range, is cost effective and instructions on how to use it will be easily available online. The robot also offers the option to be battery powered. The battery powering of the servo and ICs are separate as they require different voltages and currents. A USB power pack might be an option for powering the ICs as they all have USBs. Two Adafruit INA260 circuits have been added to allow for the voltage to be read and therefore the battery percentage to be calculated. These circuits can also read the current. This information can be sent to the GUI and used as overuse meters, telling the user that the servos are under too much stress. Unfortunately, these circuits do not come with libraries for the Nucleo board – they are only available for an Arduino - therefore an Arduino was added which communicates with the Nucleo board via USART.

A standardised connection method needs to be strong and minimise movement but also must ensure for the piece to be easily removed should a replacement be necessary. This, therefore, excludes the use of glue. Instead, screws have been added to hold each piece in place, the advantage being that pieces can be removed whilst still allowing for a strong joint.

The leg has been made up of three servos as this allows for a large range of motion. A fourth servo may be added later to improve the robot's navigation and stability over rough terrain. The image below showing the structure of a spider's leg was used as inspiration for the design of the leg.

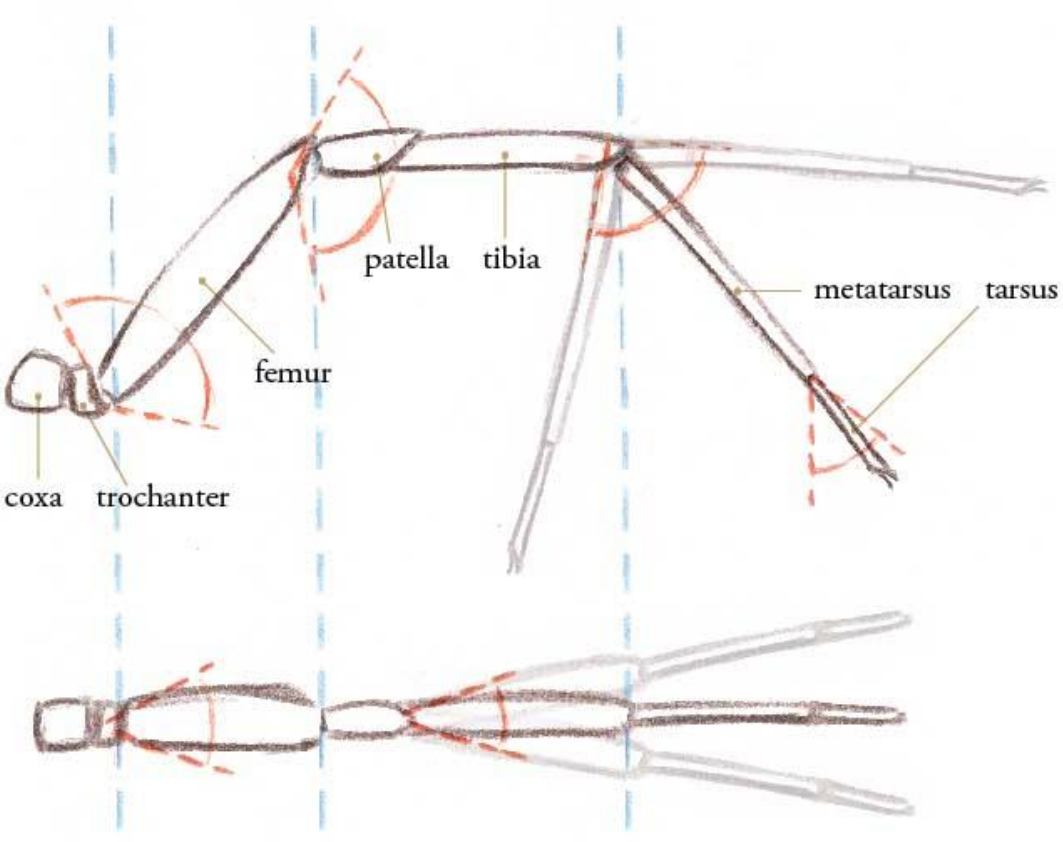


Figure 5 Side view and top view of a spider's leg showing range of movement (Laws, 2017)

Each leg has been manufactured using a 3D printer as the process is quick and readily available around the world. This manufacturing technique means that many materials could be introduced, including carbon fibre, allowing for the legs to be light and strong. Unfortunately, carbon fibre is expensive and requires special 3D printers. Standard 3D printing PLA is strong,

durable, and reasonably priced. The chassis was designed to be made from PLA. This proved too heavy so alternate materials have been investigated.

The servos selected are 996R servos with a torque rating of 15Kg/cm at 6V. These were used as they have decent torque but are also acceptably priced at £4 each. Stronger servos such as ones rated at 20Kg/cm cost around £20 each. To capitalise upon the servo's strength, the weight of the robot must be kept to a minimum.

To make the code easy to understand it has been split into separate classes and functions. The code has been commented, and sensible variable names used. The main coding language is C++ as it is one of the fastest languages and one of the best for servo control. It is also the language used by the Arduino. The movement has been based on a research paper which graphs the movement of a spider on multiple types of terrain. (Hao, et al., 2019)

Resources

The choice of hardware is important. Due to the need for it to be built quickly and easily, the components used are standard off-the-shelf components that can be easily ordered if damaged.

Most bespoke components are manufactured from a 3D printer, meaning that if these pieces are damaged, they can be easily replicated. The assembly only requires basic tools, and as no glue has been used, the pieces can be quickly replaced.

The software used is available to download and most of the software is open source.

Equipment: 3D printer, standard electronics equipment, home workshop tools.

Locations: University laboratory, library, personal workshop i.e., kitchen table and bedroom due to Covid restrictions.

Support & outside services: £120 funding from university.

Manufacturing: Personal 3D printer, online circuit board manufacturing, online component manufacturing.

Components: Approximately 20 servos connected to 2 PCA9685 servo driver boards, one for each side. The control is implemented by an Arduino or stm32 board, ultrasonic distance sensor has been used for distance measuring.

Procedures

Leg process

The first step was to choose the correct servo as this is the main driving force of the project. The servo had to meet the requirements of strength and cost. The choice was narrowed down to a 15Kg/cm servo that costs £4 per servo. Stronger servos such as 20Kg/cm servos cost around £20 each.

The initial leg design had the servos detached from the actual leg piece. A servo horn (dark blue) was attached to the servo (white), which would connect to an arm (red). This would attach to the leg with a fulcrum connected to the piece, allowing it to rotate. This was chosen as it allowed the design to be small and thin.

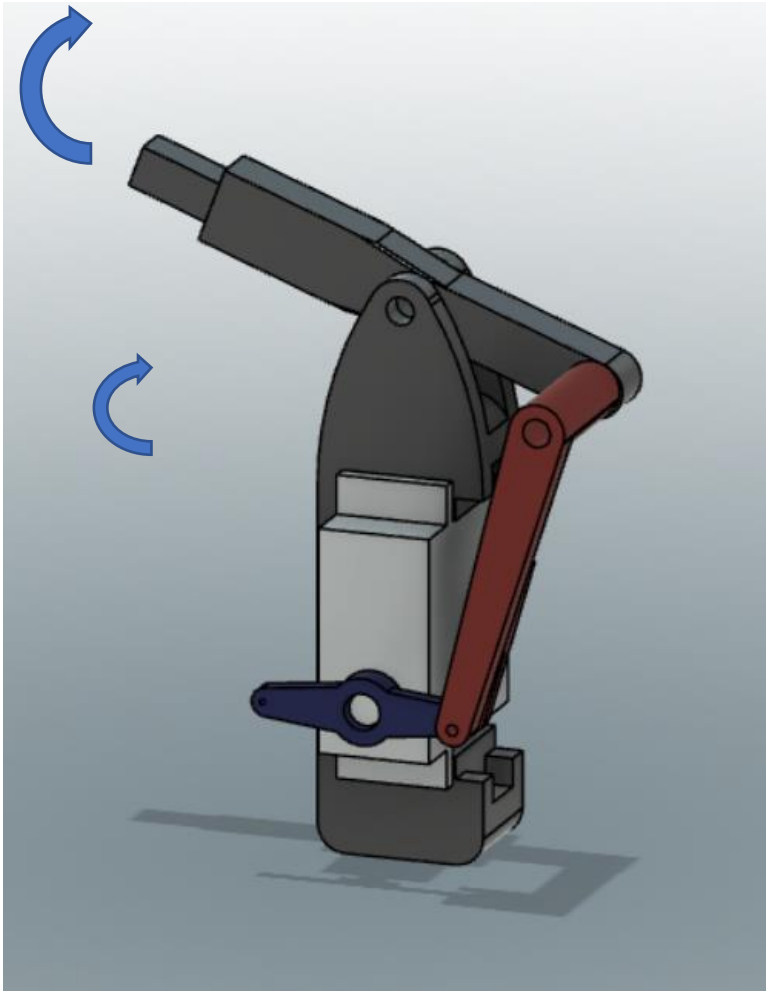


Figure 6 A 3D model of the old knee joint

The parts were then printed and assembled. This was when the first issue with the 3D printer arose. When a 3D printer prints a hole, the hole is slightly miss-shaped resulting in a piece of a planned diameter which would not fit into the designed hole: M3 bolt would not fit into a 3mm hole.

This led to the creation of a hole tester. This was a piece of plastic with holes in a range of sizes, offering the opportunity to test the hole sizes.

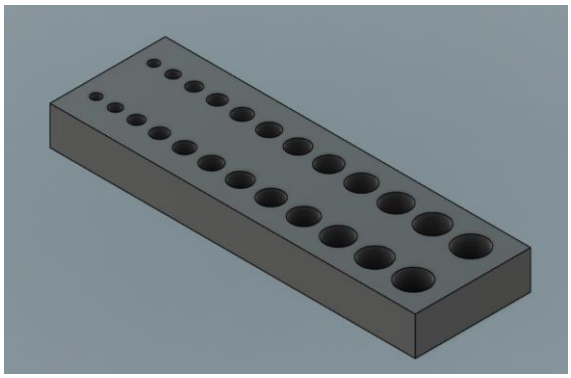


Figure 7 Hole tester

From testing, a hole needed to be 1mm larger than the bolt to achieve a clean fit. This information was utilised throughout the design process. Due to the sizes of the holes being fixed, other improvements were added, and the pieces were reprinted. Below is an image of the pieces once they were printed.

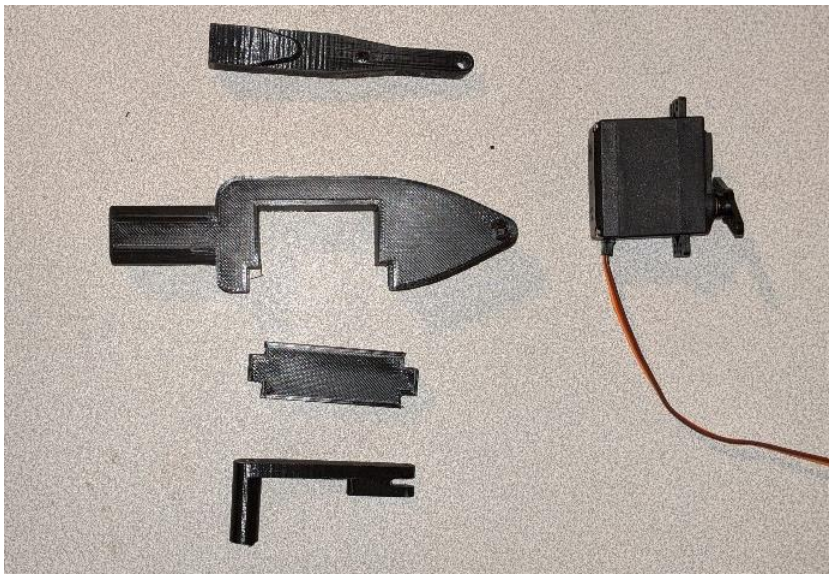


Figure 8 Printed knee pieces

Afterwards the hip was designed. The hip was complicated as it required two servos, one for sideways movement and one for the up and down movement. The design consisted of one servo doing the sideways movement and having the other servo attached on top controlling the up and down of the leg, with the leg attached to the same piece.

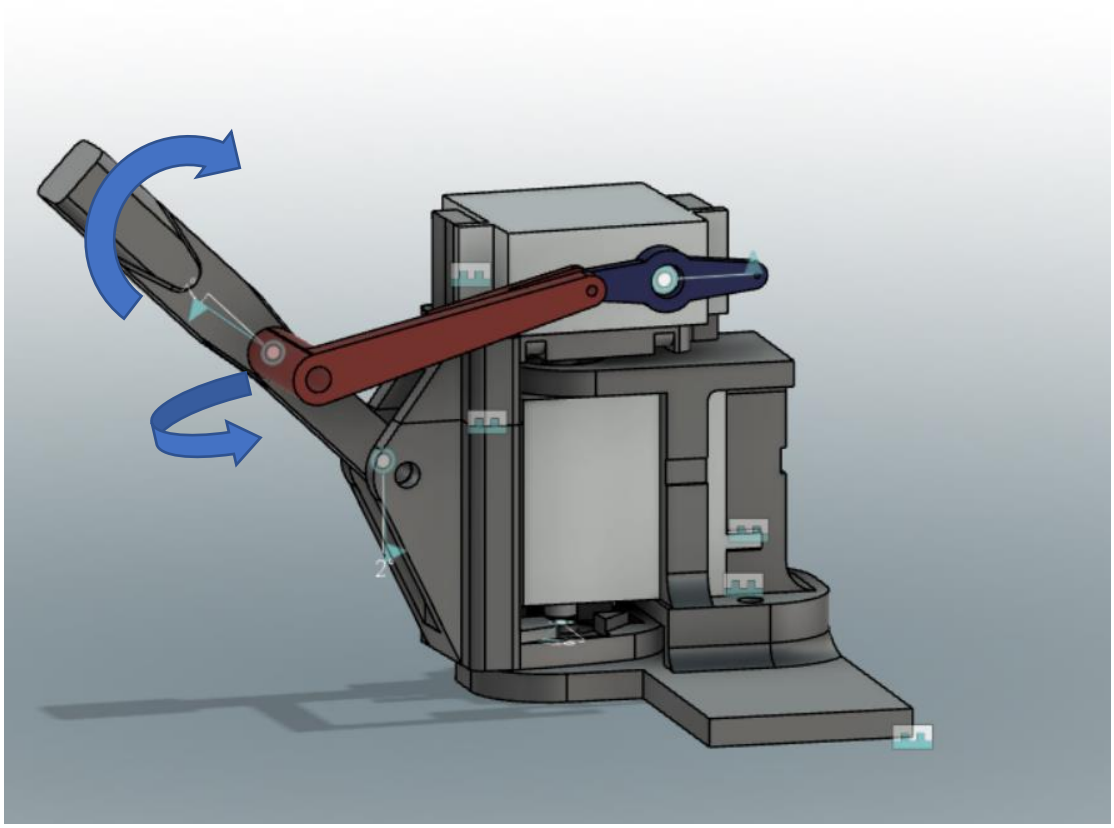


Figure 9 A 3D model of the old hip design

The base shown here is a version with extended sides to allow for it to be easily clamped for testing purposes.

These pieces were then printed and assembled.

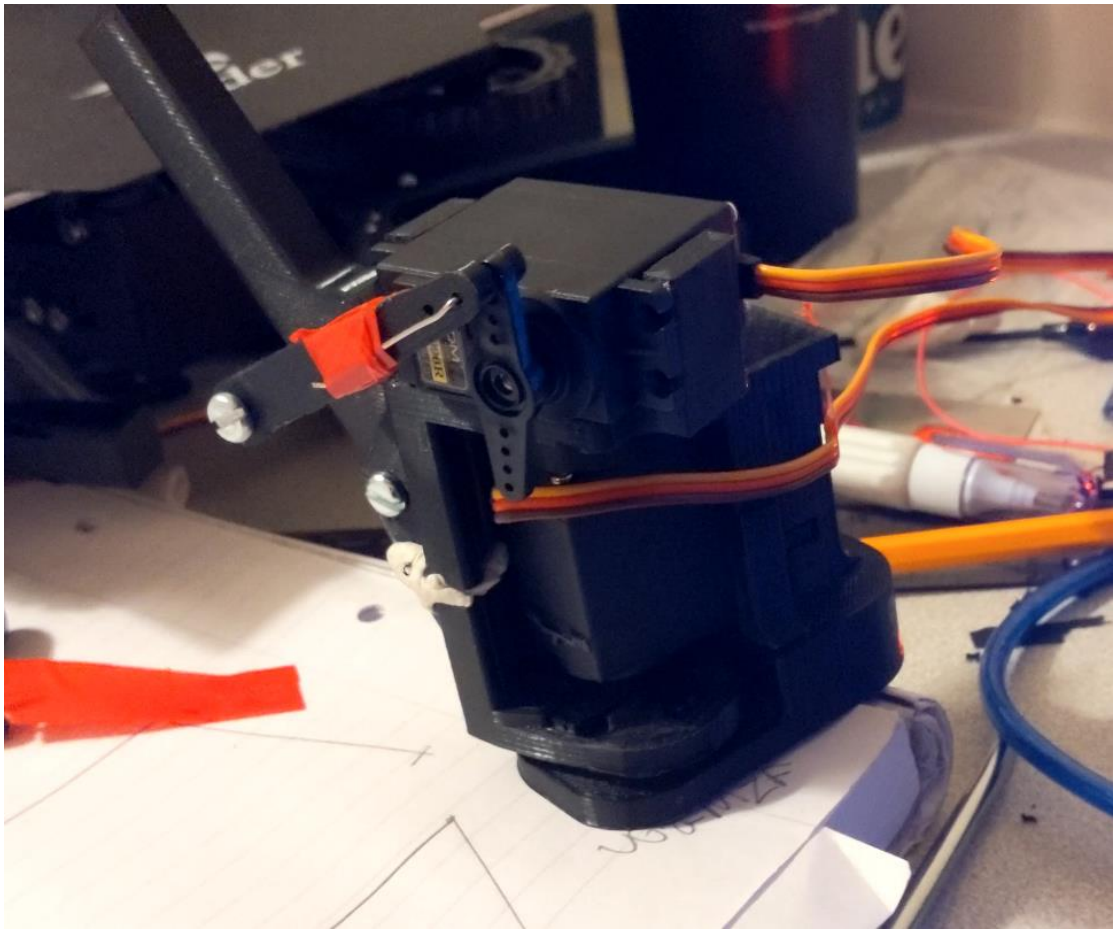


Figure 10 Assembled hip joint

Additional leg pieces were printed such an ankle, a foot, and a leg piece to attach it all together.

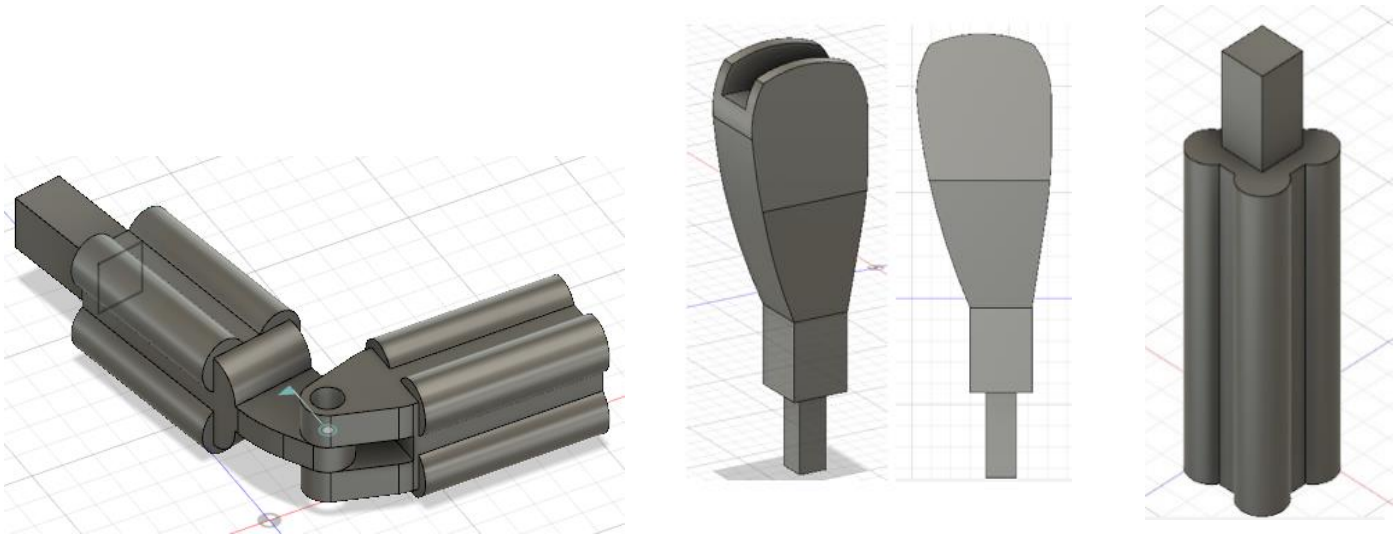


Figure 11 3D model of ankle, foot, and leg piece

This was where standardisation and modularity were introduced. The standardisation meant that if a piece were broken or faulty then only a small part had to be replaced. Future pieces, or upgrades, can be easily introduced, following the modularity design.

The design of the foot was complicated. It would be required to rotate with the movement but also have good grip on rough terrain. The design uses an RC car tyre as the grip material for the foot. An improvement to the design was to have the arc on the end of the foot slightly shallower than the arc of the tyre. The weight of the spider would compress slightly expanding the surface area in contact with the ground, increasing traction. This is the same principal used in off-road tyres on cars (Utires, 2017). These pieces were then printed and added to the design.

This was where another issue with 3D printing arose: The scaffolding created by the 3D printer when printing hollow objects. The scaffolding is created by the 3D printer to support the plastic when printing hollow pieces, to stop it from caving in. The issue is that it can be difficult to remove once the piece is printed.

This issue became apparent when printing the leg piece. The Gcode file of the leg piece was therefore reconfigured so that it would be printed standing up instead of sideways meaning that the supporting architecture in the hole for the square stability pin would not need to be removed. Shown below, with the new file being on the right:

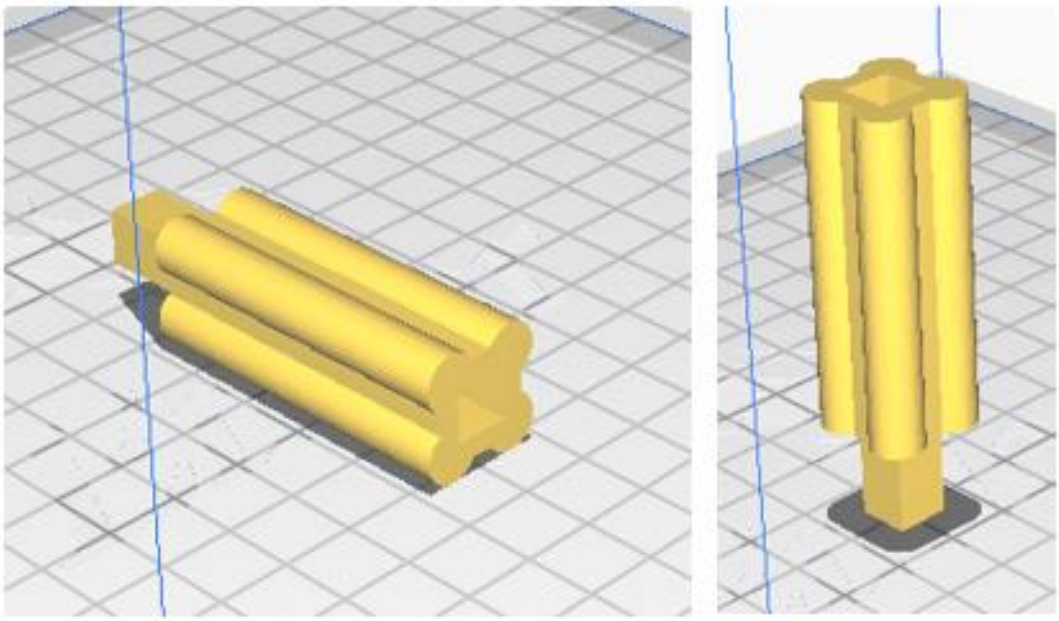


Figure 12 New leg piece print rotation

This created an issue where, due to the layers being one on top of each other, it meant that the stability pin was very weak and would break off as soon as any force is applied.
To remedy this, the stability pin was printed separately to increase strength, they were then assembled.

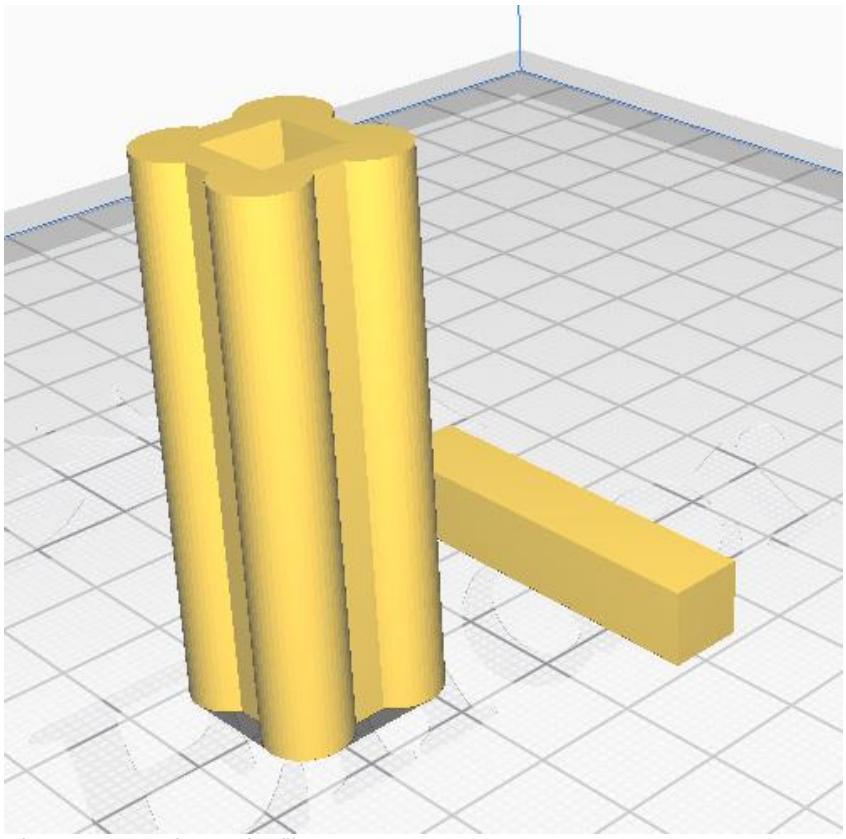


Figure 13 Leg piece print file

This issue was also apparent with the foot. Initially to remedy this a small pilot hole was incorperated for a screw to be added to the pin. This method worked but had some flaws where the pin would still break of due to rotational force causing the foot to stay connected but the pin to rotate freely. Another issue was that when adding the screw there was a high chance that it would break the stability pin. The best solution was to print the foot sideways. The only issue with this method was that the scaffolding plastic had to be dug out of the centre of the foot, increasing manufacturing time. A solution to this could be to reduce the fill percentage of the scaffolding, making it extremely weak, and there is a chance that the scaffolding would collapse whilst printing, ruining the whole print. The foot could be printed without scaffolding. However, this requires complex print settings which are time consuming and not always obtainable.

The components were then printed and assembled.

Figure 14 Fully assembled leg

The design of the leg had to be completely changed. Due to the connection arm being very flimsy, any compression force would break either the arm, the servo horn or the connections in between.

For the next leg version market research was done on the pre-existing connection between two servos. The one that looked the most promising was the metal connection below:

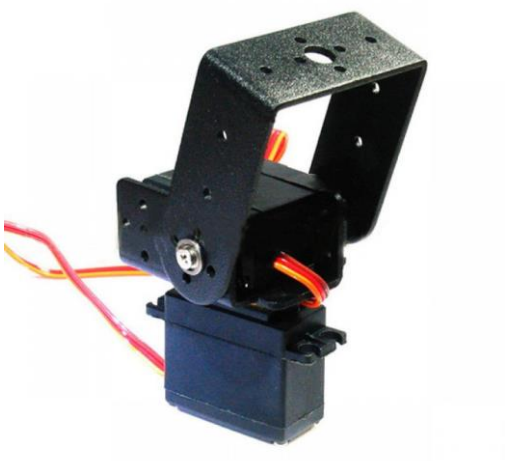


Figure 15 2 DOF Long Pan and Tilt Servos Bracket Sensor Mount kit for Robot Arduino compatible MG995 (ThanksBuyer, 2021)

This pre-made joint was the inspiration for the revised hip joint.

The main change with the new joint is that the leg is directly attached to the servos instead of via a connection arm, also the plastic servo horn was replaced with a metal one. These two changes reduced the number of weak spots. It also allowed for a greater range of movement for the servo.

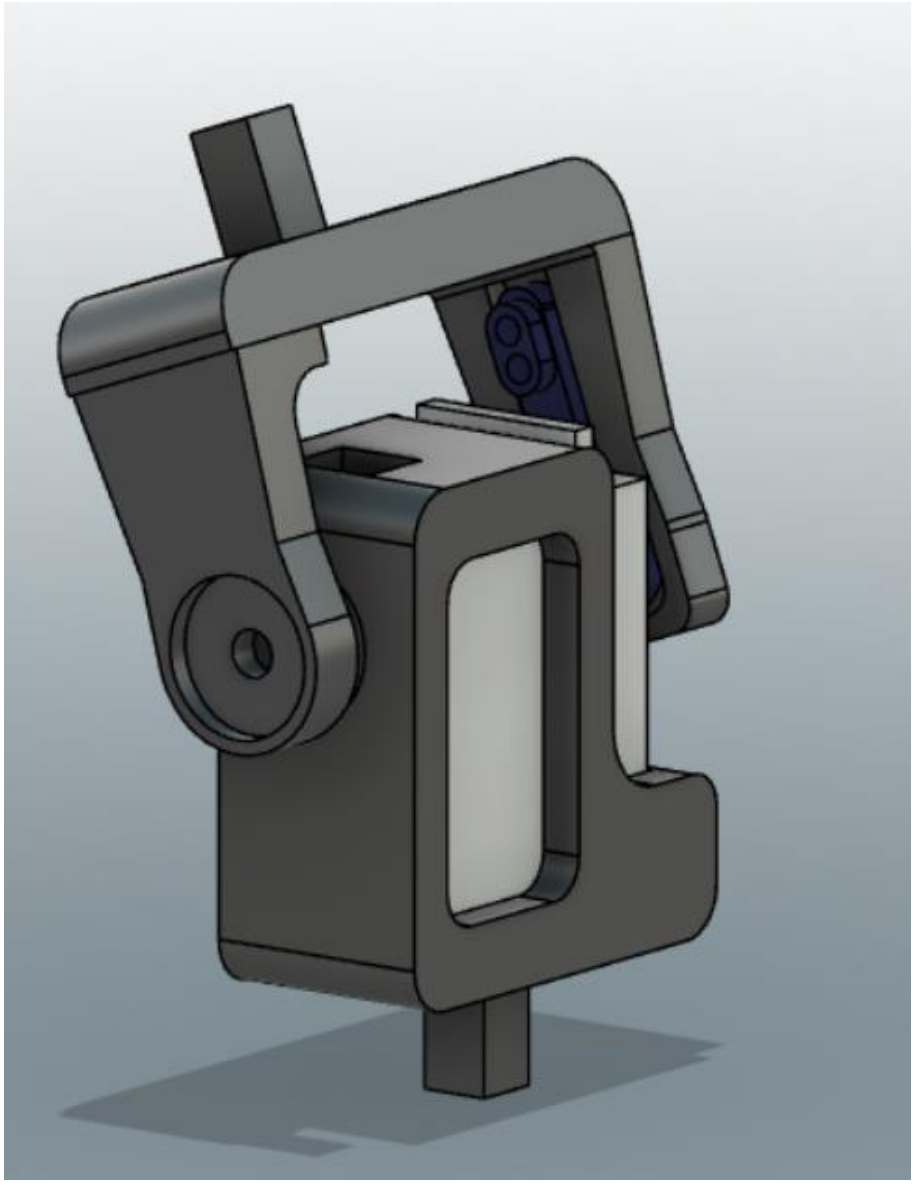


Figure 16 3D model of updated knee joint

Smaller improvements were made later to reduce its weight and to increase its strength. The modularity of the previous design is still used, meaning that if a version upgrade is carried out the older version can be easily swapped out.

The hip connection was also remodelled.

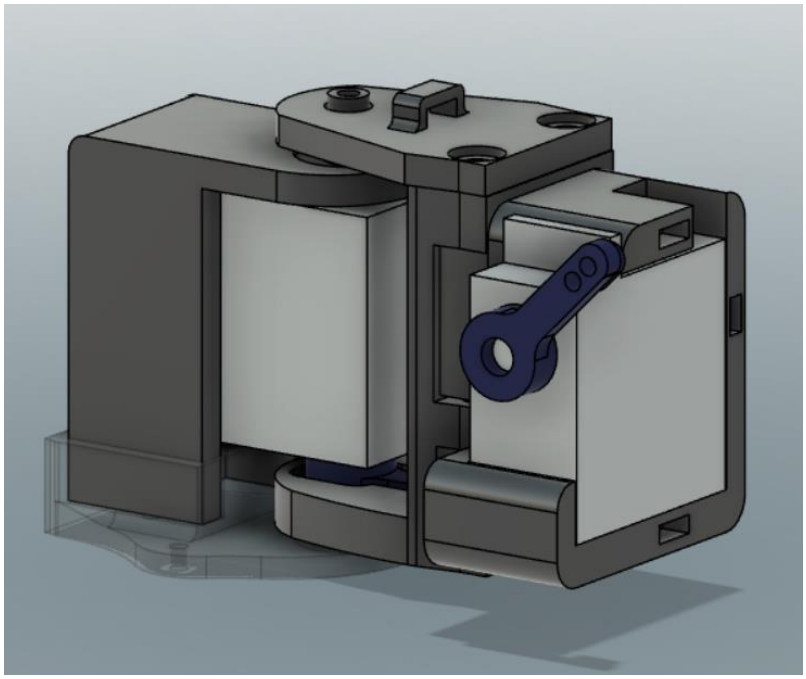


Figure 17 Remodelled hip joint

Some parts were changed whilst some were kept and improved upon. This new design is also shorter than the previous one. Some parts were printed sideways to improve the strength without increasing its weight. Some pilot holes were added to certain pieces so that a screw can be added which improves strength at construction.

The initial ankle design had a variable joint angle to allow for it to be changed whilst testing. This was replaced for a design with a fixed ankle for strength reasons.

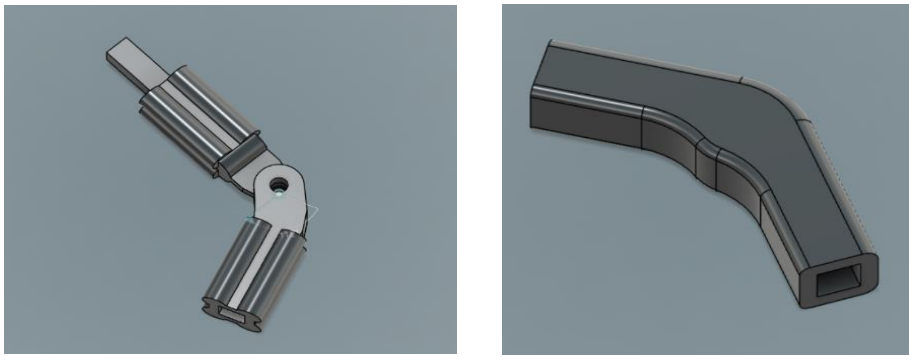


Figure 18 Old ankle designed vs new ankle design

The robot needs to know when it has contact with the ground, so a push button was added to the bottom of each foot. These buttons are connected to the nucleo board.

Figure 19 Button attached to sole of foot

When the robot was put in an upright position on the ground in order to test the buttons, the GUI showed no contact. The first attempt to remedy this was to replace the angled ankles with straight ankles, pushing the foot straight down instead of at an angle, exerting pressure on the sensor buttons.

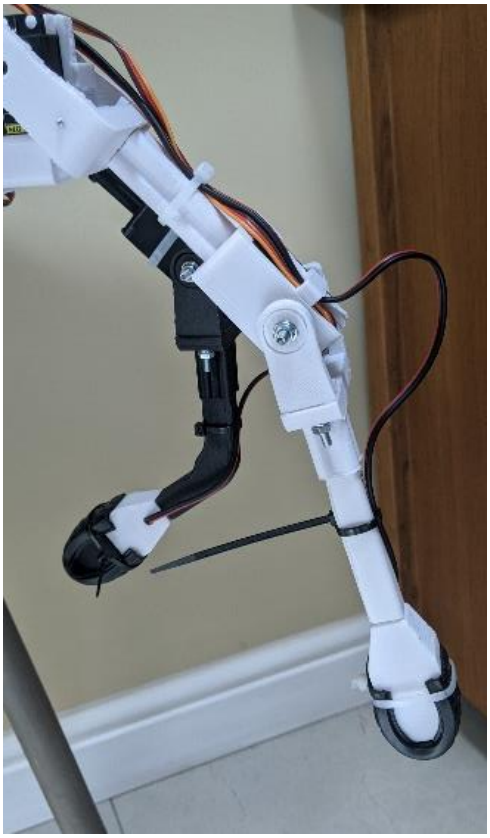


Figure 20 Old vs new ankle design

This change made it so only some feet were showing contact. The issue was attributed to two factors: One being that the buttons were not sensitive enough and the second being that some legs are taking more weight than others. The redesign uses more sensitive micro switches and it also allows for weight sensors to be incorporated which gives better feedback. This is an image of the prototype foot replacement.

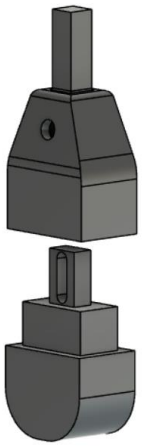


Figure 21 Prototype foot design

Thrust bearings were added to allow for frictionless joints.

To make the manufacturing process quicker and easier, multiple pieces were printed at once. This was done by splitting the components into three files: Components above the knee, components below the knee and components that attach the leg to the chassis. This meant that only three files are needed to print each leg.

Chassis process

The chassis is the part to which all the components are attached. The original design was to manufacture the chassis from multiple 3D printed parts and then attach them together incorporating a plastic plate with bar for additional strength. There would also be two PVC tubes coming from the rear where the shell would be attached. The shell would protect the spider from debris and water. This would be manufactured out of vacuum formed plastic, as it was light and cheap to manufacture

Initial design was originally drawn on a whiteboard.

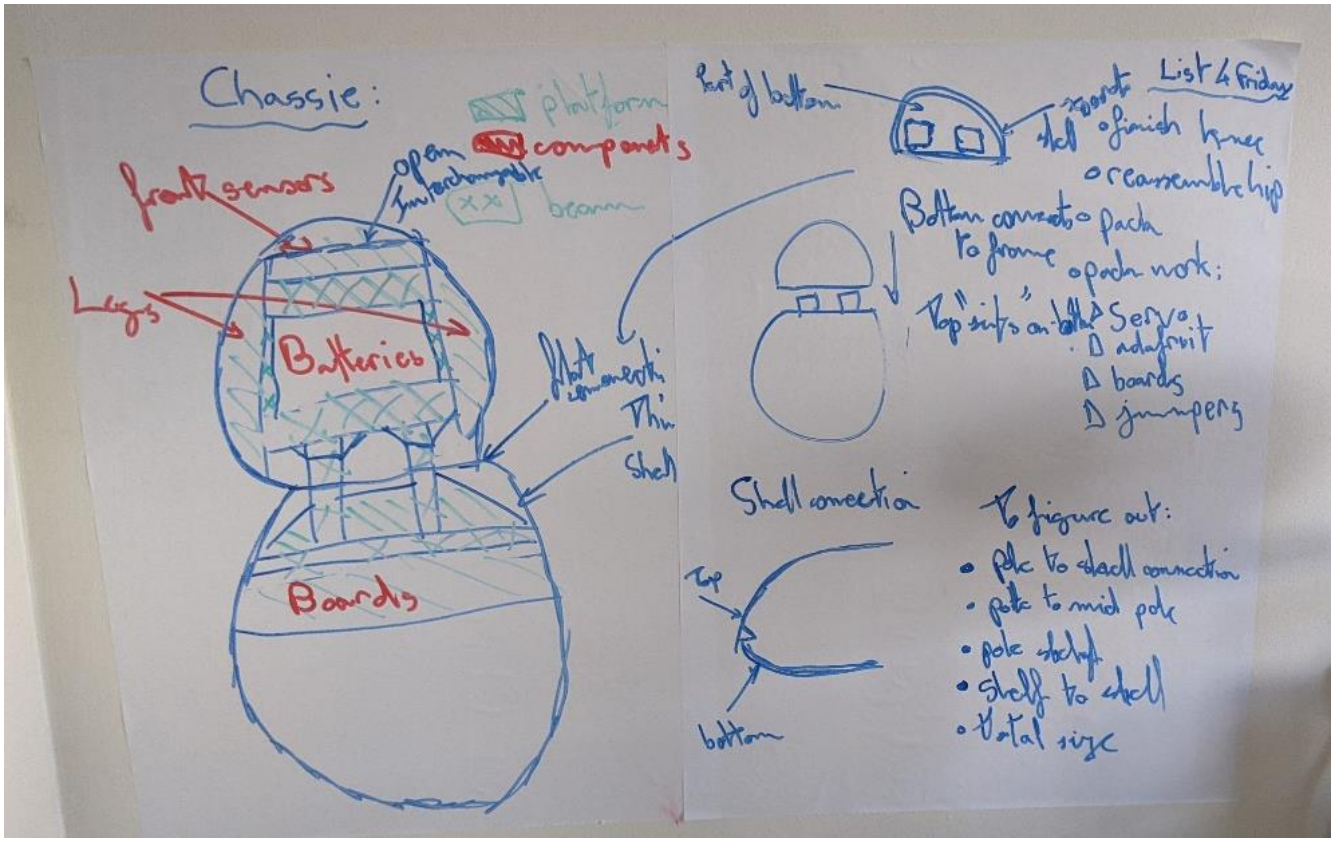


Figure 22 Initial design on whiteboard

This design was then created in Fusion360.

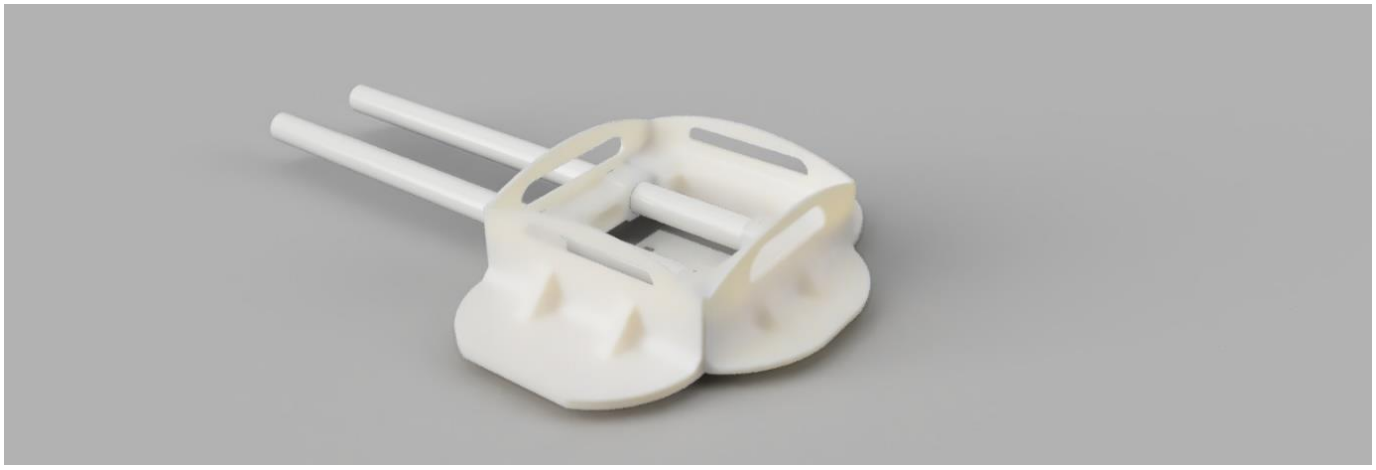


Figure 23 Render of first chassis design

Possibilities for weight reduction were noticed so the chassis was redesigned. Changes included only one pole would be used instead of two, whilst other pieces were made smaller or completely removed.

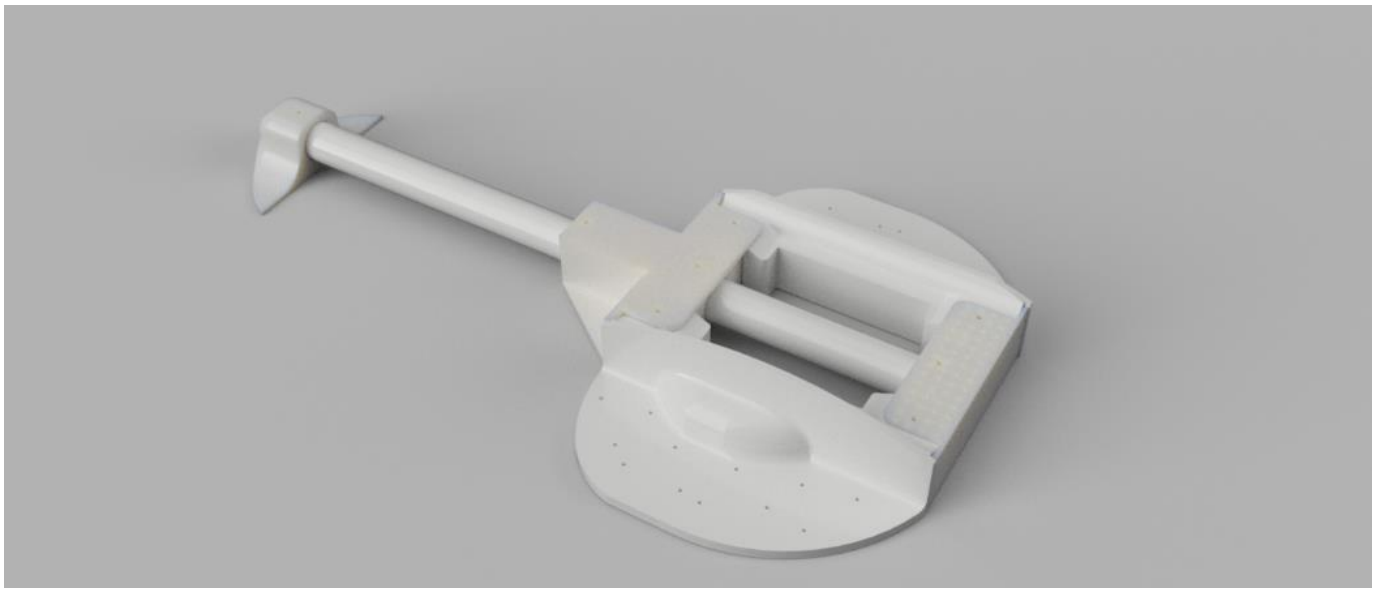


Figure 24 Render of lightweight chassis design

Each part was then converted into a 3D print file and printed. Some parts such as the two side fins were too big. They were therefore split in half and had a small block holding it together.

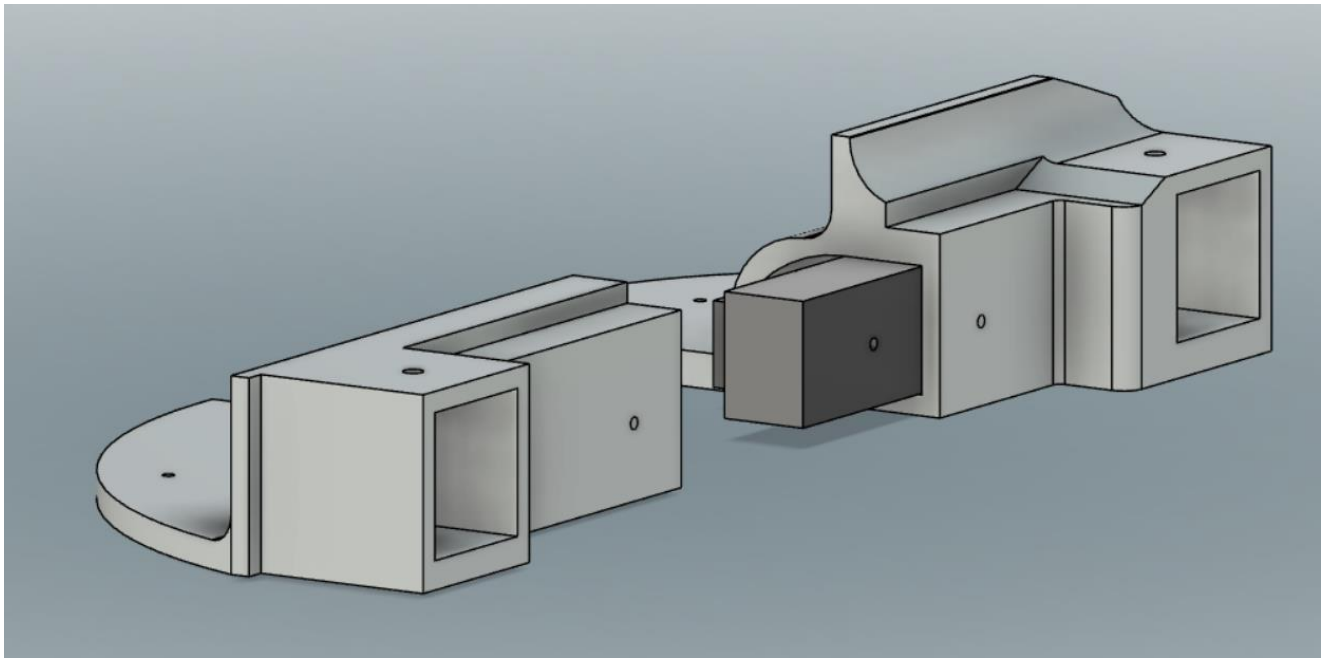


Figure 25 Joining structure of side fins

Each piece was printed and assembled. After assembly of this design, it was still too heavy.

A drastic third redesign was necessary using a totally different manufacturing process. For this, an aluminium composite plate was used. Before the other components were manufactured and attached, there were concerns that due to the thickness of the chassis plate material a screw would not have sufficient grip, therefore a destructive test was done. This test involved screwing a piece of metal to a piece of the composite and seeing if it could be pulled off. The outcome was that piece could only be removed after a significant amount of force was applied, and the concerns were subdued.

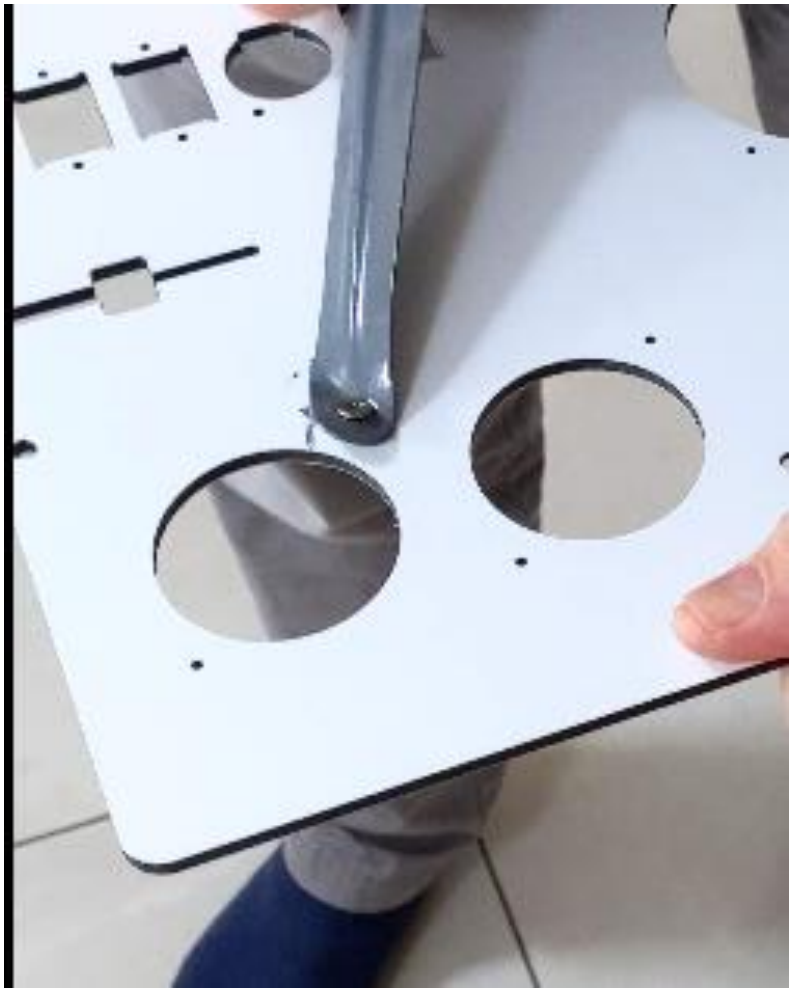


Figure 26 Chassis material destructive test

The chassis plate was bespoke manufactured. This allowed also for the shape of the base to be changed to allow for better servo movement. The aluminium composite is made up of a thick layer of plastic sandwiched between two sheets of aluminium. The plastic allowed for screws to be used and have good grip whilst the aluminium allowed the piece to be light and strong. It was slightly narrower but longer allowing for a wider range of angle for the leg movement. After a weight test was carried out it proved that this material had drastically reduced the weight of the chassis, the aluminium base being under half the weight of the 3D printed plastic one.

Figure 27 Aluminium composite chassis

Electronics process

Main boards

The project uses a Nucleo board for the main processing, an Arduino for sensor management and a Raspberry Pi for the camera live feed and internet connectivity. The Arduino and Nucleo board are connected via USART as this allows for communication between them.

Servos and drivers

Each leg consists of three servos with four legs per side, totalling 24 servos for the entire project. Due to each servo's high current requirement being 50mA under no load, it was not possible to have them powered off the ICs. Therefore, an external power source was used. To control all the servos, a PWM servo driver board was required. After research it was concluded that the PCA9865 board would fit the requirements. The board has slots for 16 servos therefore 2 were used, one for each side. This board allows for expansion as only 12 slots of the 16 are used, meaning more servos could be added to the legs. The board was powered via the Nucleo boards' 5V power supply pin, communicating via I2C.

Batteries

The first choice for a power source was Lithium polymer batteries (Lipo) as these are the most used and have a high capacity. Lipo batteries are only available in certain voltage levels, 3.4V or 7.2V being the closest to the required 6V nominal voltage of the servos. If Lipo batteries were to be used a circuit would have to be added to reduce the voltage, and this would mean more potential points of failure. The main issue of Lipo batteries is safety. If Lipos are not charged and discharged properly they can combust and explode. The alternative was Nickel–metal hydride (Nimh) batteries, they are safer than Lipos and available in increments of 1.5V with satisfactory capacity. The final choice was two 6V 2400mAh batteries.

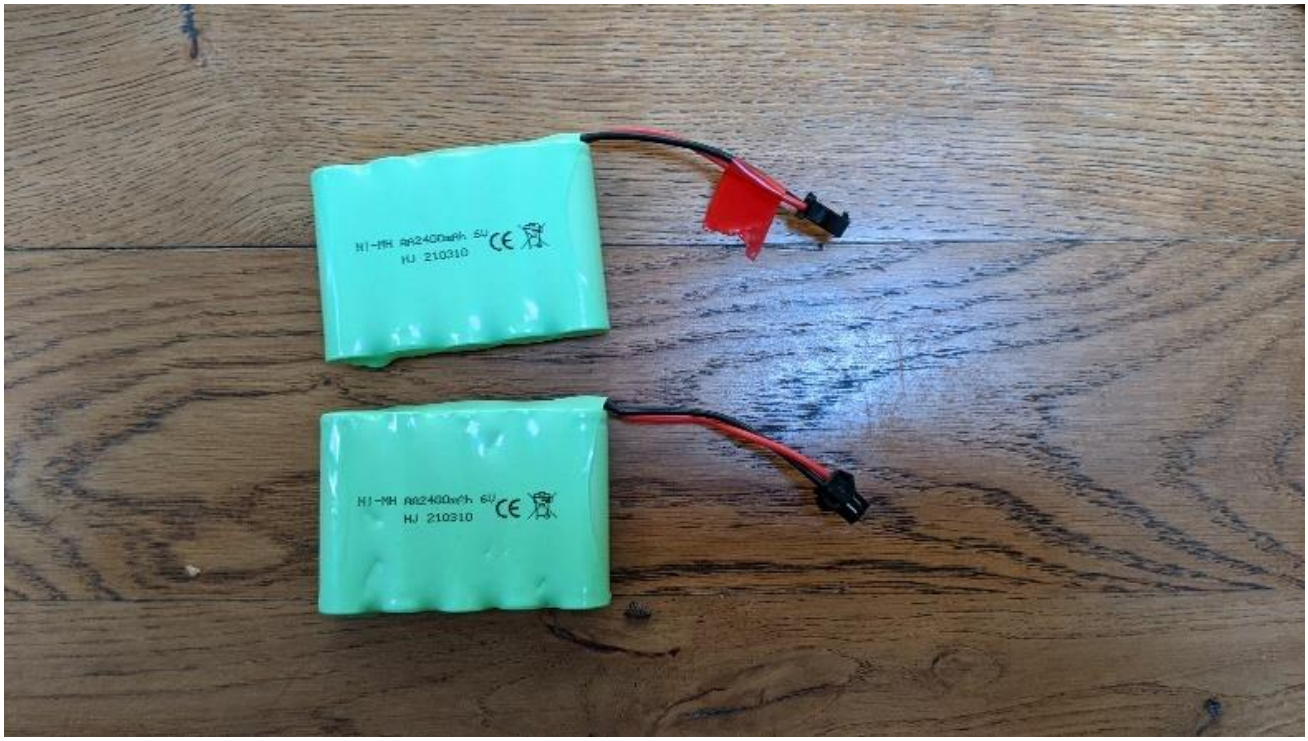


Figure 28 The two chosen Nimh batteries

These batteries needed circuit protection so inline fuse holders and fuses were added, the fuses chosen were 3-amp fuses to protect all components.

IC power supply

To power the ICs a USB battery pack is used. This had the advantage of built-in circuit protection, but it was heavier than standard batteries. The USB port of the ICs were used to power them, meaning they cannot be plugged into and modified whilst powered.

The battery pack had to meet certain requirements, the amount of USB ports, weight, capacity, and output current.

This was the lightest pack that had the required output current. It only had two USB ports, so a USB divider was purchased. The Raspberry Pi had to have an individual port due to it having the highest current requirement, whilst the Arduino and Nucleo board were connected to the splitter. A test was carried out to measure its capacity by connecting all the ICs including a USB webcam connected to the Raspberry Pi and measuring how much the percentage dropped. The result was that after an hour the percentage had dropped by less than 20 per cent.

Bluetooth

To allow for wireless connection between the board and a PC, a Bluetooth dongle was purchased. It functions as a standard com port communication. However, initially it was unreliable, so the code was updated: The communication function by the robot and the GUI taking it in turns to send one piece of information. This is a number between 0 and 1000 which corresponds to a piece of information, the letter x is added to the front of the number allowing any unsynchronised data to resync.

Foot connection

Electronic circuits were created to power the buttons on the soles of the feet in order to allow each signal from the button to be connected to the Nucleo board. A multiplexer was considered. This was rejected as it was too unreliable and made it difficult to use interrupts.

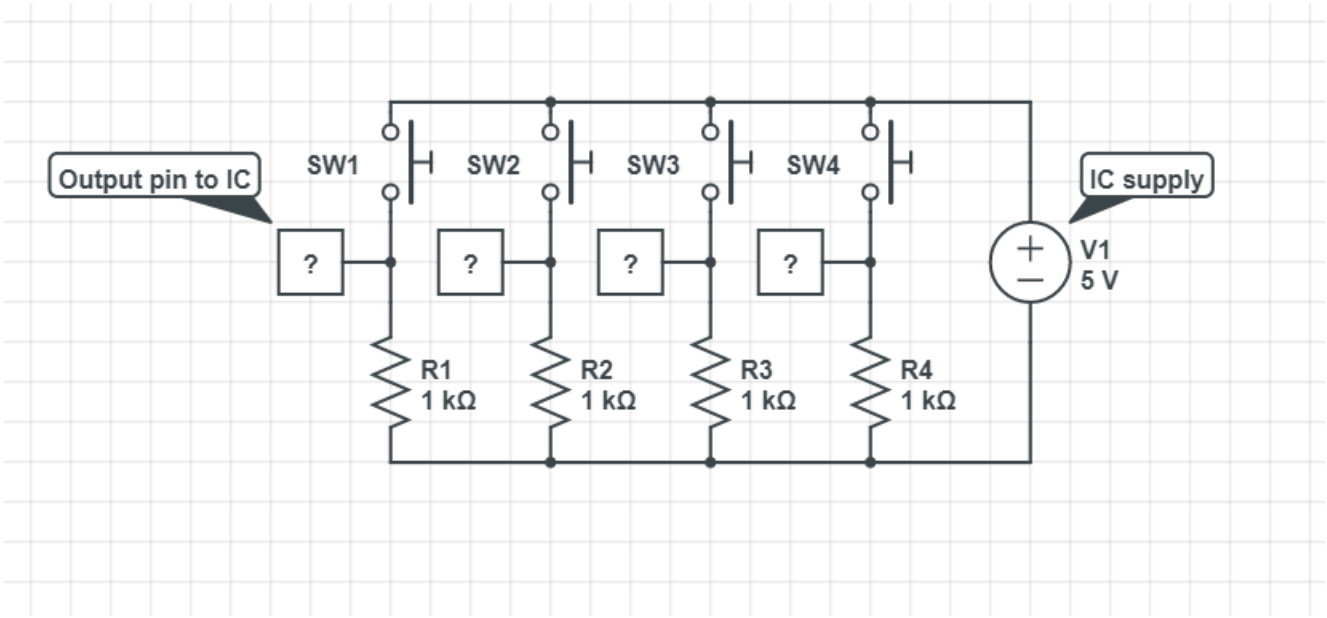


Figure 29 Diagram of button circuits

Two of these circuits were created, one for each side, each powered via the Nucleo board's 5V supply pins.

Voltage measurement

To read the current and voltage levels of each battery, two INA260 Adafruit circuits were connected to calculate the percentage of each battery, it can also measure the current. If the current is high, it suggests that a servo is stuck or that there is a short circuit or that the servos are under too much stress. An improvement would be to add a relay connected to each circuit enabling each side to be turned off should an issue arises.

Sensor and camera connection

The sensors are connected to the Arduino using easily expandable I2C. The inclusion of an Arduino to control the sensors gave the Nucleo board less to do. Additionally, libraries for more advanced sensors such as the INA260 are easily available for the Arduino but prove difficult to find for Nucleo boards.

A USB webcam which is connected to a Raspberry Pi was used for the live feed.

Assembly process

After the legs were printed, they needed to be assembled. Glue was avoided as this did not allow for the pieces to be removed and replaced or upgraded. The pieces were assembled using screws. Pilot holes reduced the chance of the components cracking or splitting.

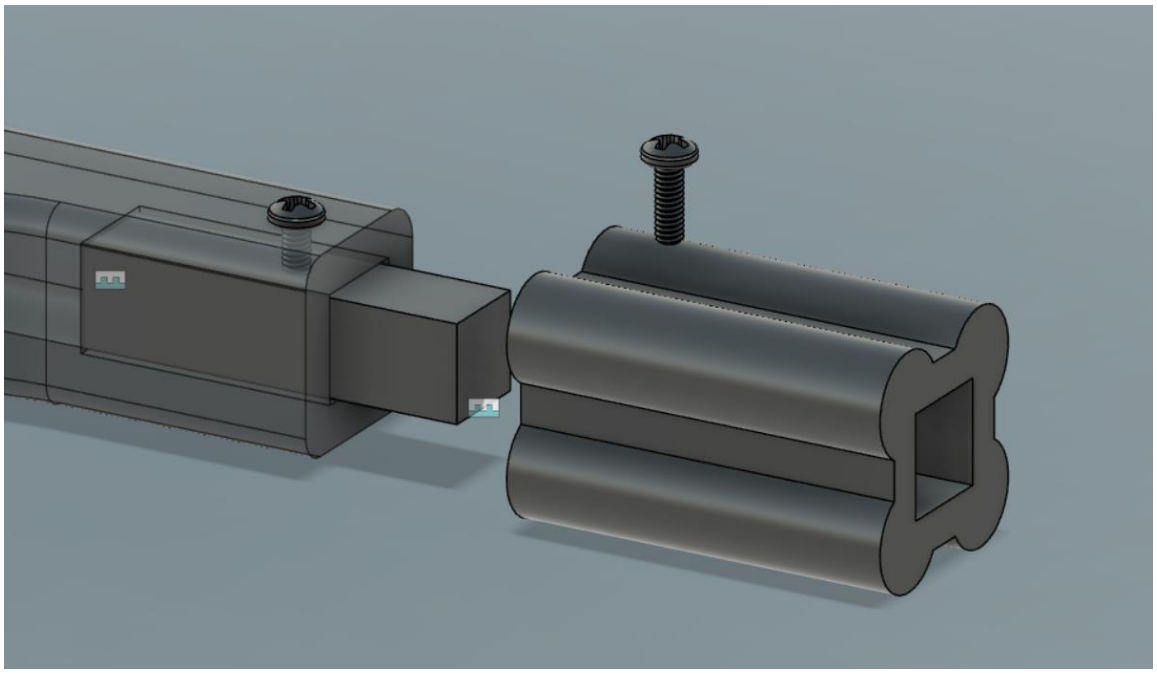


Figure 30 Leg piece assembly

The assembled legs had to be attached to the base. The pieces were screwed to the base, each leg having four screw holes: One on each side, one at the front and one at the back. The screw at the back attaches the chassis to the servo holder with a clearance hole going through the plate. The front screw attached the plate to the chassis but also the screw strengthens the pin, the latter holding the rotator.

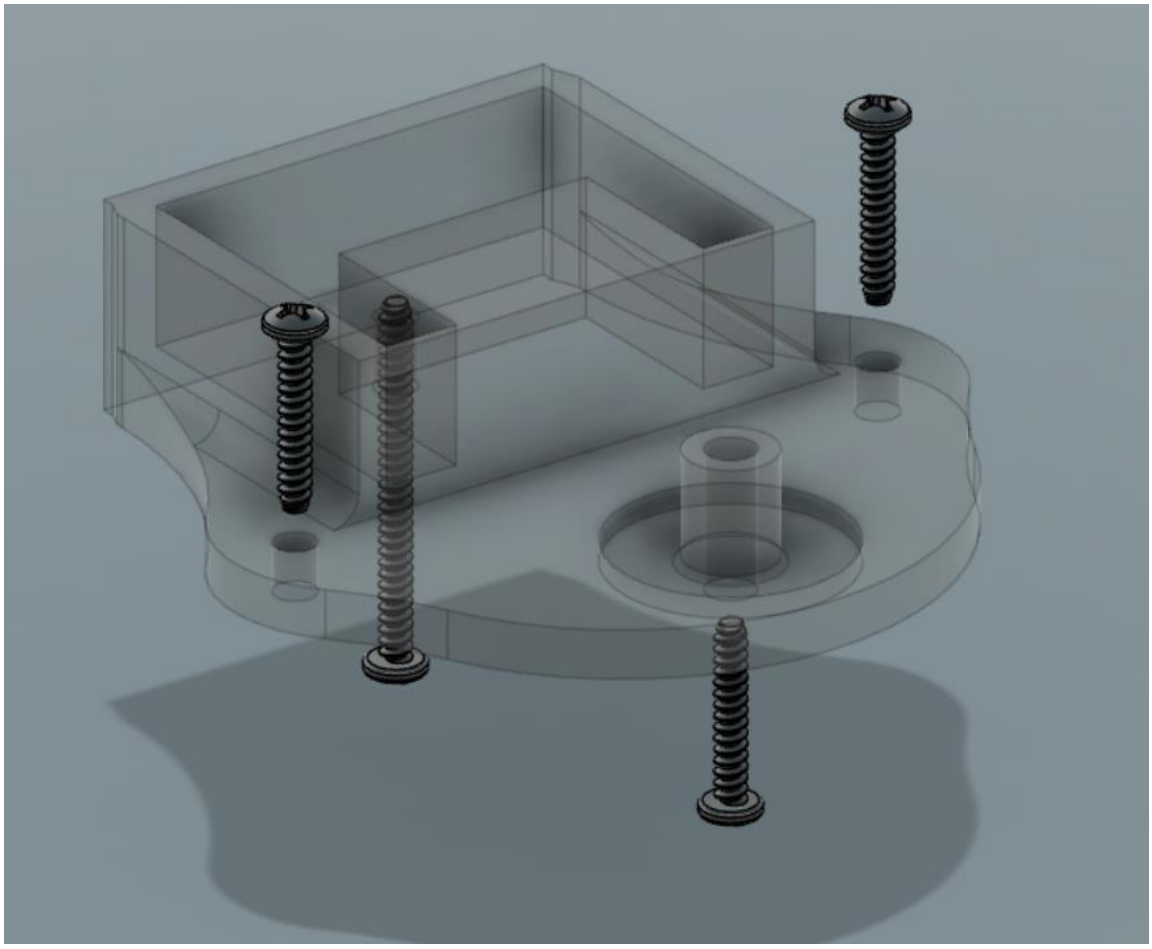


Figure 31 Leg base plate with screws

Small squares of rubber where added which were sandwiched between the attachment plate and the chassis. This absorbs some shock, and it also means that if the plate is not flat the rubber allows for better contact between the two components. Washers were added to spread the load of the screws to stop the plate from cracking.

The first six legs were attached then a standing test was done. The test was to see if the legs could support the base. This was repeated with four legs on the ground and four legs idle.

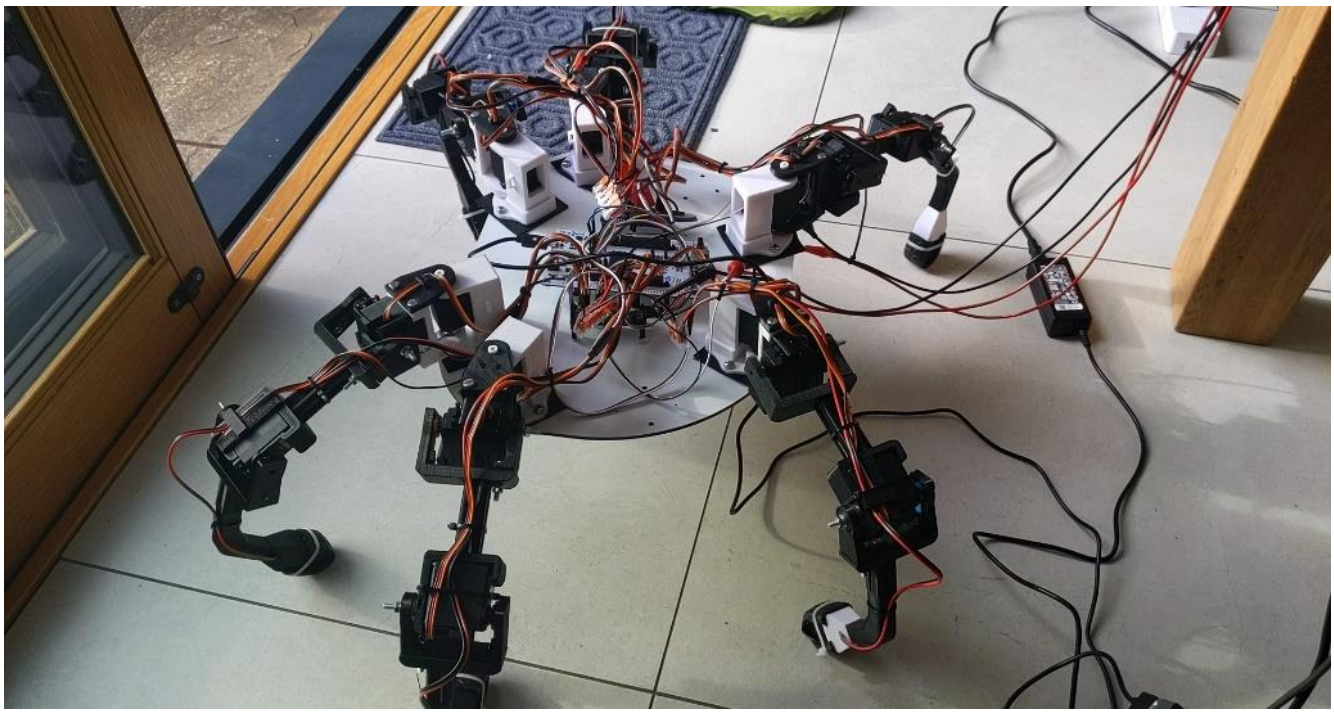


Figure 32 Six leg standing test

To improve cable management, the wires coming from the servos on each leg were zip-tied together. The wires were also zip-tied to the legs stopping them from moving but also as protection.

The three ICs were added next. To increase the amount of space to attach components, a thin plastic plate was printed. This piece was then attached via screws to the top of the legs. The three ICs were added to this plate, secured via nylon PCB mounts that were attached via screws. The other circuits were attached via nylon PCB mounts to the base.

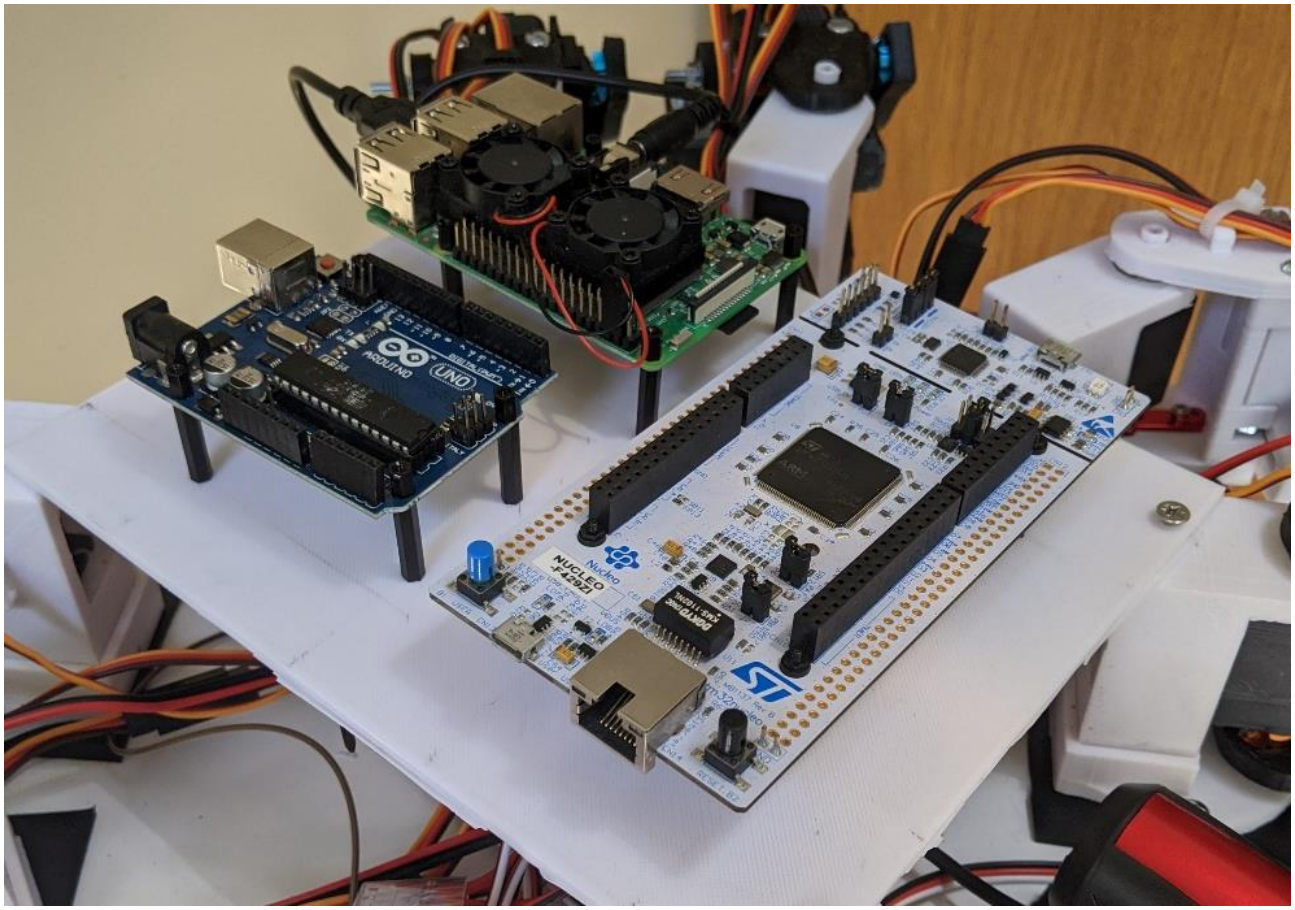


Figure 33 ICs attached to top plate

The batteries were attached using 3D printed clips which lightly clamp the battery on each side and these clips were held in place via screws.

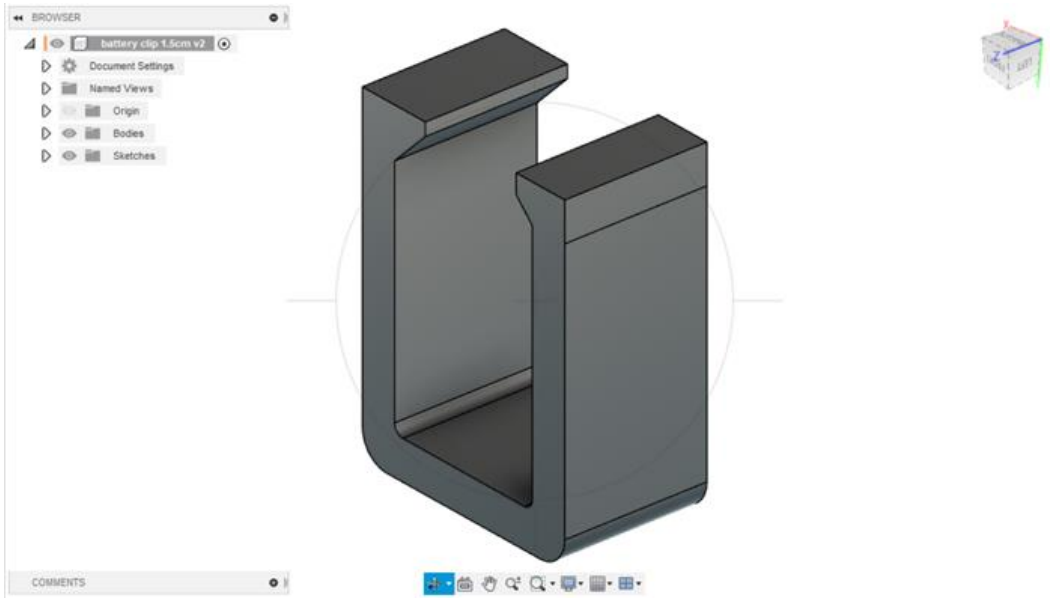


Figure 34 Model of battery clip

The USB webcam was attached via a simple screw mount to the front of the chassis plate.

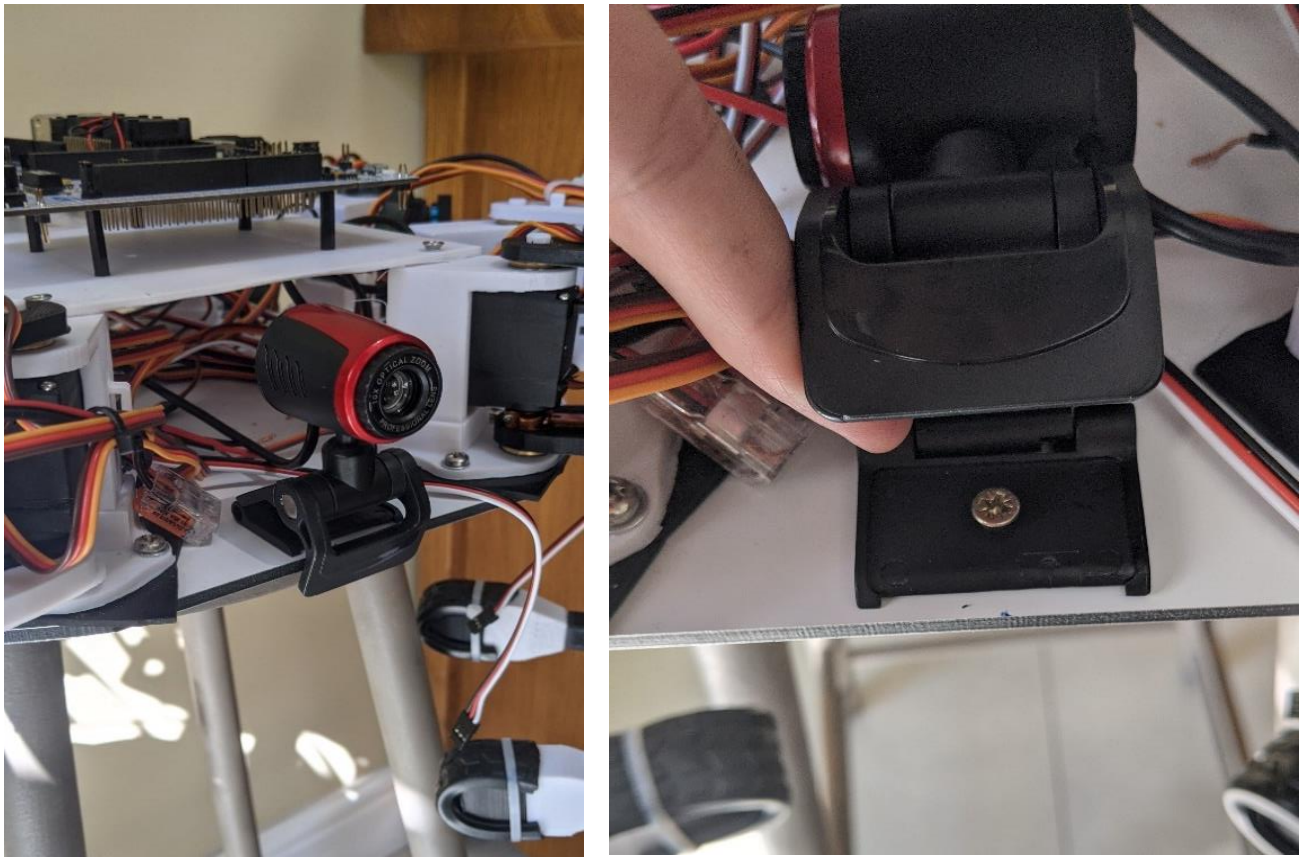


Figure 35 Camera attached to base

A bread board was added via double sided tape where the components are connected to be supplied power from the ICs.

The components were wired together using jumper leads. Although this is not aesthetically pleasing it functions for prototyping.

Coding process

All code is available in the GitHub repository as mentioned in the appendix.

A MATLAB simulation was created to test the leg. Below is an image of possible coordinates the leg can move to. This is without considering real-life restrictions.

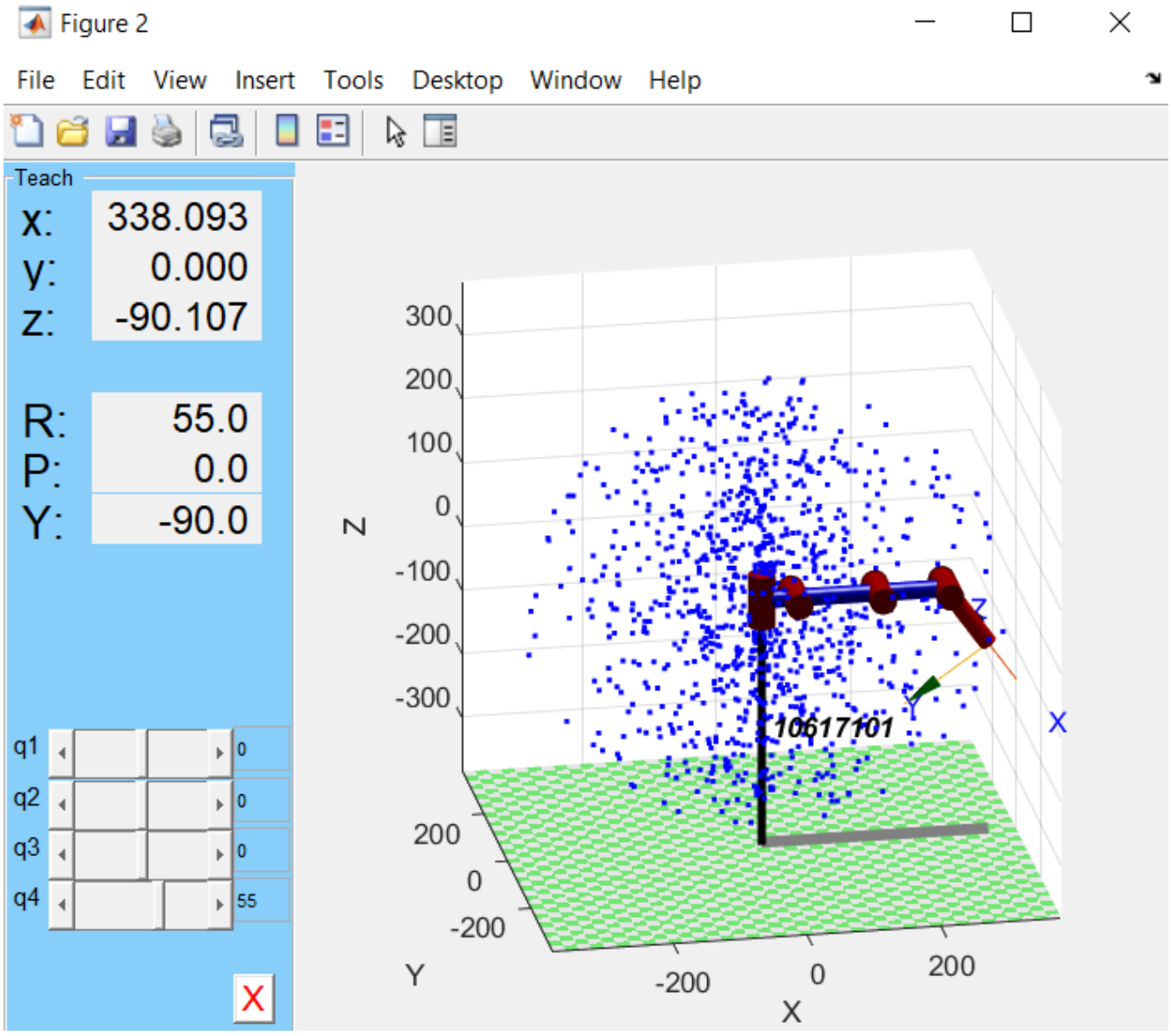


Figure 36 MATLAB simulation

In order to control the servos via the Nucleo board, an Adafruit PCA9865 PWM board was used, one for each side. The communication between the Nucleo board and the Adafruit board uses I2C, it has two data lines controlling the board. According to the data sheet, the Nucleo board used has three pairs of I2C pins. Each board was therefore connected to separate pairs of I2C pins. Unfortunately, during testing, one of the pins failed meaning it could not be used anymore.

As the alternate I2C pins could not be found, the address of the PWM Adafruit board had to be modified, both boards could now use the same pins. This function is due to I2C allowing for multiple devices to be connected on the same data lines. The Nucleo board can then send data to the boards separately if they have different addresses. The address of the PWM board is hard coded via solder joints. To change the address, one of the joints was soldered over.

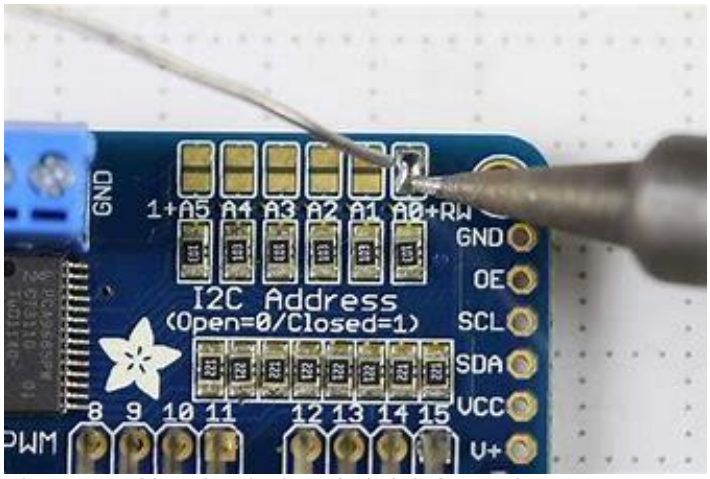


Figure 37 Addressing the boards (adafruit, 2021)

The address of one of the boards now changed they were connected using only one I2C port. This was an improvement, a different board of only one I2C port could be employed.

The Adafruit PCA9865 board has a host of libraries available for the Arduino, which were used for testing purposes but none for the Nucleo board. After months of research and trying multiple different techniques, a forum post had the solution (Charter, 2020). This code allowed for the servos to be controlled via the Nucleo board. The angle of the servo is controlled by modifying the duty cycle of a PWM signal.

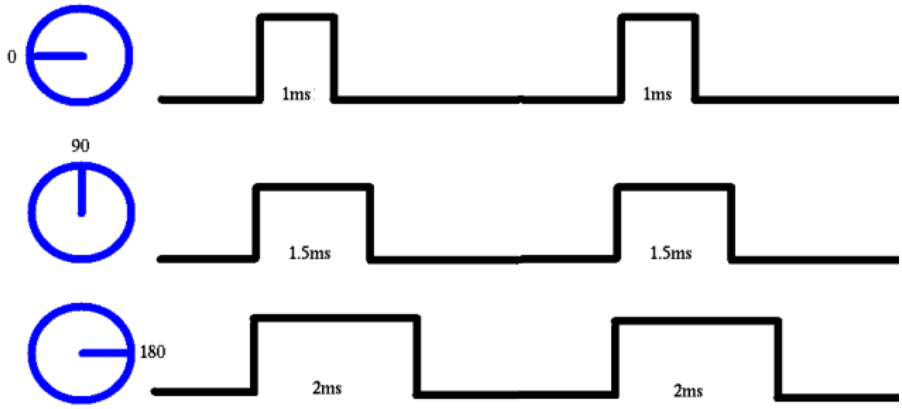


Figure 38 PWM servo control (Musyafa, 2014)

The code that controls the servos was made into a servo class for easier use. Initially the servos were controlled by inputting an integer value between 150 and 500, moving the servo to a specific angle. To make the servo easier to use, a function was created converting an input angle, from an offset, into a number. The servo then moves accordingly. The offset is the starting point of the servo: The integer value when the angle is set to 0 degrees.

```
void SERVO::setServo(float angle){
    // if(angle > _Max){
    //     angle = _Max;
    // }
    // else if (angle < _Min){
    //     angle = _Min;
    // }
    int num = ((((angle)*2.466666))+(_Center));
    setServoOnOff(_Position, 0, num);
}
```

Figure 39 Servo control function

The conversion factor of 2.466666 was calculated, due to the value and the angle being directly proportional, using the equation below, with k being the conversion factor.

$$\frac{value}{angle} = k$$

This new leg class was used to control each leg of the spider, with each leg class made up of three servo classes. Kinematic equations were added to each leg class.

The forward kinematic equation allows for the position of the end effector to be calculated using certain joint angle parameters.

```
487   void LEG::fkunc(int q1,int q2,int q3, float &x, float &y,float &z){
488       q2= q2*-1;
489       float L0 = 50;
490       float qa = 90-q3+q2;
491       float temp = (L0 + (cosd(q2)*L1) + (sind(qa)*(l2+L3)));
492       y = temp * sind(q1);
493       x = temp * cosd(q1);
494       z = (sind(q2)*L1) - (cosd(qa)*(l2+L3));
495       return;
496   }
```

Figure 40 Forward kinematics function

The reverse kinematics equation allowed for the angles of the joints to be calculated when given a position of the end effector. The variables such as the link lengths of the leg and the angle of ankle could be varied, and the equation varied accordingly.

The MATLAB simulation was used to test the forward and inverse kinematic equations of the leg, by comparing the results of the MATLAB simulation to the results from forward and inverse kinematic functions.

The leg class required starting variables which could be changed. All the functions varied accordingly. These included the length of each link in the leg, the angle of the ankle, the side and the position of each leg, each joint max and min angle and the value for when the leg is at its centre position.

To find the value of the leg's centre position a function was created, allowing for the angle of each leg joint to be quickly changed. This calibration involved inputting numbers for each of the joints until the leg was straight and flat.

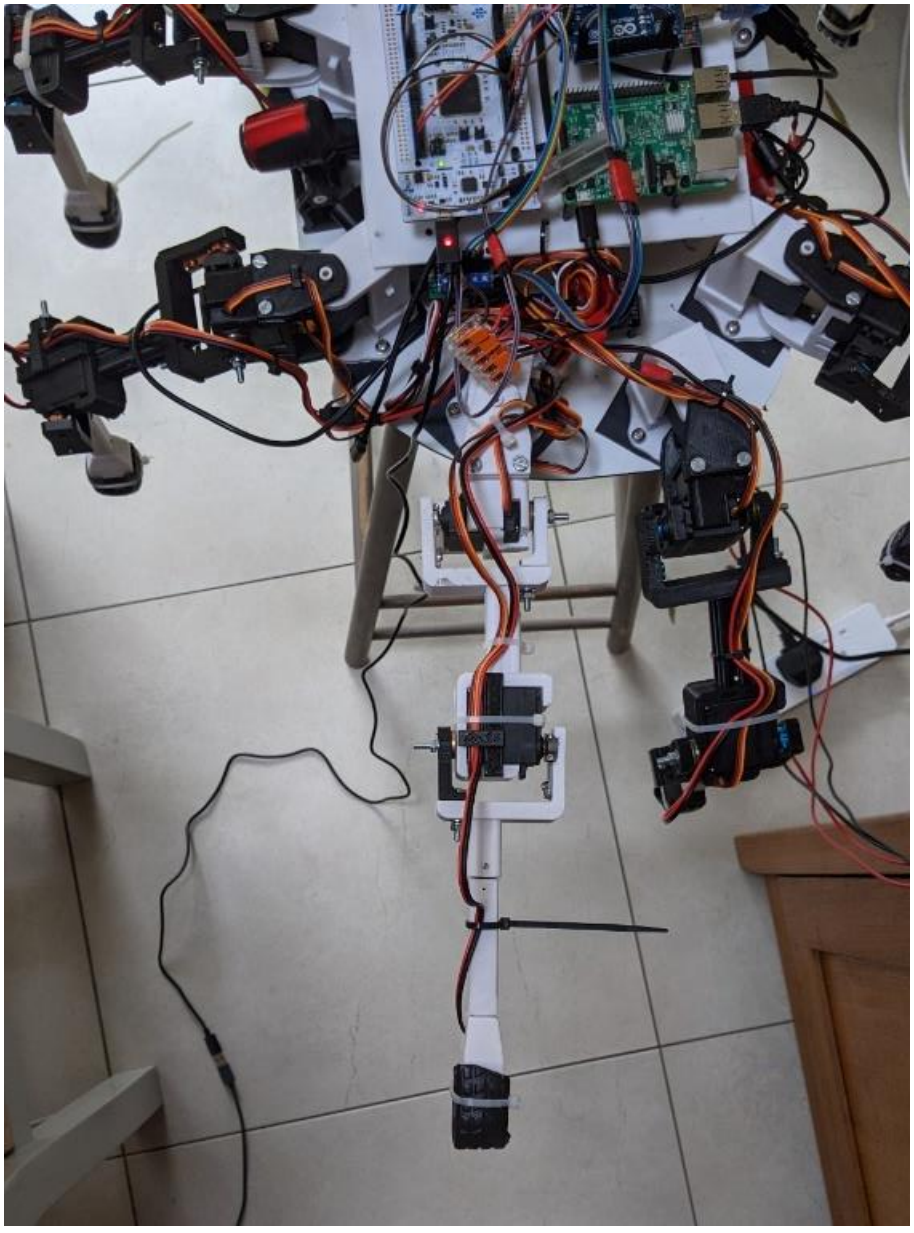


Figure 41 Straight leg for calibration

These numbers, having been saved to an array, were inputted into the leg's start variables. This is repeated for all legs. This is necessary as it aligns up all the coordinate systems making it easier to plan a trajectory for the leg.

The first step was to make the robot stand. Variables were inputted for the leg to achieve standing position. All the angles for the knee and vertical hip movement were the same, allowing for the weight to be spread more evenly. For the horizontal movement they needed to be different. The logical positions for these angles were to have the middle two legs straight whilst having the other two legs at opposing 45-degree angles.

For a demonstration a second stance was created, the ‘intimidate stance’. This stance means that the spider had its front two legs in the air, with the other six on the ground.

Figure 42 King baboon tarantula (Citharischius crawshayi) with fangs out, Kenya, Africa (Brakefield , n.d.) (Cipollini, n.d.)

An issue arose with the robot’s previous standing position. It did not have the stability to do this, so the standing stance had to be modified. The position of the front four legs was changed so that the front two legs were at an angle of 10 degrees and the front middle legs at 45 degrees. This increase in area of the robot support polygon, meant the centre of mass was within the polygon, allowed the robot to stand.

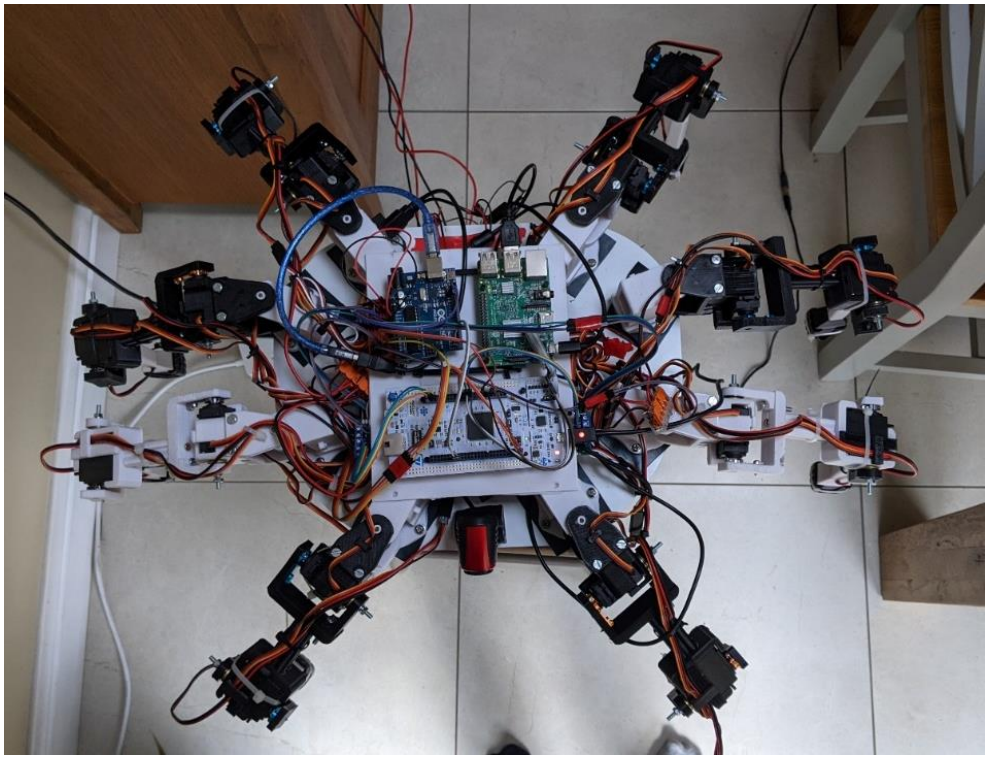


Figure 43 Original standing position

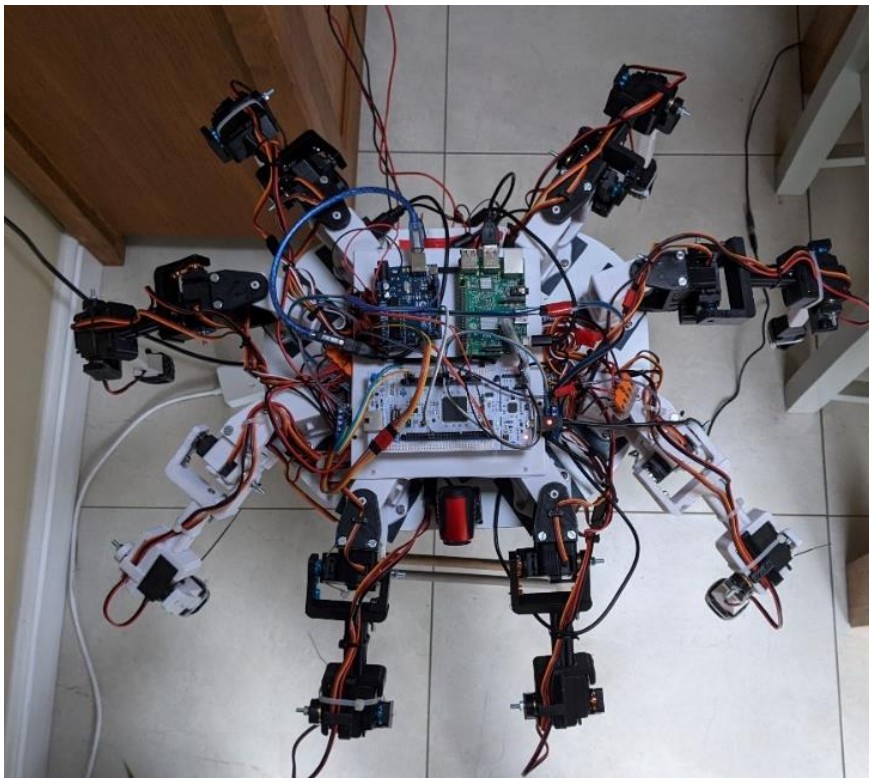


Figure 42 New standing stance

With this new position it can now achieve the intimidation stance.

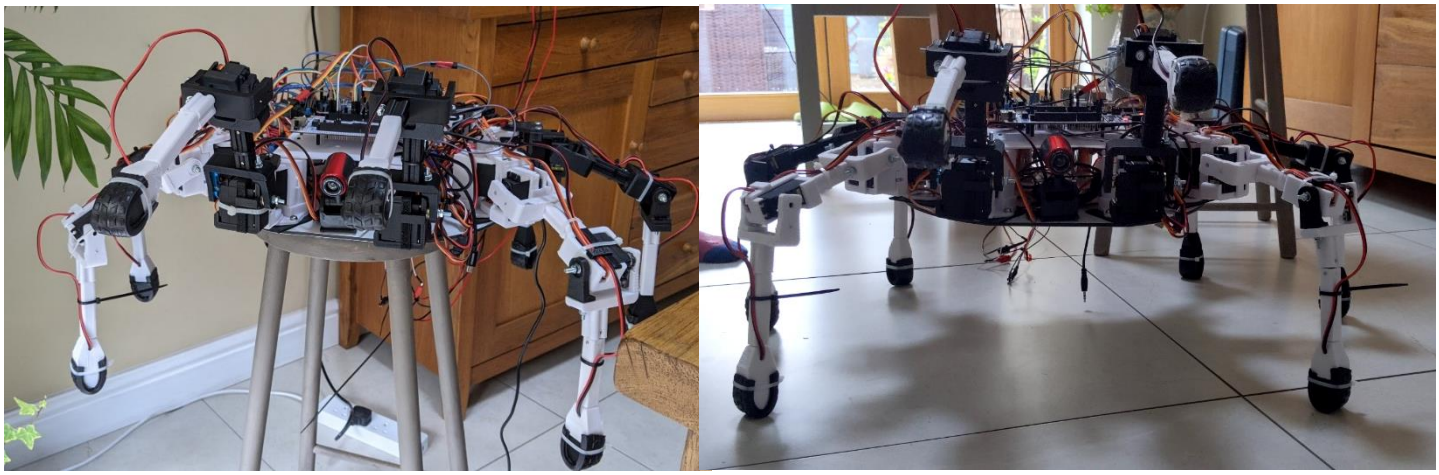


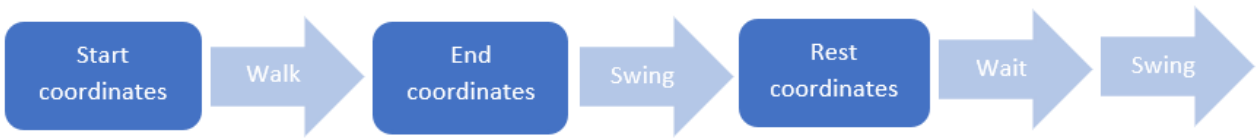
Figure 43 Robot spider in intimidation stance.

The changing of modes is controlled via the GUI, enabling the user to select which mode is required. On the board, LEDs show to the user which mode is selected when the legs are not powered.

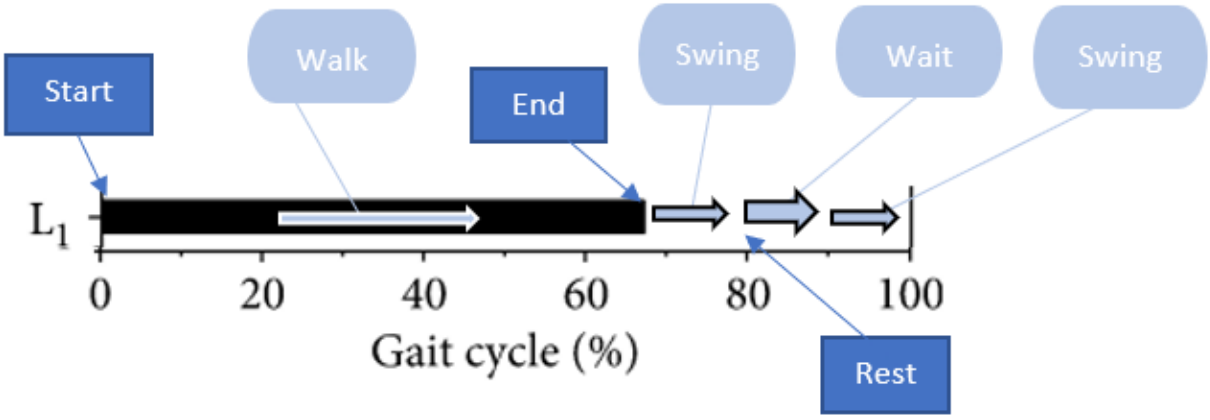
From the Scientific document (Hao, et al., 2019), it can be extrapolated that a spider has three phases: Walk, swing, and rest phase. The walk cycle is when the foot is on the ground and moving the body. The swing phase is when the leg is moving but it is not in contact with the ground and the rest phase is when the leg is off the ground and is not moving.

It does these phases by moving between 3 sets of coordinates. These coordinates are the start and the end of the walk phase and the rest coordinates.

The movement follows the diagram below:



This process can be related to the walk cycle chart (Hao, et al., 2019).



From this information, a function was created that generated an array composed of 100 steps with each step being a set of coordinates. The inverse kinematics function was used to turn these coordinates into joint values which were sent to each leg to move. This function did require several inputs; the start coordinates, the end coordinates, the rest coordinates, and the amount of time the foot is on the ground. The time for the swing was pre-set for all the

equations: To swing for ten steps, then wait at the rest coordinates, then return to the start coordinates. The amount of time the foot was on the ground was gathered from the research paper. The rest coordinates were chosen by moving the leg to a position that seemed adequate, then calculating the coordinates by using the forward kinematics equation. The coordinates for the start and end were more complicated. For the robot to walk, the legs needed to be moving on the ground at the same speed, the speed chosen was 3cm per 14-time steps. This was chosen by having the leg move 15cm in 70-time steps. Fifteen cm was chosen as it was a distance that both the front and middle legs could move without conflicting with each other. The start coordinates for the front leg were the closest coordinates that the leg could move to, these being 6cm on the x axis and 6 cm on the y axis. The z coordinates could be altered at will. This would alter how high the spider body is off the ground. The end coordinates were calculated by transforming the start coordinates by 15cm in the forward direction.

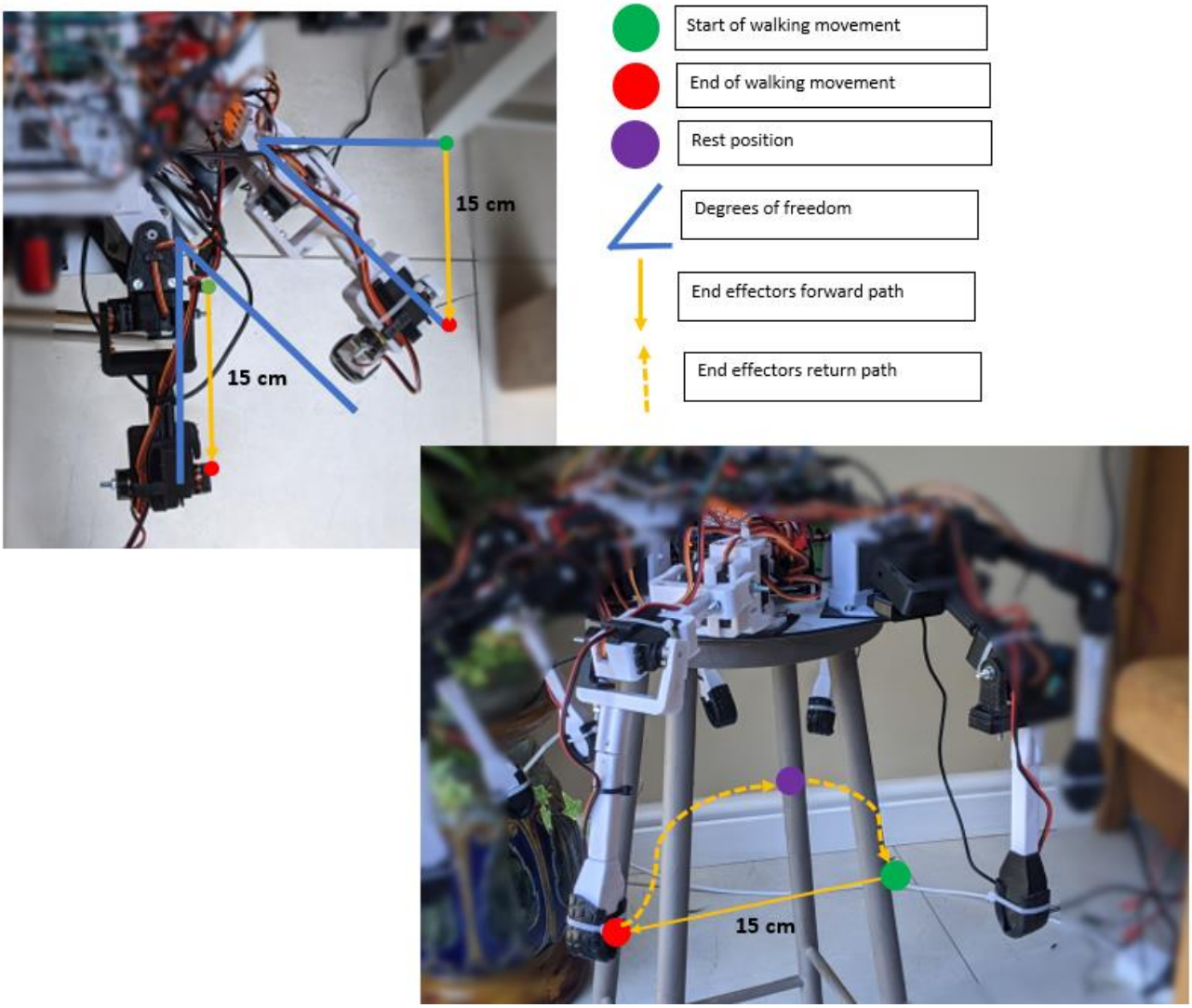


Figure 44 Diagram of leg movement

With these values, the function created a series of coordinates going from the start coordinates to the end coordinates. After reaching the end coordinates the leg moved to its rest position where it stayed until the start of the walk cycle again.

The first version saved each of the coordinates to an array of 100 elements. This method was flawed as over 2000 float values had to be saved requiring a lot of data. To remedy this, a new cycle function was created, outputting the correct coordinates for the input cycle number. Thus, reducing it to 24 float values.

The velocity of the leg was controlled by moving the program forwards through the array when wanting to go forwards and backwards through the array when wanting to go backwards. As shown below.

```
n = n - velocity;
if (n > 99){
    n =0;
}
if (n < -1){
    n = 99;
}
```

Figure 45 Function to control speed

To avoid the legs from colliding, the possible angles of the horizontal hip joint were limited: The legs could go past 45 degrees, this also made the coordinates easier to calculate. Once the coordinates for those two legs had been calculated, the method could be repeated for all legs.

This method was used again for three different arrays need to be generated; one for the forwards and back movement, one for side-to-side movement and one for rotational movement, where the speed was the number of degrees the leg rotated in a certain amount of time steps.

The feet were connected via separate pins to the Nucleo board. The state of each foot was sent to the GUI to be displayed to the user, this was achieved by checking the state of each foot repeatedly.

Future development will allow integration of the ground connection algorithm into the movement. When a foot is lowered as it returns to its starting point, the button at the end of the foot will be poled to check if there is contact. The foot will then continue for more cms to ensure a good contact has been made. It will then stop as new ground is detected. This is how the buttons will function once fully integrated into the movement cycle.

Multithreading and Communication

Multithreading is the ability of a CPU to run multiple threads of code concurrently, by running each thread for a certain amount of time and then pausing the thread before moving on to another thread. This gives the impression of the code running at the same time. Multithreading was used in the project to run the Bluetooth code and the servo movement code simultaneously. However, Bluetooth communication was found to be affected by the code which require strict timings in order to run properly. It could interrupt the sending or receiving of data,

causing race conditions. To solve this, the Bluetooth code could be run without multithreading on the Nucleo board, but in practice the process was slowed down drastically. Another solution was for the Arduino to handle the Bluetooth communication - currently only handling the sensors. At this point in my testing, the Bluetooth functions but not with a 100 per cent accuracy. This is an area to be improved on later.

The main issue with the communication was achieving correct timings, if not, the data would be jumbled. To reduce the number of errors, the data was reduced to a 3-digit number, the receiver knowing how many characters it was waiting for. The letter x was added to the start of the data to make it easier to be deciphered if it would not be sent in the correct order: The communication functions by the robot and GUI sending one piece of data in turn. An issue came up when the GUI code was run on a more powerful computer. Because the code ran faster timings were confused. A frame rate cap was introduced to the GUI to stop this. Another issue meant that data sent was sometimes in error, possibly due to the Bluetooth code on the Nucleo board being run with multithreading. This resulted in the sending or receiving of the data being interrupted, therefore leading to corruption.

Once the data had been received, it was added to a First In First Out (FIFO) buffer which allowed for the data to be stored. This data was then removed one at a time from the buffer and dealt with accordingly. If data needed to be sent, it was added to another FIFO buffer. When sending, data is popped from the FIFO. This data is sent. When the FIFO is empty then an idle signal is sent instead.

The main code for the robot is currently being improved by gradually turning it into an API. This is being achieved by creating classes for the servos and legs by grouping functions together. The legs are being grouped up. The full API creation has not yet been completed due to the code for the walking being still in development. Once that code is finalised, the user will be able to input a direction for the spider to move.

GUI

The GUI was created to display the data from the robot for the user to control. Different software was investigated, the final decision being Processing. It had the features I needed and was open source, meaning that it could be downloaded and modified to meet any user’s need. The GUI needed to include the camera feed, a text box input, sensor feedback, controller feedback and a “Dashboard”. Processing uses the Java language. Most of the UI elements were created

using the controlP5 library as these are aesthetically pleasing, are very well optimised and have a host of customisation options.

The colours were chosen via an online theme picker which selects the colours for each component based upon a choice of three. These were added to the project with their correct names.

The textbox input allows for the user to input more advanced commands such as toggling the Bluetooth. There also needs to be to be a text box that displays what has been typed and the output of the command. This was achieved using the ControlP5 library.

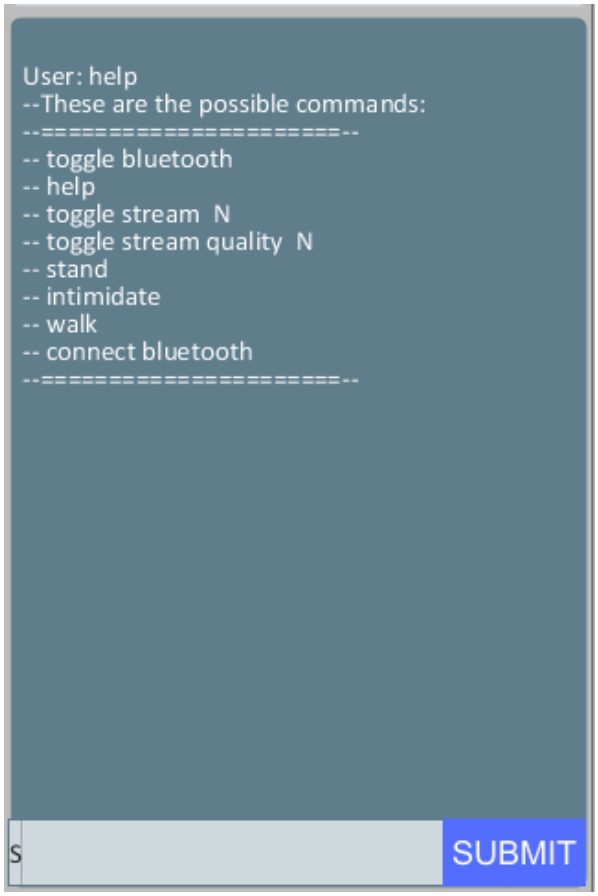


Figure 46 GUI textbox for commands

The ControlP5 library also includes sliders and meters. These were used to visualise feedback from the robot. They display the battery percentage and the current from each side of the robot. This indicates if the servos are too stressed. This section also has sensor feedback for the feet of the robot. If a foot has contact it will relay that info and display it via a blue dot at the spider's foot.

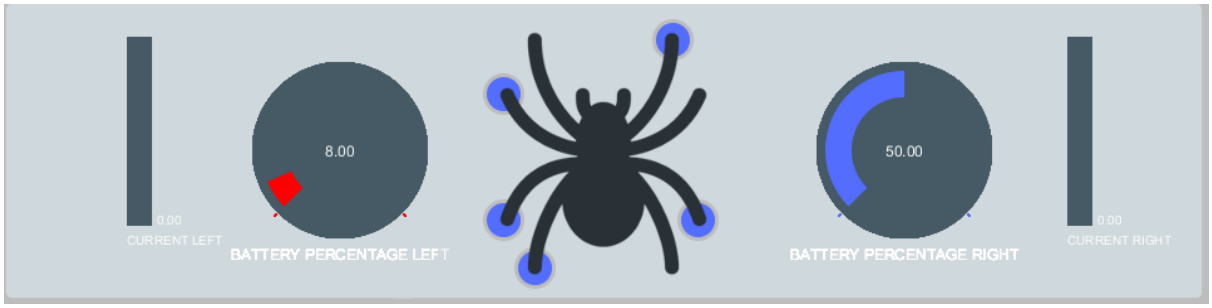


Figure 47 GUI spider status display

The camera feed, which is hosted on a local webserver, is extracted by Processing and is displayed as an image on the GUI. If the GUI cannot get any data from the feed, then a placeholder image is displayed to show there is no connection. An Xbox controller was used for the control It is an off-the-shelf component which can easily be replaced. The controller feedback is displayed by having a small dot follow the motion of the controller stick. This controls direction, there also two images which correspond to a rotational movement, controlled by the Xbox controller's bumpers.

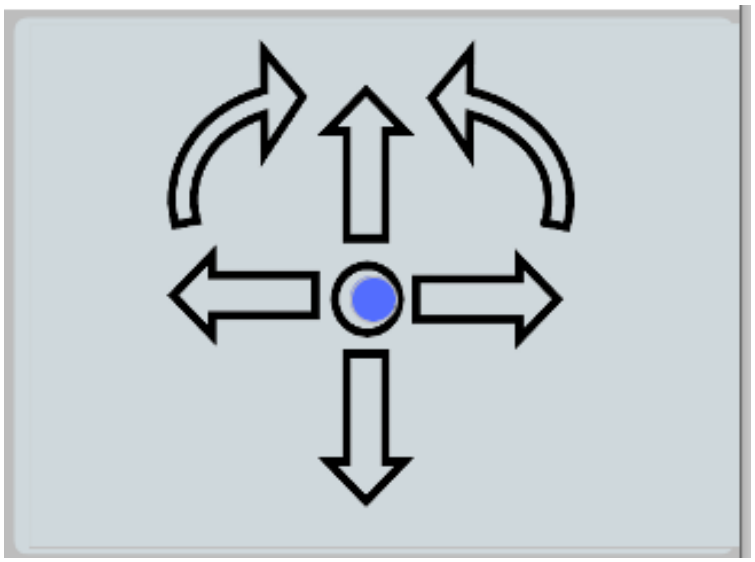


Figure 48 GUI controller feedback

The “Dashboard” displays general information about the robot or the GUI. Currently, it only displays if Bluetooth is on and connected. A flashing red or green dot displays to the user that GUI is functioning and has not crashed. Other symbols will be added such as if the camera is connected or if a controller in connected.

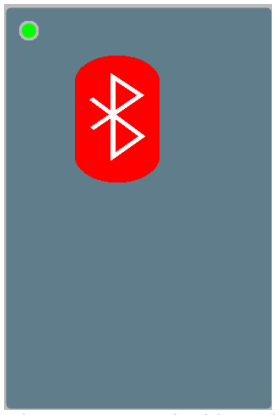


Figure 49 GUI dashboard display

With Processing the code can be exported as an .exe file once completed. This is the current GUI.

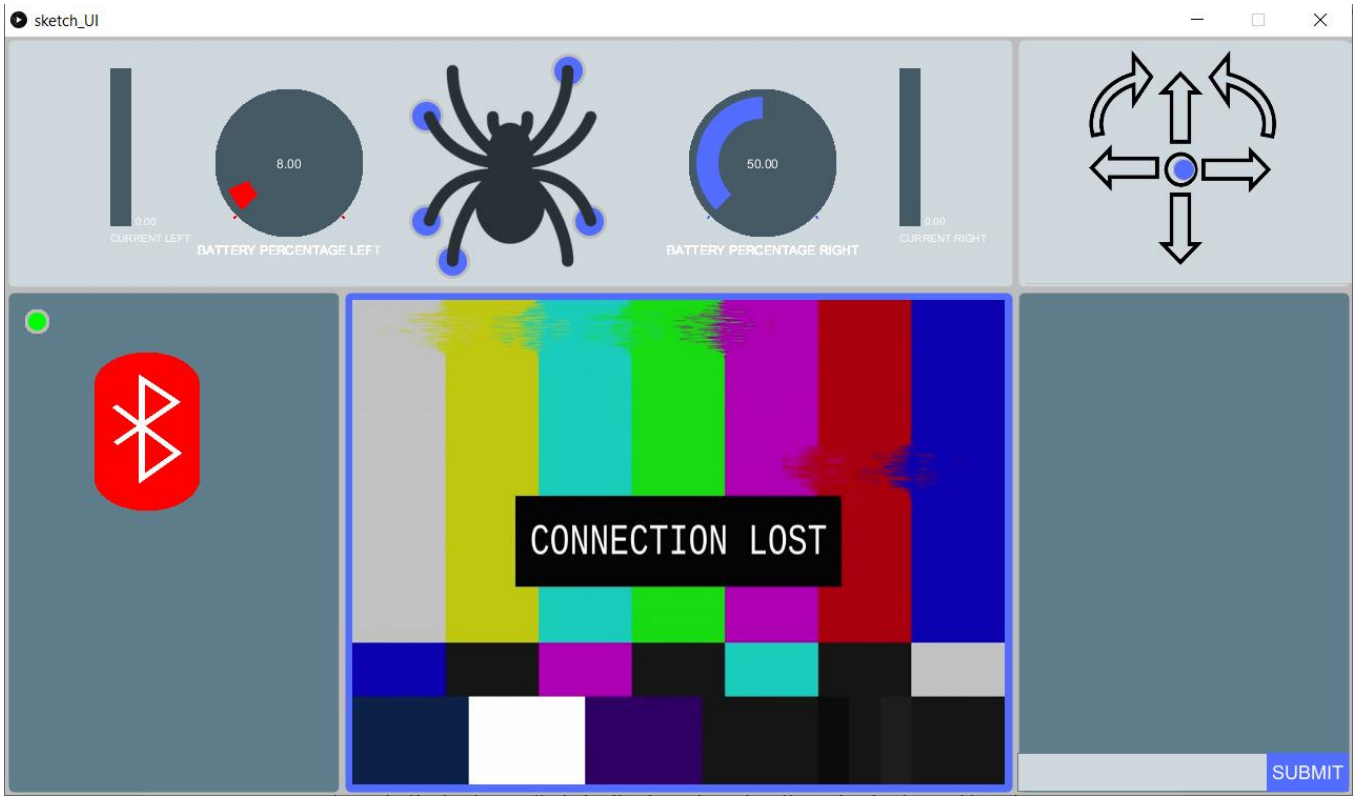


Figure 50 Full GUI

Each section was split into separate files making it easier to use and to find certain components, also allowing for it to be easily expanded upon.

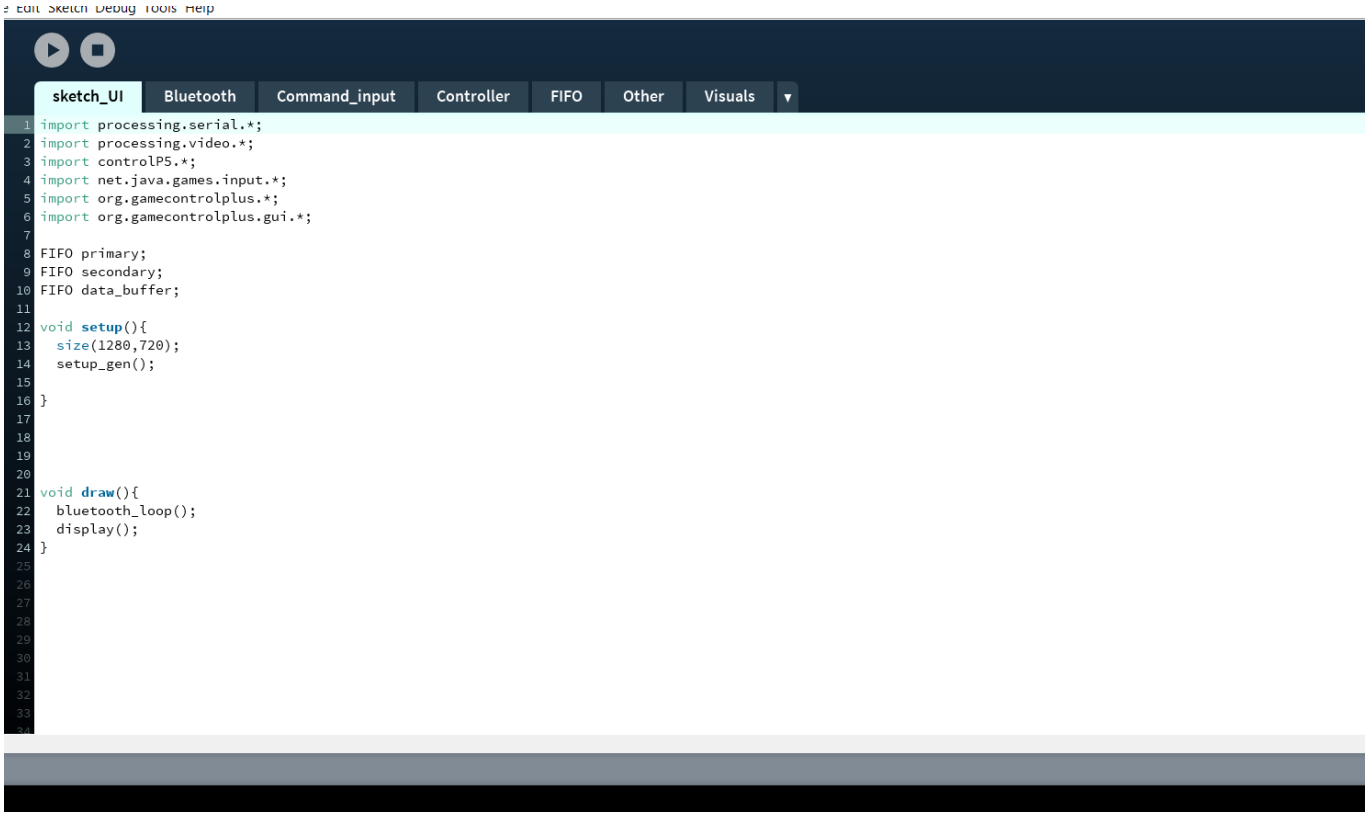


Figure 51 GUI code in multiple files

Onboard sensors - Arduino

An Arduino was added to read and manage the onboard sensors. To allow for expandability all the sensors were connected via I2C which connects all the devices to two data lines. The reason for using a separate IC instead of the Nucleo board is to make the reading easier. An Arduino was used as it is more recognisable, with more advanced sensors, such as Lidar which is more likely to have a library available for Arduino. To read the batteries' percentage and its current, an INA260 was used. This circuit is manufactured by Adafruit and allows for high and low side current and voltage reading. This circuit is controlled by the Arduino.

The code functions by first initialising the two INAs.

```
void setup() {
  Serial.begin(38400);
  // Serial.println("Hollow world");
  // Wait until serial port is opened
  while (!Serial) { delay(10); }

  // Serial.println("Adafruit INA260 Test");

  if (!ina260.begin()) {
    // Serial.println("Couldn't find left INA260 chip");
    while (1);
  }
  // Serial.println("Found left INA260 chip");

  if (!ina260_r.begin((uint8_t)65)) {
    // Serial.println("Couldn't find right INA260 chip");
    while (1);
  }
  // Serial.println("Found right INA260 chip");
```

Figure 52 Arduino code to start INA readings

The commented-out code is used for bug fixing. The main code alternates between querying the current and voltage, read by the INAs every half a second and then converts it into a number. This is then sent to the Nucleo board. A challenge was to try to convert two numbers between 0 and 100 into one number between 0 and 1000 to make it easier to send.

This was achieved by averaging the unit value of each number, that as the unit value of number to be sent. The numbers were then modified to make it easier to send.

```
void loop() {


  current_r = ina260_r.readCurrent();
  // Serial.println("right current");
  // Serial.println(current_r);
  if (current_r < 0){current_r = 0;}
  current_l = ina260.readCurrent();
  // Serial.println("left current");
  // Serial.println(current_l);
  if (current_l < 0){current_l = 0;}
  current_r = (current_r/3000)*100;
  current_l = (current_l/3000)*100;
  float remanderc_r = int(current_r)%10;
  float remanderc_l = int(current_l)%10;
  int remanderc = ((remanderc_l+remanderc_r)/2);
  int decimalc= (floor(current_r/10)*10)+(floor(current_l/10)*100);
  Serial.println((decimalc+remanderc)/2);
  delay(500);
  busvoltage_r = battery_percentage(ina260_r.readBusVoltage());
  busvoltage_l = battery_percentage(ina260.readBusVoltage());
  float remanderv_r = int(busvoltage_r)%10;
  float remanderv_l = int(busvoltage_l)%10;
  int remanderv = ((remanderv_l+remanderv_r)/2);
  int decimalv= (floor(busvoltage_r/10)*10)+(floor(busvoltage_l/10)*100);
  Serial.println((decimalv+remanderv)/2+500);
  delay(500);
```

Figure 53 Arduino code that reads and sends INA data

This system needs to be improved as currently there is no backup if the INAs are not initialised properly, causing an error in the code.

The Arduino is connected to the Nucleo board via USART. USART is a communication method which uses a transmit and receive wire. This was chosen because as it is simple and reliable.

This system is expandable. To add more sensors, they only need to be connected via I2C. the code below shows how the Arduino handles reading the sensors and sending them to the Nucleo board.

Firstly, the I2C address of the sensor is declared at the top of the file followed by the address of the register to read. The values currently used are placeholder values. These actual values can be found on the data sheets of the sensor.

```
12  int sensorAddress = 0x40;
13  #define Register 0x32
14  int Data, Send_data;
15  int rangeMin,rangeMax;
16
```

Figure 54 Arduino sensor variables

Other variables are then declared. The “Data” variable is where the raw data from the sensor is saved and the “Send_data” variable is where the data is saved after it has been modified. The range values are the max and min values of the data received from the sensor.

The actual reading of the sensor is done within the main loop of the code. This achieved with the code below.

```
 99  Wire.beginTransmission(sensorAddress);
100  Wire.write(Register);
101  Wire.endTransmission();
102  Wire.requestFrom(sensorAddress,1);
103  if(Wire.available() <= 1){
104    Data = Wire.read();
105  }
106  Send_data = map(Data,rangeMin,rangeMax,0,500);
107  Serial.println((Send_data)+1500);
108  delay(500);
109
```

Figure 55 Arduino code for sensor reading

The code starts by sending the address of the sensor, then the address of the register in which the data is stored. It requests the number of bytes that the data will be sent. The code then reads the I2C lines for the received data. This data is remapped in order to fit within the range of data that can be sent. An offset is added, which enables for the Nucleo board to know from which sensor the data received comes from. The transformed data is then sent via USART to the Nucleo board. For every sensor added the offset needs to be increased.

Some sensors have more than one register where data is saved. In this case two registers will be written and then the number of bytes requested will need to be increased to two. As shown below.

```
Wire.beginTransmission(sensorAddress);
Wire.write(Register);
Wire.write(Register2);
Wire.endTransmission();
Wire.requestFrom(sensorAddress,2);
if(Wire.available() <= 2){
  Data = Wire.read();
  Data2 = Wire.read();
}
Send_data = map(Data,rangeMin,rangeMax,0,250);
Serial.println((Send_data)+1500);
Send_data2 = map(Data2,rangeMin2, rangeMax2, 251,500);
Serial.println((Send_data2)+1500);
delay(500);
```

Figure 56 Arduino code for multiple register reading

Onboard camera – Raspberry Pi

A camera was added to the front of the robot to display what it sees to the operator. The camera used is a simple USB webcam which can be easily replaced if broken or can be upgraded with a higher quality camera or a stereo camera for more advanced computer vision capabilities. The camera was attached via simple screw mount. Due to the camera stream being extremely difficult to send via Bluetooth, the stream was sent over the network. This was achieved by connecting the camera to a Raspberry Pi 3B, which has built-in internet connectivity. The camera stream was then transmitted using a library called Motion, which sends the data to a local webserver.

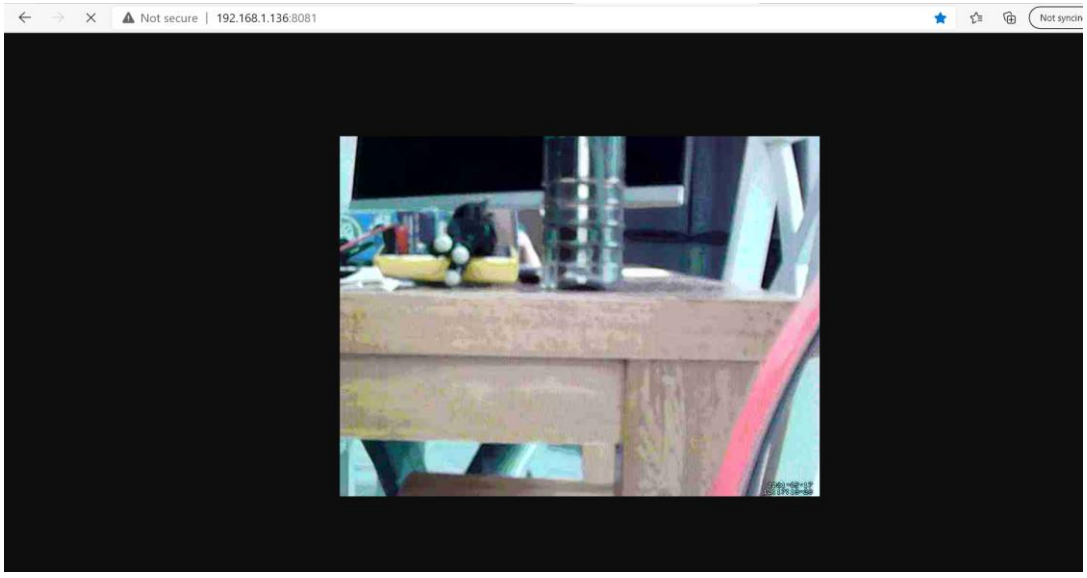


Figure 57 Camera feed on website

The GUI extracts the frame from the webserver and displays it on the GUI. The camera data is only available if the PC is connected to the same network as the Raspberry Pi. This feature could be expanded so that the data is streamed to the web allowing for remote web control. This raises the issue of security and therefore will need to be explored further before it can be implemented.

With the camera stream being sent to the network, it can be easily retrieved for computer vision. For example, the OpenCV library was used for facial detection. This could be combined with an infrared sensor to detect bodies in rubble.

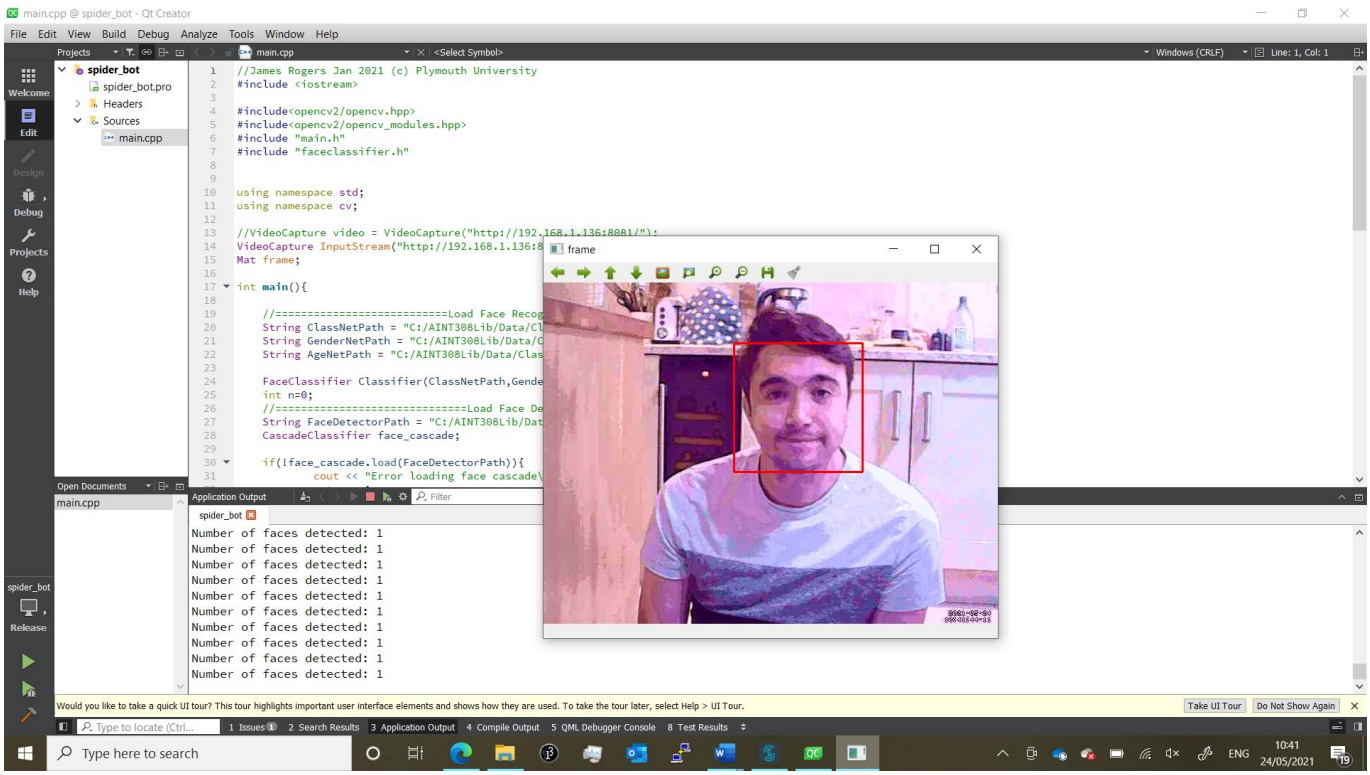


Figure 58 Face recognition demonstration

A further improvement would be the addition and integration of stereo vision. The use of stereo cameras to create a disparity map of the terrain ahead of the spider would allow for the operator to evaluate the terrain. This would also allow for the robot to analyse the terrain and generate a set of movements to overcome it. This could be achieved using the Raspberry Pi, as it has superior processing power.

Important steps to be able to replicate, verify, extend or be taken over

- Code is commented
- Code turned into an API
- Simple modularity
- README added
- Pre-made 3D print files
- Easily expandable

An important design philosophy of the project is that it can be easily recreated and manufactured. Another important part is that the project is expandable and modifiable depending upon the user's needs. This has been reflected in the physical design by making the parts interchangeable and upgradable. This also means the pieces are easily replaced. Most pieces have been 3D printed on a low-grade 3D printer. Even with a basic 3D printer it can be cheaply produced almost anywhere. The chassis plate for this project is a bespoke parts but is not complex and could easily be recreated using other materials if absolutely necessary.

All the electronic components used are off-the-shelf components which can be easily purchased from shops, for example an Xbox controller is used which is widely available and has design features that many people are already very familiar with.

The modularity of the project allows it to be easily upgraded by the user, modified to fit their needs, or additional upgrades can be fitted in the future. The code is also all commented, it can be easily navigated and modified.

To make the process even simpler, the code comes with a README file to allow the API to be used easily. The 3D print files are also prebuilt by me, the user just has to load up the file and apply their own print settings.

Testing, data, and results

Testing was carried out throughout the design and build process.

These tests are listed in chronological order of construction.

Test	Outcome	Page	Comments
Screw through 3D printed hole	Fail	15	A hole tester will have to be created
Strength test of leg piece	Fail	19	Stability pin broke, redesign is needed
Strength test of initial leg	Fail	20	The connection arm broke, both joints will need to be redesigned
Inverse kinematics	Eventually pass	43	Initial tests failed but with changing it eventually matched the MATLAB simulations
Initial Bluetooth test	Pass	52	Data was being sent and received but not with %100 accuracy and therefore will need to be worked on.
PWM driver test with separate I2C pins	Pass	40	The PWM functions with separate I2C pins
USB webcam test	Fail	60	The framerate is about 1fps this is not satisfactory the quality will be reduced
USB webcam test with lower quality	Pass	61	With the quality reduced it functioned with much higher FPS, the issue may also be due to low internet speeds
Multiplexer test for buttons	Fail	33	The data received from the multiplexer was incorrect, a custom circuit may have to be constructed
Ankle strength	Fail	22	Adjustable ankles will have to be replaced for fixed ankles
Initial GUI test	Pass	52	The GUI functioned but was slow this may be due to lack of optimisation
Updated GUI test	Pass	53	With optimisation the GUI is much quicker
Destructive test of aluminium composite	Pass	28	Metal was attached via screw to the composite, and it was tried to be removed, the outcome was that a significant amount of force was required to remove it
Chassis weight comparison test between aluminium composite and PLA	New chassis lighter	29	Both chassis were weighed against each other, the aluminium chassis was lighter
Custom built circuit test	Pass	33	The custom circuit was tested and functioned as designed
INA test	Pass and fail	33/57	The INAs read the current but not the voltage this may be due to wiring
6 legs standing test with two idle legs	Pass	37	It can stand with 6 legs on the ground and 2 idle.
4 legs on the ground with 4 idle	Pass	37	It can stand with 4 legs on the ground and with 4 idle.

Button contact with angled ankles	Fail	23	Did not show any contact, may be due to angled ankles and therefore will be replaced for straight ankles.
Button contact with straight ankles	Fail	24	Only some were shown to have contact. Foot will have to be redesigned to used more sensitive button or weight sensors
Battery pack discharge test	Pass	32	The percentage only dropped by %18 in one hour
Forward kinematics	Pass	43	Worked in only a few tests due to it being simpler than inverse kinematics
PWM driver test using one I2C pin and different addresses	Pass	41	After one of the I2C pins broke, the PWMs had to be connected, they function just as well as before
Initial walking algorithm	Fail	48	Functioned for 2 legs but used up to much memory and therefore cannot be used for 8, a new system will have to be created
New walking algorithm	Pass	50	New system using less variables and has minimal impact on speed which was a concern
First “intimidate” pose test	Fail	45	The spider fell forwards when raising the front two legs, a new stance will have to be used
“intimidate” pose test with new standing position	Pass	47	New stance means the spider can stand on six legs
Machine vision test using USB webcam	Pass	61	The stream from the webcam was grabbed and machine vision used, this opens more machine vision possibilities

The results of this project were the design and creation of a robot that shows the proof-of-concept of all-terrain navigation, with a user-friendly control, sensor feedback and an easy-to-understand graphical user interface. This base is a perfect stepping-off point to be improved upon.

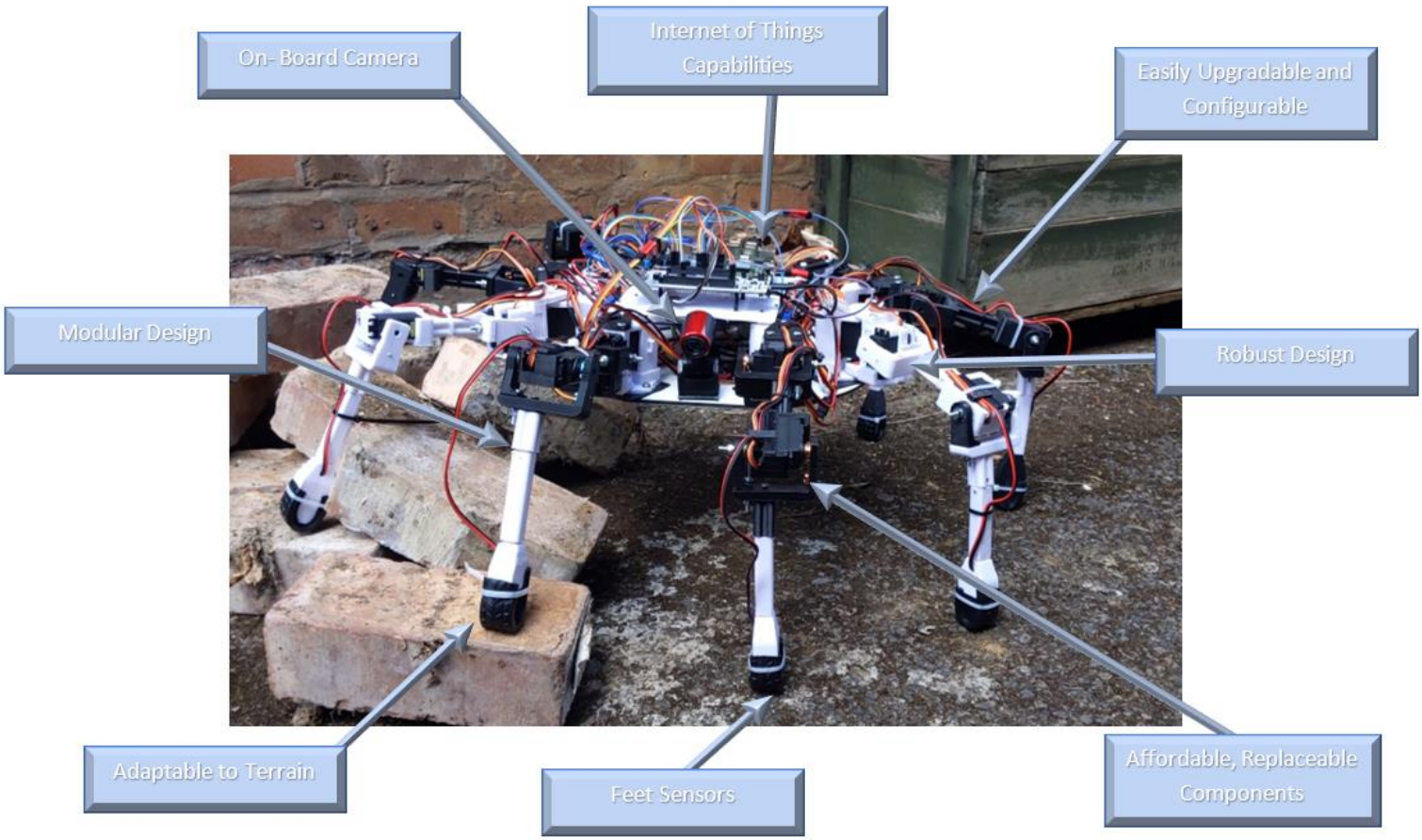


Figure 59 What STEVE can do

Detail project costings

Number	Item name	Quantity	Cost per item	Total cost	Date
1	DSD TECH HC-06 Wireless Bluetooth Serial Transceiver Support Module Slave and Master Mode for Arduino + 4PIN Dupont Cable	1	£ 10.99	£ 10.99	19-May
2	PLA Filament 1.75mm Black, GEEETECH New 3D Printing Filament PLA for 3D Printer and 3D Pen, 1kg 1 Spool	4	£ 19.99	£ 79.96	Multi
3	Maxhood USB 2.0 1 to 2 Y Splitter Cable, USB Power Splitter Type A Male to Double USB 2.0 Female Jack Sync Data Charging Cable Cord 30CM/ 1ft (One Side Only for Charging) (1M/2FM)	1	£ 8.39	£ 8.39	24-Apr
4	Short Micro USB Cable, DETHINTON [5 Pack 25CM] Short Nylon Braided High Speed USB to Micro USB Charging Cables	1	£ 7.99	£ 7.99	21-Apr
5	Standard Car Fuses Holder, Bst4UDirect 100 Pcs Assorted Auto Car Standard Blade Fuses Replacement Kit, 6 Pcs Inline Waterproof ATC/ATO 14AWG Wiring Harness Automotive Blade Fuse(M)	1	£ 9.69	£ 9.69	21-Apr
6	GETIHU Power Bank, LED Display 10000mAh Portable Charger, 4.8A 2 USB Ports High-Speed Battery Pack	1	£ 15.99	£ 15.99	19-Apr
7	FandWay 750Pcs 15-Sizes M1/M1.2/M1.4/M1.5/M1.7 Nickel-Plated Phillips Pan Head Small Self Tapping Screws	1	£ 8.99	£ 8.99	11-Apr
8	MakerHawk 10pcs Servo Extension Cable Lead Wire 320mm 12.59inch 3 Pin Cord JR Male Head and Futaba Female Head	1	£ 5.99	£ 5.99	10-Apr
9	Diymore 6PCS MG996R Metal Gear High Speed Torque servo motor Digital Servo 55g	24	£ 4.82	£ 115.68	06-Apr
10	Kyrio 60PCS PCB Board Kits 20PCS Double Sided PCB Prototype Boards 20PCS 2/3Pin Printed Circuit Board Screw Terminal 20PCS Male/Female Header Connector for DIY Soldering	1	£ 14.99	£ 14.99	05-Apr
11	electrosmart 20m 20AWG 2 Pin Red Black Wire cable	1	£ 6.49	£ 6.49	03-Apr
12	10 Pairs 15cm Long JST SM 2Pins Plug Male to Female Wire Connector	1	£ 6.99	£ 6.99	01-Apr
13	Rubber Sheet 300mm x 300mm x 2mm	1	£ 8.15	£ 8.15	01-Apr
14	Adafruit INA260 High or Low Side Voltage, Current, Power Sensor	2	£ 14.89	£ 29.78	30-Mar
15	DollaTek 3Pcs 74HC4051 8-Channel-Mux Analog Multiplexer Selector Module	1	£ 6.99	£ 6.99	26-Feb
16	DSD TECH HC-05 Bluetooth Serial Pass-through Module Wireless Serial Communication for Arduino	1	£ 10.99	£ 10.99	15-Feb
17	PERFETSELL 5 PCS 25T M3 Metal Servo Horn Rocker Arm Aluminium Alloy RC Servo Arm Horn CNC Upgraded Steering Gear Arm Dual Clamping Tool for Futaba Savox Xcore HL HSP HD Power Go Tech Motor, Blue	5	£ 6.91	£ 34.55	30-Dec
18	SPST 12mm x 12mm x 10mm Tactile Push Switch Button PCB Single Pole Single Throw	1	£ 2.94	£ 2.94	30-Mar
19	HD Webcam Web Cam Camera With Microphone For PC Desktop Computer Laptop UK	1	£ 8.32	£ 8.32	23-Feb
20	5 piece 25mm bolts	4	£ 1.50	£ 6.00	
21	5 piece 50mm bolts	4	£ 1.50	£ 6.00	
	Total			£ 405.86	

Project management

A Logbook was created and was updated on at least a monthly basis. A Gantt chart was created for better time management.

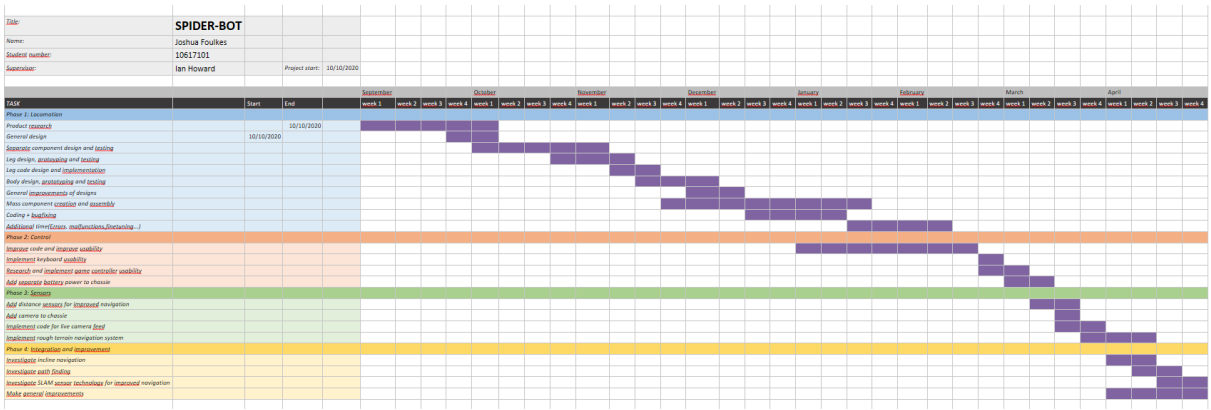


Figure 60 Gantt chart

The timings were changed slightly: The Bluetooth requiring more time and areas such as the camera requiring less time. The legs had to be redesigned and rebuilt, this increased the amount of time taken for that section. Also, some parts were done simultaneously such as the printing of pieces and the coding of the GUI.

Improvements

There are certain general improvements that could be done to the project.

- Upgrade the servos to some that are having a higher torque rating and that are more precise. For this project I was limited on cost. Due to the large number of servos needed, the servos used were less than ideal. The combined servos find it difficult to lift the assembled body at certain angles. The servos used in this project are not extremely accurate and have a large amount of rotational float when set to the desired angle.
- The servos that control the forwards and backwards movement for the legs could also be replaced for servos that require less torque. This improvement would reduce the current of each leg enabling longer battery life.
- The Raspberry Pi allows the possibility of internet connectivity. This would enable the user the option to change how the robot communicates from Bluetooth to via Wifi. The range will be improved if there is internet connectivity. This also means that it could be controlled remotely from someone on a different network.
- Currently I2C has been set up via the Arduino for the addition of more sensors. Additional sensors would allow for better terrain navigation. These additional sensors could be simple ultrasonic sensors to detect whether there is an object in front of the robot or Lidar, as an example, for advanced environment mapping.
- The spider currently does not have a shell, it is vulnerable to water and rubble. The addition of shell would not only protect it but also improve the aesthetics of the project making it resemble more a spider as was the initial intention. The shell would be manufactured using vacuum-formed plastic as it is extremely light and cost effective.
- A separate power source for the sensors will need to be added if more sensors are going to be included. The sensors are all powered via the ICs' power pins. As more sensors are added the current draw will increase which could damage the ICs if the current is too high.

Further improvements – but with possible drawbacks:

- The longevity of the battery life of the robot could be improved by using larger or more batteries, but this would increase the weight of the robot putting more stress on the servos.

- Creating a custom-built PCB, connecting the ICs and the other circuits, would improve the aesthetics of the design and make the assembly easier and less likely to fail due to loose wiring. But this would mean a move from off-the-shelf components to bespoke manufactured items.

With access to more time and funding:

- An additional servo allowing the leg to rotate with the terrain whilst keeping the body straight or by changing the angle of the body to better suit the terrain.
- The addition of an accelerometer would enable the robot to read the chassis tilt and adjust accordingly.
- The addition of further servos on the legs would improve terrain navigation allowing the leg to obtain to more coordinates, but also would allow the foot to have a better contact angle offering improved grip.

Conclusion

This report has shown the creation and design process for a proof-of-concept all-terrain mobile robot and sensor platform for the primary use of search and rescue and having the ability to be adapted for numerous secondary purposes. This project has proved the concept and is an ideal starting point for further improvements and additions. The final product does meet the initial design philosophy, including, but not limited to: modular design, ease of manufacture and simple to use. It has also shown a route for improvements, such as with the addition of further sensors and necessity for upgrades such the need for more effective servos.

Limitations for this project were time, cost, and access to a professional laboratory. Covid restrictions also meant that I was unable to work with my peers. I underestimated the complexity of the algorithm required to allow the robot to walk. I was able to complete the walk cycle process, but further time is required to complete the walking algorithm. Cost limitations meant that the servos used were not as powerful as they could be, and this restricted the potential movement of the assembled legs. Access to a laboratory and dedicated workshops would have speeded up the prototyping and numerous build and test phases.

The project was ambitious and with a methodical approach to design, build and testing the outcome has highlighted what was achievable within the limitations and where the concept can be improved.

I am hoping to continue this project throughout my Master’s.

Table of figures

References

adafruit, 2021. *Adafruit PCA9685 16-Channel Servo Driver.* [Online]
Available at: https://learn.adafruit.com/16-channel-pwm-servo-driver/chaining-drivers

Army Technology, 2021. *iRobot 510 PackBot Multi-Mission Robot.* [Online]
Available at: https://www.army-technology.com/projects/irobot-510-packbot-multi-mission-robot/

BBC News, 2021. *How long can survivors last under rubble?.* [Online]
Available at: https://www.bbc.co.uk/news/world-32485586
[Accessed 18 05 2021].

Boston Dynamics, 2021. *Spot | Boston Dynamics.* [Online]
Available at: https://www.bostondynamics.com/spot#id_third

Brakefield , T., n.d. *King baboon tarantula (Citharischius crawshayi) with fangs out, Kenya, Africa - stock photo.* [Online]
Available at: https://www.gettyimages.co.uk/detail/photo/king-baboon-tarantula-with-fangs-out-kenya-africa-royalty-free-image/200352623-001?adppopup=true

Built In, 2021. *Robots, 12 Examples of Rescue.* [Online]
Available at: https://builtin.com/robotics/rescue-robots

Charter, P., 2020. *Issues with PCA9685 module on Nucleo_F446ZE, MBed 6.0.* [Online]
Available at: https://forums.mbed.com/t/issues-with-pca9685-module-on-nucleo-f446ze-mbed-6-0/9664?fbclid=IwAR2vSmhhOXK_xbwIPboG0ZnNmxkBlQZ__zn_Bk5gQL9UA3VikDMggOk0dkY

Cipollini, B., n.d. *DO TARANTULAS MAKE NOISE?.* [Online]
Available at: https://animals.mom.com/tarantulas-make-noise-10434.html

Hao, X. et al., 2019. *Analysis of Spiders' Joint Kinematics and Driving Modes under Different Ground Conditions.* [Online]
Available at: https://www.hindawi.com/journals/abb/2019/4617212/

Laws, J. M., 2017. *Spider anatomy for artists.* [Online]
Available at: http://johnmuirlaws.com/spider-anatomy-artists/

Musyafa, a., 2014. *Implementation of Fuzzy Logic Control (FLC) In Horizontal Axis Wind Turbine Prototype with Airfoil Profile NREL Standard S83 at Low Rate Wind Speed - Scientific Figure on ResearchGate.* [Online]
[Accessed https://www.researchgate.net/figure/The-servo-position-by-a-PWM-signal-Somers-and-Dan-2002_fig2_301609138].

Pet, n.d. [Online].

Robugtix, 2021. *T8X.* [Online]
Available at: https://store.robugtix.com/t8x/

Ross, D., ed., 2019. *Hyundai Elevate walking car.* s.l.:Enginerring and Technology.

Savage, A., 2013. *Inside Adam Savage's Cave: Awesome Robot Spider!.* [Online]
Available at: https://www.youtube.com/watch?v=-vVblGIIMgw

Teledyne FLIR, 2021. *FLIR PackBot 510 | Teledyne FLIR.* [Online]
Available at: https://www.flir.co.uk/products/packbot/

ThanksBuyer, 2021. *2 DOF Long Pan and Tilt Servos Bracket Sensor Mount kit for Robot Arduino compatible MG995.* [Online]
Available at: https://www.thanksbuyer.com/2-dof-long-pan-and-tilt-servos-bracket-sensor-mount-kit-for-robot-arduino-compatible-mg995-16281

Utires, 2017. *How Off-Road Tires Work.* [Online]
Available at: https://www.utires.com/articles/how-off-road-tires-work/

Bibliography

Os.mbed.com. 2021. *Full API list - API references and tutorials | Mbed OS 6 Documentation*. [online] Available at: <https://os.mbed.com/docs/mbed-os/v6.9/apis/index.html> [Accessed 31 May 2021].

Pears, R. and Shields, G., n.d. *Cite them right*.

Processing.org. 2021. *Libraries \ Processing.org*. [online] Available at: <https://www.processing.org/reference/libraries/> [Accessed 31 May 2021].

Ross, D., 2021. *E&T*, (1).

Schlegel, A., 2021. *processing GUI, controlP5*. [online] Sojamo.de. Available at: <http://www.sojamo.de/libraries/controlP5/> [Accessed 31 May 2021].

Appendix

- Link to project video: URL: https://youtu.be/Fj9wb6O4FhM
- Link to GitHub repository of code: https://github.com/jfoulkes123/Spider-robot.git