Design of a holonomic five legged robot

Final Report

L. Steyn
04496486

Submitted as partial fulfilment of the requirements of Project EPR400
in the Department of Electrical, Electronic and Computer Engineering
University of Pretoria

November 2017

Study leader: Dr. J.D. Le Roux

Copy 2 of 4

Part 1. Preamble

This report is a description of the work I completed during the year on my final year project, design of a holonomic five legged robot.

This report contains a copy of my approved project proposal and documentation on the technical parts of my project. These can be found in parts 3 and 4 respectively. The technical documentation contains a detailed recording of the steps taken to overcome design challenges. This includes circuit diagrams, algorithm flowcharts and test results. This section appears on the CD that accompanies this printed report.

This project does not build on any previous project. Instead it is a completely different approach to the holonomic exploration robot problem that was also addressed in earlier years. Although this project has a similar goal to that of previous years, it does not build on these as the locomotion is completely different.

This document has been language edited by a knowledgeable person. By submitting this document in its present form, I declare that this is the written material that I wish to be examined on.

My language editor was ____________________

_________________________ _________________________

Language editor signature *Date*

I,____________________ understand what plagiarism is and have carefully studied the plagiarism policy of the University. I hereby declare that all the work described in this report is my own, except where explicitly indicated otherwise. Although I may have discussed the design and investigation with my study leader, fellow students or consulted various books, articles or the internet, the design/investigative work is my own. I have mastered the design and I have made all the required calculations in my lab book (and/or they are reflected in this report) to authenticate this. I am not presenting a complete solution of someone else. Wherever I have used information from other sources, I have given credit by proper and complete referencing of the source material so that it can be clearly discerned what is my own work and what was quoted from other sources. I acknowledge that failure to comply with the instructions regarding referencing will be regarded as plagiarism. If there is any doubt about the authenticity of my work, I am willing to attend an oral ancillary examination/evaluation about the work. I certify that the Project Proposal appearing as the Introduction section of the report is a verbatim copy of the approved Project Proposal.

________ _________________________

L. Steyn *Date*

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

LED	light emitting diode
PWM	pulse width modulation
IK	inverse kinematics
PID	proportional, integral and derivative
FPU	floating point arithmetic unit
CAD	computer-aided design
SMD	surface-mount device
LDO	low-dropout
RGB	red, green & blue
BJT	bipolar junction transistor
IDE	integrated development environment

Part 2. Summary

This report documents the development of a robot intended for exploration in unknown terrain by moving holonomically and using its legs for locomotion.

What has been done

A mathematical simulation program was developed in the Python programming language in order to investigate the algorithm for inverse kinematic calculations. This was later extended to a full simulation of the robot with the addition of a GUI and a plotting window that plotted a representation of the entire robot in a 3-dimensional Cartesian coordinate system. This platform was then used to develop the algorithm to make the robot take steps, responding to inputs from the user. When the algorithm was sufficient for a first real world test, the algorithm was implemented on a STM32F7 microcontroller. Leg segments and a chassis were designed in CAD software and printed on a 3D-printer. Servo motors were installed and the robot started to move. To control the robot, an Android application was created to serve as a user interface for the robot. The smartphone uses Bluetooth to communicate elementary commands to a Bluetooth module located on the robot, which passes all the commands to the microcontroller via a serial connection. The robot was heavier than expected and the added weight of batteries and a few unforeseen components made it necessary to implement torsional springs in some of the joints to take some of the force caused be the weight of the robot off of the servo motors.

What has been achieved

The robot can successfully receive and interpret commands from the Android application. The robot is able to move holonomically, therefore it can start moving in any direction without requiring rotation. It is also able to rotate around its own axis without requiring translation. The robot is able to walk on loose and slippery surfaces as well as being able to cross small obstacles.

Findings

Weight plays a very important role in the correct functioning of the robot. If the servo motors are unable to successfully carry the weight of the robot, the robot struggles to perform basic tasks it would otherwise have been able to complete easily. The quality of the servo motors also place a large restriction on the performance and, more specifically, the accuracy that the robot is able to achieve. The inexpensive motors were used because little torque was required for the horizontal servo motors have large slop in the gearbox and can therefore not be moved with high repeatability.

Contribution

There is no specific software package that had to be mastered for the completion of the project. Instead it was just the application of previously mastered software on new problems in ways that extended skills. Although programming in Python had already been mastered,

plotting in 3 dimensions as well as creating a GUI in Python were new skills to be acquired. Using the Bluetooth module of a smartphone through an application developed in Android Studio was a new skill to be mastered as well.

The mathematical model of the robot as well as the movement algorithms were developed from first principles by the student. The electronic hardware consists mainly of the microcontroller and its support electronics, implemented as recommended in the application notes provided by the manufacturer. The interface circuits, used for digital inputs and outputs to the microcontroller, were built using knowledge from prior modules. All of the 3D-printed parts, used in the project, including the battery holders, torsion springs, gears and the LED lens holder were designed from start by the student in a CAD package.

Physical skills gained though the progress of this project includes the soldering of 0.4mm pin pitch SMD components, such as the LQFP100 package in which the STM32F7 microcontrollers are available.

The student came into contact with new electronic hardware in the form of servo motors. The control of one of these with a microcontroller had to be mastered before being able to control all 15 at once using no external control hardware. The use and proper implementation of Lithium-Ion batteries was also new to the student prior to this project.

Part 3. Project identification: approved Project Proposal

This section contains the problem identification in the form of the complete approved Project Proposal, unchanged from the final approved version.

1. Problem statement

Motivation. Robots used for exploration need to be highly manoeuvrable to cross terrain not possible for humans and cars. The motivation for this project is to develop a robot using five legs, capable of crossing rough terrain, that can move holonomically. A robot like this can be used to explore autonomously and can also be used to carry supplies or equipment to remote places.

Context. A vehicle is considered holonomic if it does not suffer from a phenomenon called the parallel parking problem [1]. This phenomenon restricts the vehicle to forward and backward movements only, while slightly turning the front wheels. The vehicle is therefore capable of moving in arcs but never sideways. One approach to achieve holonomic movement in vehicles is to swop the cylindrical wheels found on most vehicles for spherical wheels [2]. While this solution solves the parallel parking problem, it introduces a new restriction of only functioning properly on relatively flat surfaces, unless very complicated suspension mechanisms are implemented. This solution is therefore not suitable for any off-road application. This project will extend on this to solve the problem of terrain without sacrificing holonomic movement. LS3 [3] and BigDog [4] are examples of existing legged robots that function in a way that mimics the way four legged animals walk. These robots were both developed by Boston Dynamics to help humans carry loads across terrain that is not accessible by car. They can therefore follow someone on a foot trail as well as being able to navigate to a given location using GPS.

Technical challenge. The aim of this project is to build a five legged holonomic robot which is capable of moving in any direction from a standing position as well as being able to rotate about its own axis. The control system of the robot will consist of a processor calculating all the required joint angles of all the legs to move the legs in a way that the robot moves in the desired direction. The engineering challenge in this project is the development of a control system algorithm that is fast enough to make the robot react in real time while still being thorough enough to ensure that the robot does not damage itself or its surroundings while moving.

Limitations. One of the technical limitations that makes this project challenging is the trade-off considering the length of the robot legs. Longer legs will mean more manoeuvrability for the robot and faster movements but will also mean increased torque required by the servo motors. This causes a larger power consumption and therefore a shorter battery life. It is also easier to construct a larger robot but this will again eventually lead to bigger power requirements.

2. Project requirements

ELO 3: Design part of the project

Mission requirements of the product

A five legged holonomic robot will be built in this project. The requirements of the robot that would determine whether the project is successful can be summarized in the list below.

- The robot should be able to move in any direction from a stationary position.
- The robot should be able to rotate about its own axis while remaining in the same position.
- The robot should be controlled remotely by using a smartphone application.
- The robot should use five legs to execute any of the required movements.
- The robot should be able to move on both smooth and coarse surfaces.
- The robot should be able to move on both flat and slanted surfaces.

Student tasks: design

The tasks that are vital to ensuring that the product meets the mission critical requirements are listed below.

- The mathematical analysis and design for the movement of the legs should be done on paper.
- The design should then be implemented in a graphical mathematics package such as Python for further refinement.
- Once the algorithm design is sound, electronic design can commence in a simulation environment such as LTSpice.
- The design can then be implemented in electronics using a microcontroller and support electronics together with some driving circuitry.
- A smartphone application should be developed to remotely control the robot.
- Each subsystem should be tested for isolated functionality as well as interaction with other subsystems to make sure all requirements are met.

<u>ELO 4: Investigative part of the project</u>

Research questions

The investigative section on this project will focus on the amount of legs of a legged robot and the effect on stability. Do five legs provide more stability when walking on slippery surfaces? Can the robot still work without some of its legs?

Student tasks: experimental work

The experiments that will be conducted to address the research questions above are listed below.

An experimental setup to test the ability of the robot to move on a variety of surfaces is required.

- The robot will be placed on both smooth and coarse flat surfaces to perform the same manoeuvres.
- The robot should be able to perform these manoeuvres with a similar degree of difficulty and in similar time.

Another experimental setup is required to test the robot's ability to handle small obstacles and bumps.

- The robot will be placed on a surface with bumps and small obstacles such as rocks.
- It should be able to execute the same set of manoeuvres mentioned in the experimental setup above in similar time.

3. Functional analysis

The functional analysis of the system can be shown best in a flow diagram. This can be seen in Figure 1 below.

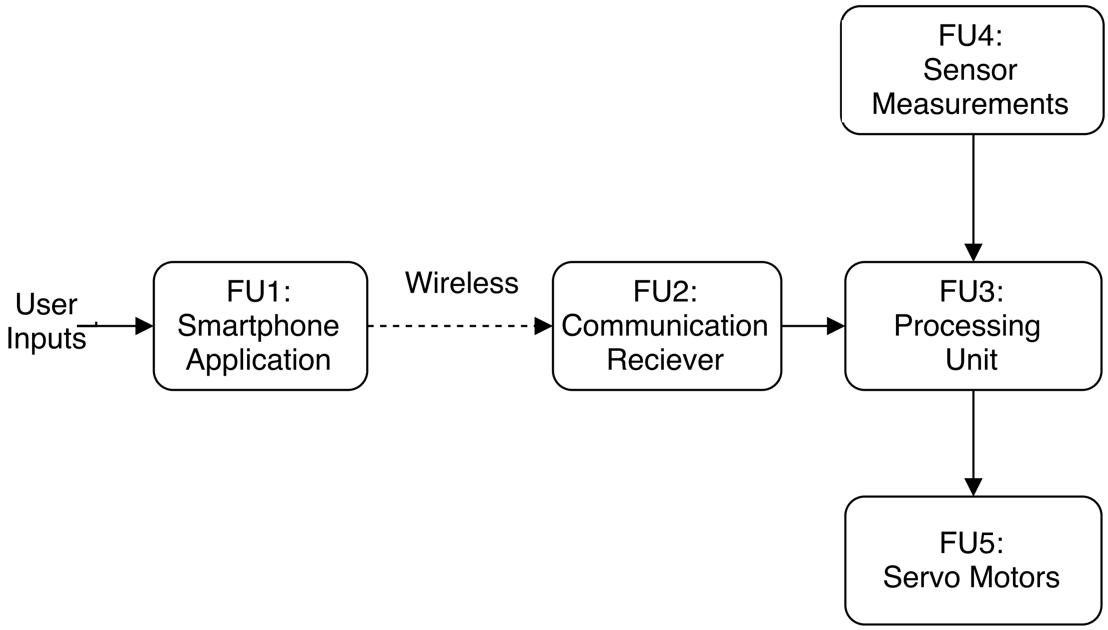


Figure 1. Functional block diagram of the product.

In Figure 1 above, the process is shown to begin with the input from the user into a smartphone application. This consists of a user interface on the screen of the smartphone that the user can interact with. The user interface will allow the user to make the robot translate as well as rotate. This information is sent to the microcontroller on the robot via wireless technology. The robot analyses this combination of translation and rotation commands and breaks it into a separate vector for each of the legs. The final position of each leg as well as the route to each individual leg destination is computed next. This makes use of a process called inverse kinematics (IK). The manoeuvres requested by the user can be realized through a series of movements with servo motors installed on the joints of the robot legs. In order to make a limb move, the exact required angle of each joint is calculated and communicated to the servo. This process is repeated continuously to make the robot react to the varying inputs of the user.

4. Specifications

Mission-critical system specifications

SPECIFICATION (IN MEASURABLE TERMS)	ORIGIN OR MOTIVATION OF THIS SPECIFICATION	HOW WILL YOU CONFIRM THAT YOUR SYSTEM COMPLIES WITH THIS SPECIFICATION?
The robot should be able to move in a given direction in 30 degree increments from a stationary position.	Proving that the robot can move straight forward, left, backwards and right (directions spaced 90 degrees apart), will prove that the robot is holonomic.	The robot will be placed on a grid on the floor and instructed to move in one of the chosen directions at a time.
The robot should be able to rotate 90 degrees without translating the centre of the body by more than 5% of the body diameter.	Rotating 90 degrees proves that the robot does not suffer from the parallel parking problem. 5% of the robot diameter can still be considered negligible and prove that no translation is required to rotate.	The robot will be placed on the grid on the floor and instructed to rotate. The translation of the centre of the body will be noted.
The time it requires to complete a manoeuvre should not vary more than 25% between smooth and rough surfaces.	A variation of less than 25% in time can still be considered to be a similar time. This proves that the robot can move over different surfaces with similar difficulty.	The robot will be instructed to complete a specific set of movements on a flat smooth surface and the time to completion will be recorded. Various obstacles will be placed in the way of the robot (small blocks, sand, etc.), and the robot will be instructed to repeat the specific set of instructions and the time difference will be noted.
The robot should be able to walk at a speed of at least 100mm/s.	100mm/s is a reasonable speed for the scale of the robot. This proves that all of the subsystems function together well.	The robot will be instructed to walk at maximum speed in a straight line for one meter. The time to completion will be measured.

The robot should be able to walk on a surface with a 10% incline without falling over.	An incline of 10% proves that the robot platform is stable while not putting too much strain on the servo motors.	The robot will be placed on a slanted surface and instructed to translate and rotate. It should not fall over while performing these manoeuvres.

Table 1. Mission-critical system specification

Field conditions

REQUIREMENT	SPECIFICATION (IN MEASURABLE TERMS)
The robot should be in range of the wireless smartphone controller.	For wireless communication to work reliably, the user with the smartphone should be less than 10m from the robot.
To prevent falling over, the robot should walk on surfaces close to horizontal.	The robot should not be operated on surfaces with an incline of more than 10%
The robot should stay dry to protect electronics.	The robot should never be operated near water or wet surfaces.
The robot should work in normal temperature conditions for South Africa.	The robot should be operated in the temperature range 10^{o}C to 40^{o}C.

Table 2. Field conditions

Functional unit specifications

SPECIFICATION	ORIGIN OR MOTIVATION
FU1. The smartphone application should communicate commands from the user interface to the robot at a frequency of at least 10 Hz.	The robot should be in constant communication with the smartphone application to update the trajectory vectors. An update frequency of 10 Hz will update the robot fast enough to make the robot feel responsive.

FU2. The inverse kinematics calculator of the robot should be able to calculate the joint positions correctly to move each leg to the desired location.	The joints should all be calculated correctly in order for the robot to make the correct set of movements to walk.
FU3. The servo motor angles should all be within 5 degrees from the calculated values.	The servo angles have to be accurate in order to make the robot walk predictably. A tolerance of 5 degrees can still be considered accurate for hobbyist servo motors.

Table 3. Functional unit specifications

5. Deliverables

Technical deliverables

DELIVERABLE	DESIGNED AND IMPLEMENTED BY STUDENT	OFF-THE-SHELF
Microcontroller for control of the robot.		X
Control code for inverse kinematics and servo control.	X	
Android application with a user interface for control of the robot.	X	
Wireless module for communication between the smartphone interface and the robot		X
Circuits implemented on PCB for interfacing all hardware with the microcontroller.	X	
Servo motors for movement of the joints.		X
Robot body and legs.	X	
Simulations on all implemented software and analogue design	X	

Table 4. Deliverables

Demonstration at the examination

1. The robot will be placed on a grid on the floor and the examiners will be shown that the robot is capable of holonomic movement.
2. The centre of the robot will be noted on the grid and the examiners will see that the robot is capable of rotating around its centre without translating.
3. The robot will execute a specific set of movements while the time to completion is being recorded.
4. Small obstacles will be placed in the way of the robot to make movement more challenging. This includes small blocks to step over as well as a change in surface such as sand.
5. The robot will repeat the set of movements over the obstacles while the time to completion is measured again.

6. The examiners will see that the robot is capable of moving with similar effort over various surfaces.

6. References

[1] J. Reeds and L. Shepp, "Optimal paths for a car that goes both forwards and backwards", *Pacific Journal of Mathematics*, vol. 145, 1990.

[2] K. Tadakuma, R. Tadakuma and J. Berengeres, "Development of holonomic omnidirectional vehicle with omni-ball: Spherical wheels", *Intellegent Robots and Systems*, vol. 2007, 2007.

[3] B. Dynamics. (2012). Ls3 - legged squad support systems. Accessed 2017-04-12, [Online]. Available: `http://www.bostondynamics.com/robot-ls3.html`.

[4] M. Raibert, K. Bankespoor, G. Nelson and R. Playter. (2008). Bigdog, the rough-terrain quaduped robot. Accessed 2017-04-12, [Online]. Available: `http://www.bostondynamics.com/img/BigDog-IFAC-Apr-8-2008.pdf`.

Part 4. Main report

1. Literature study

In this section of the report a brief overview of the existing research that is relevant to this project will be given, as well as a short discussion on how this existing research will be used to aid in the design of a five legged holonomic robot during the course of this project.

1.1 Background and context

Legged vehicles are preferred to wheeled vehicles for exploration and rescue in remote areas because of their ability to cross rough terrain quickly with a severely reduced risk of getting stuck. The price to pay for this superior drive-train is far more complex electronics and movement algorithms [1]. Instead of just driving the wheel motors and steering, each limb actuator of each leg has to be controlled to move to a calculated position. While traditional wheeled vehicles use DC motors for locomotion, legged vehicles would normally make use of stepper motors, servo motors or DC motors with rotational feedback. All of these options significantly complicates the vehicle drivetrain in both electronic hardware and software. The advantage of legged vehicles in this case is that no extra mechanical or electronic systems are required for the steering of the robot, it is usually integrated in the software for the drivetrain

Inverse kinematics is the mathematical process used to calculate the required angles of the limbs of a structure to reach a specific set of coordinates. This is the functional inverse of the forward kinematic process. This is the process of using known angles for bends in an articulated design such as a robotic leg or arm and finding the Cartesian coordinates of any of the segments, as a result of the angles originally used. The preferred method, however is to plan to move to a known Cartesian coordinate point and calculate the bend angles required to reach this point. In the paper [2], a method is discussed for solving the inverse kinematic equations involved in a seven degrees of freedom robotic arm. This method takes into account specified minimum and maximum values for each actuator and degree of freedom in order to avoid self-collision. This is an important aspect to take into account while doing the calculations since the mathematical equations do not take the physical properties of any of the limbs of their actuators into account. The entire model is simplified to lines and points in a 3-dimensional Cartesian coordinate system. One of the most important properties that need to be adhered to manually is the angular limits of any joint. This could be because of physical construction where the segments can only bend up to a point, or because of limits inherent to the actuator. Servo motors often have a limited range of motion, for example 180 degrees.

To obtain the Cartesian coordinates desired at any given time, the method described in [1], namely Sine pattern methods is used. In [1], the method was applied to a quadruped robot.

Although the robot used in the paper had only four legs, it is possible to adapt this method to any number of legs. Adaptation is especially required for an odd number of legs because the method relies on the symmetry of a quadruped robot to schedule the lifting of the legs.

In the conference paper [3], motion planning of omnidirectional robots is discussed and a sophisticated, yet simple and efficient method of route planning is proposed. This method is based on vehicles that use three omnidirectional wheels in combination to form a resultant force vector in the desired direction. These type of vehicles differ largely in terms of locomotion when compared to the holonomic legged design used in this project, but there are a few key similarities. Both these designs can move holonomically, therefore they have the ability to move in a straight line while rotating, move in an arc without rotating, or any combination of the two.

A team from Instituto Tecnologico de la Laguna in Mexico designed a hexapod robot [4] similar to the robot designed in this project. The purpose of the hexapod was to investigate its potential for use in exploration of areas that are hard to reach by any commonly used means of transportation. Legged vehicles are more suited to cross rough terrain, but rough terrain complicates the design of the drive-train. In applications where the surface is smooth, or close to smooth, open loop control can be applied where the leg is simply moved to the desired position and it is assumed by the designer that the robot foot is making contact with the ground at this point, therefore supporting part of the distributed robot weight. When the robot is crossing rough terrain, where the surface consists of mainly bumps and holes, this assumption could be false. In such a scenario, the robot could lift a leg while under the impression that its weight is being supported by the other legs. If this is not the case, the robot could fall over and possibly damage itself or be unable to rectify itself.

To avoid this problem, closed loop control should be used in the height positioning of the legs. This involves having a sensor in the system that could provide feedback on the state of the foot. The hexapod in [4] used miniature resistive force sensors attached to the bottom of each foot of the robot. The robot therefore has the ability to take analogue measurements from these sensors and determine the weight distribution of the individual feet of the robot. This data is used to confirm that all robot feet are making contact with the surface and correct the situation if this is not the case. This type of feedback is also useful when the robot is operated on slanted surfaces because of the effect that the center of gravity has on a slanted surface.

In the article [5], the development of an omnidirectional legged robot, named MORITZ, that climbs though pipe networks is discussed. Even though the configuration of this robot differs very much from the five legged robot developed in this project, there are still some common problems that could be helpful. The article discusses the importance of a planned gait pattern. In the case of MORITZ, the gait pattern is even more important because a small decrease in traction could result in the robot falling. An important note in this article, that also applies to this project, is that the calculations of forces and weight should not be done with all of the legs able to help carry the load. This is necessary for when the robot is not carrying a load while stationary, but walking/crawling. In this case the robot has to lift at least one leg at a time and therefore the full load should be supported by the remaining legs.

If the servo motors implemented in the robot, built in this project, is not capable of easily lifting the robot with at least one leg free, torsional springs could be used. The article [6] discusses the use of torsional springs in a robot in order to keep tension on the walls of a pipe. Torsional springs are the angular equivalent of normal tension or compression springs and deliver force as a linear function of angle. The idea behind the use of these springs is that instead of the servo having to apply all of the force required all the time, the spring does a substantial part of this and the force required by the servo is significantly reduced.

1.2 Application of background to this project

Inverse kinematics as described in [2] is used extensively in this project because of its ability to easily transform desired Cartesian coordinates, into the required actuator angles. This method can be applied to any system with specified degrees of freedom and can therefore be simplified to solve the inverse kinematic equations for the system designed in this project.
The robot built in this project is not designed for highly optimized motion in one (forward) direction, but rather with a focus on holonomic motion. This means that a complex motion planning algorithm is not really necessary since the motions will likely be short, slow and frequently varying direction. The symmetry required by the method outlined in [1] does not exist in a robot with five legs and it is therefore not worth adapting this complex algorithm. A different, simplified scheduling technique is therefore implemented in the design of this robot. The method used will rely on information from the current position of the legs as well as the safe limits of leg movement.

The robot discussed in [3] is also capable of holonomic movement. This similarity means that some of the methods discussed and applied in [3] can be used to aid in the design of the algorithm used in this project.

The paper [4] makes the case that a robot like this should be designed in a closed loop control system configuration. This means that some feedback information is required for making decisions as well as confirming that the motions have been executed successfully. A feedback sensor will be included in the design of the five legged holonomic robot in this project.

If the robot is operating on a slanted surface, without it being aware of this, while the centre of gravity shifts over the lowest contact making foot, the robot could fall over even if all of its legs are making contact with the surface. In a paper on reactive robot navigation [7], it is proposed that the use of a digital inclinometer can aid in solving this problem. The sensor provides information on the current tilt of the robot in two dimensions. This sensor in combination with the feedback sensors on the robot feet can be used to ensure that the robot levels itself automatically to avoid tipping over. The data collected from this sensor can also be used to collect information on the terrain. In the journal article [7], this data is used for hill climbing as well as finding valleys in unexplored areas. In order to enable the robot designed in this project to walk on slanted surfaces, a digital inclinometer could be used. Some of the reactive navigation techniques discussed in [7] could also be implemented to aid the robot in navigating on slanted planes.

In a paper on the effects of slippery surfaces on biped robots [8], methods on avoiding falling over of a biped robot is investigated. Since these robots have to balance themselves, to stay upright, an unforeseen slippery patch on a surface could be fatal for the robot. If it were possible to foresee a slippery surface, slowing down the walking gait and increasing foot surface would help increase the traction of the robot and could therefore lower the risk of slipping. Since a five legged robot is inherently stable and there is no balancing required, slippery surfaces may influence the traction of the robot but there is very low risk of falling over on level surfaces that are slippery. It is therefore suitable to just slow down movements on slippery surfaces to increase the traction where possible.

If the robot is not able to stand properly with the chosen servo motors, torsional springs could be used similar to [6] to lift most of the robot's weight and reduce strain on the servo motors.

2. Approach

This section outlines the initial approach to solve the functions shown in the functional block diagram. The functional block diagram of the project can be found in Figure 1 (in Part 3) in this document.

The core of the functional block diagram is the processing unit, which is implemented in the form of a 32 bit microcontroller with an algorithm embedded in firmware. The microcontroller that will be implemented in the final design will be determined by the amount of digital inputs and outputs (IO) required, as well as the amount of mathematical equations required per second.

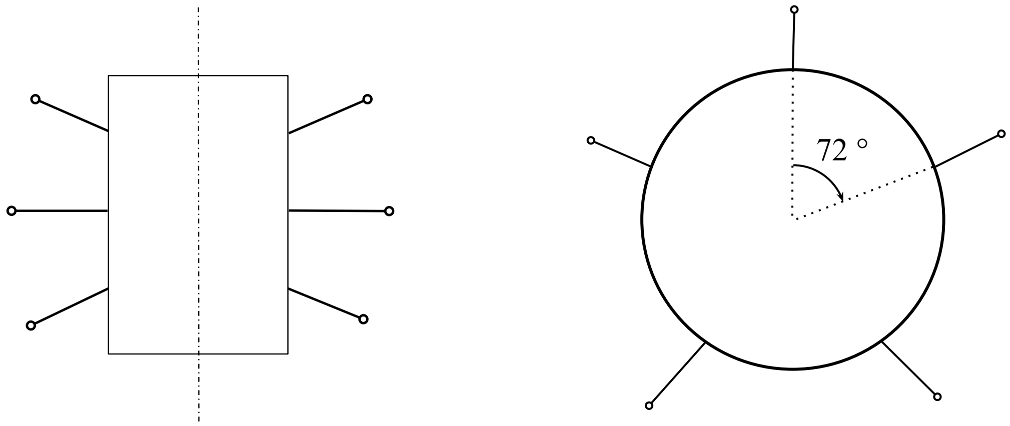
The microcontroller will act on instructions received through a wireless communication channel that originate from a smartphone application. The application will only act as a user interface for the robot and will output basic instructions for the robot to follow.

2.1 Design alternatives

The two wireless technologies built into most modern smartphones that best suit the needs of the communication channel is Wi-Fi and Bluetooth. Journal article [9] makes some comparisons between these two similar but different technologies. Wi-Fi has a much higher bandwidth than Bluetooth as well as providing for a far superior range. Wi-Fi networks are generally much more complex than Bluetooth networks and are therefore not generally suited for low level implementation. However, Bluetooth is better suited for peer to peer communications, whereas Wi-Fi is mostly intended to be used with a router. Network security for Wi-Fi is much more advanced and therefore much more secure than that of Bluetooth.

After the commands are passed from the smartphone to the microcontroller, the microcontroller interfaces to actuators to execute the movements desired by the user. When considering possible actuators that can be used, three possible solutions come to mind. These are servo motors, geared DC motors and stepper motors. Geared DC motors are small, light and easy to implement but lack feedback on shaft position. Such feedback needs to be implemented manually to be able to make controlled movements. Stepper motors have fixed step resolution, therefore making controlled movements is easier once the current position is known. The drawback of these motors is that a homing mechanism and routine needs to be implemented to find a reference. Stepper motors are also heavy and require high current even when stationary. Servo motors are basically geared DC motors with a positional feedback control system built-in. Once the motor receives power and a desired position, it will attempt to reach the desired position. The disadvantage is that these have a limited range of motion.

The overall shape of the robot, more specifically, the placement of legs around the robot will



(a) Symmetric chassis layout. **(b) Equally spaced chassis layout.**

Figure 2. Examples of two different chassis layout options for legged robots.

greatly influence the robot's performance when moving forward in a straight line, sideways or over obstacles.

A popular design in hexapod robots is placing the legs in groups of three on opposite sides. This means that the legs are all aligned in a similar direction while the body is usually a long rectangular shape, similar to that found on insects. The advantage of this design lies mainly in the ability to move forward very quickly since the legs are placed optimally for forward motion. While holonomic movement is possible, it is usually much slower than forward or backward motion. This design is usually more suited to robots with an even number of legs due to the symmetry of the design.

An alternative design approach is to space the legs evenly around the robot's chassis. The robot's chassis would normally be round, or a polygon shape, with the same number of sides as legs. In the case of a five legged robot, the legs would be positioned $\frac{360^o}{5} = 72^o$ apart. This equally spaced design approach has the disadvantage of not being particularly fast in any given direction, but the advantage that it can manage the maximum speed in a specific direction. Figure 2 shows an examples of both layouts.

2.2 Preferred solution

Since the data transfer in the communication channel between the smartphone and the robot consists only of simple commands and the required bandwidth is therefore low, Bluetooth will be used instead of Wi-Fi. Bluetooth has the advantage of being less expensive and far less complicated to implement. The added security that Wi-Fi offers is not required for this application. The disadvantage of using Bluetooth instead of Wi-Fi is the very limited range that Bluetooth offers.

The commands received via Bluetooth will be actuated with the use of servo motors. These compact units are ideal for this scenario as they have all of the advantages of geared DC motors with all of the required control systems built-in. Servo motors have the advantage that they are usually light for the torque they can provide, which is ideal for a robot where weight is often a problem.

This will all be implemented in a circular shaped chassis with equally spaced legs as this design is better suited for holonomic movements than the symmetric design as discussed above.

3. Design and implementation

3.1 Background

A large part of the design of a legged robot depends on the design of a simple leg - more specifically the degrees of freedom that a single leg has. The degrees of freedom that a single leg has is determined by the amount of dimensions that the leg can make a controlled movement in, independently from any other dimensions. There are two common underlying designs in robot leg design - these are for two and three degrees of freedom respectively.

Figure 3 shows an example of a common design that has two degrees of freedom. The design therefore contains two joints in one leg. In this example, the first joint is positioned vertically to form the hip op the robot leg. This allows for the side-to-side motion required for walking. The second joint is positioned horizontally to allow the lowest limb to move up and down. The two degrees of freedom controlled by this leg design is therefore the horizontal angle of the leg and the height of the foot. These two (and only these two) parameters can therefore be controlled completely independently.

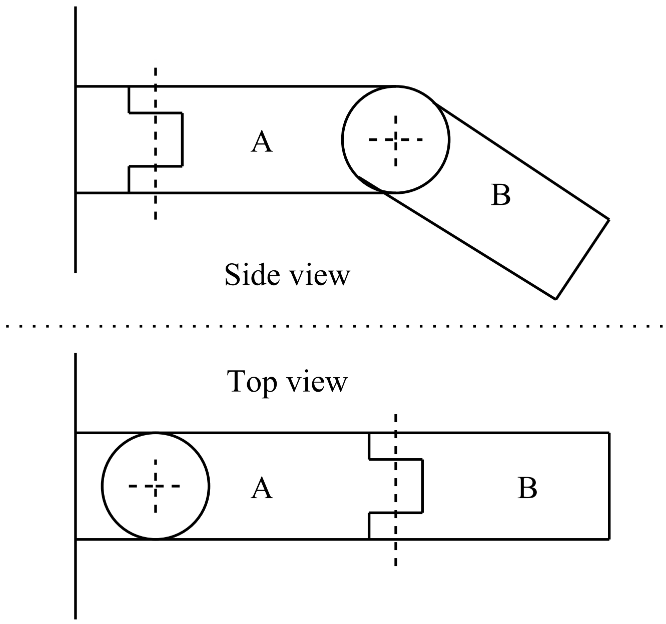


Figure 3. Example of a leg design with two degrees of freedom.

An example of a robotic leg with three degrees of freedom can be seen in Figure 4. This design is similar to that of the leg with two degrees of freedom seen in Figure 3, with the only addition being a second horizontal joint further down from the first. This additional limb means that the horizontal distance from the foot to the hip can be controlled as well as the height of the foot. These two controlled parameters, together with the horizontal leg angle which can also be controlled independently as in the case of the two degrees of freedom design, means that this design has a total of three degrees of freedom.

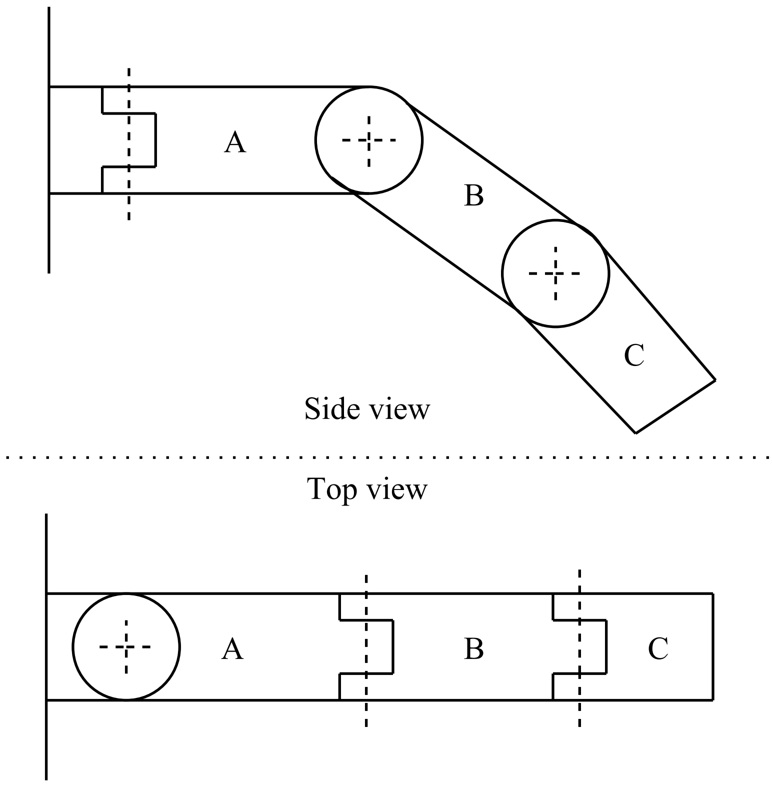


Figure 4. Example of a leg design with three degrees of freedom.

The design with three degrees of freedom has the advantage of being able to lift a leg without altering the horizontal position of the leg. This means that the robot is able to walk without altering the height of the body of the robot. The robot is therefore able to cross rougher terrain much better because of the ability to alter the height of the robot feet as the terrain requires. With the design that has two degrees of freedom, the foot height is a function of the horizontal extension of the leg. With this design the main advantage is the simplicity - both mechanically and in software. The cost and power consumption will also be much lower because of the reduced amount of actuators.

Due to the much greater flexibility of the design with three degrees of freedom and the ability to cross rougher terrain, this will be the platform implemented in the final design.

3.2 Theoretical analysis and modelling

Since each leg has three degrees of freedom, there are three angles that need to be controlled in order to move the leg to a desired position. These will be referred to as theta (θ), phi(ϕ) and alpha(α) throughout this document. Theta describes the horizontal angle between the tangent of the body and leg segment A as shown in Figure 5.

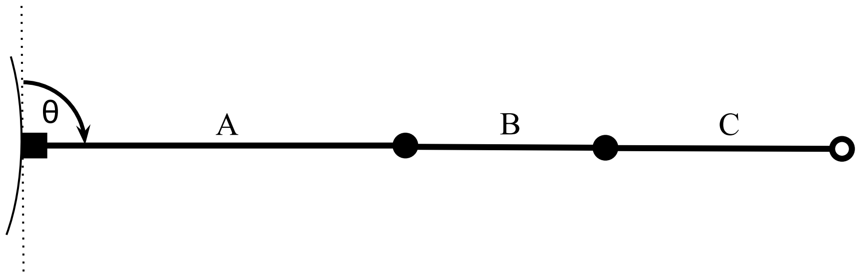


Figure 5. Top view of the leg design model for mathematical analysis.

Phi describes the angle between leg segments A and B viewed directly from the side. In the same plane, alpha describes the angle between leg segments B and C. Both of these angles can be seen in Figure 6.

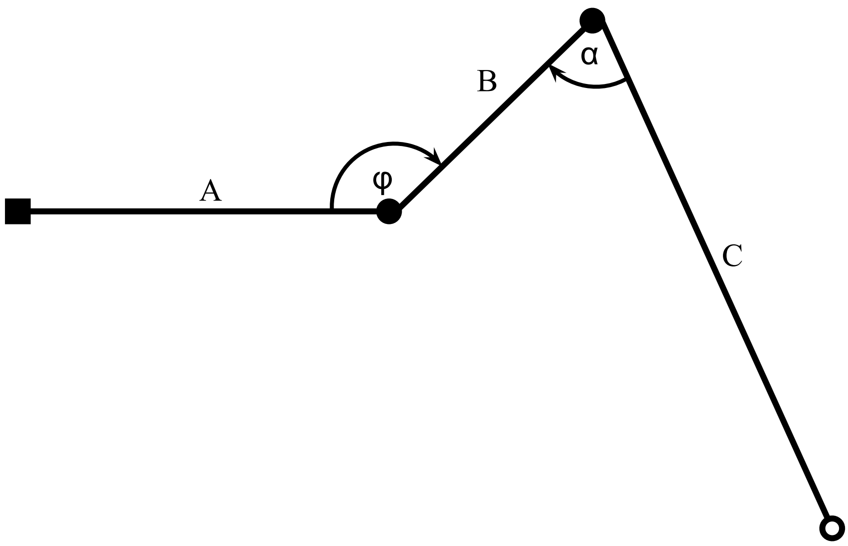


Figure 6. Side view of the leg design model for mathematical analysis.

In order to control the robot in a Cartesian world, it is necessary to find a transfer function for a leg that can translate a coordinate vector in the domain $[\theta, \phi, \alpha]$ to a vector in the domain

$[x, y, z]$. This is done through mathematical analysis of a single leg.

By using Figure 5 and some simple trigonometry, theta can be found to be

$$\theta = arctan\left(\frac{x}{y}\right) \tag{1}$$

where x, y and z are the desired coordinates for the foot of the relevant leg. It is important to note that the hip of the leg where section A meets the chassis is the origin of the Cartesian system for this analysis. This can be seen in Figure 7

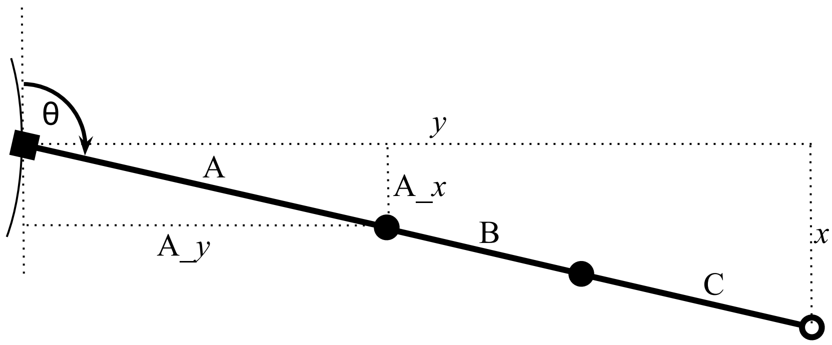


Figure 7. Annotated top view of the leg design model for mathematical analysis.

In order to do an analysis of phi and alpha, it is necessary to add a few construction lines to the generic model in Figure 6. The additions can be seen in Figure 8 and the additions are explained below.

From Figure 8:

- Z_{body} is the nominal height difference between the robot chassis and ground. This value is constant.
- *Reach* is the horizontal distance between the hip and the foot.
- D is an imaginary line between joint 2 and the foot that completes the triangle BCD that will form the basis of this analysis.
- E_x is the horizontal distance between joint 2 and the foot.
- E_y is the horizontal line that completes triangle DE_xE_y. Quantitatively $E_y = Z_{body}$ since the ground is assumed to be level in this analysis.
- c is the angle between B and D.
- b is the angle between C and D.

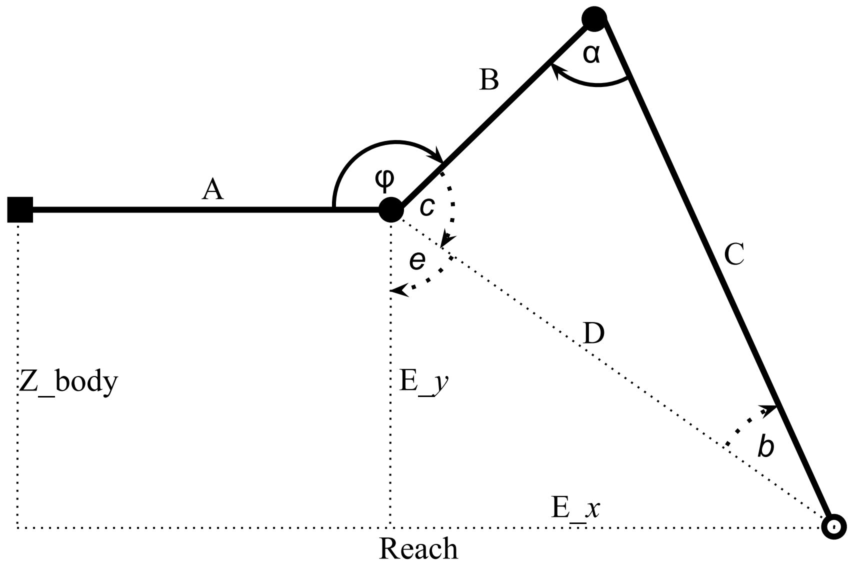


Figure 8. Annotated side view of the leg design model for mathematical analysis.

- e is the angle between E_y and D.

$$\therefore Reach = \sqrt{x^2 + y^2} \tag{2}$$

By using triangle BCD and the Cosine rule,

$$cos(\alpha) = \frac{B^2 + C^2 - D^2}{2 \times B \times C} \tag{3}$$

$$\therefore \alpha = cos^{-1}\left(\frac{B^2 + C^2 - D^2}{2 \times B \times C}\right). \tag{4}$$

The value of E_x can be calculated similar to that of reach, but first finding the x and y component lengths of segment A as shown in Figure 7.

$$A_x = A \times sin(\theta) \tag{5}$$

$$A_y = A \times cos(\theta) \tag{6}$$

$$E_x = \sqrt{(x - A_x)^2 + (y - A_y)^2} \tag{7}$$

The Sine rule can then be used on triangle DE_xE_y to find,

$$\frac{sin(c)}{C} = \frac{sin(\alpha)}{D} \tag{8}$$

$$\therefore sin(c) = C \times \frac{sin(\alpha)}{D} \tag{9}$$

$$\therefore c = sin^{-1}\left(C \times \frac{sin(\alpha)}{D}\right). \tag{10}$$

Angle e can easily be found as

$$e = tan^{-1}\left(\frac{E_x}{E_y}\right). \tag{11}$$

With all of this calculated, phi can be calculated as

$$\phi = 270^o - c - e. \tag{12}$$

The analysis above is valid for the generic 3 segment design shown in Figures 5 and 6 where the hip is at the origin of the Cartesian system. For the real robot, the calculations need to be performed for five legs. In order to avoid using five different Cartesian systems, it is necessary to be able to rotate and translate this generic model to suit the requirements of any of the legs.

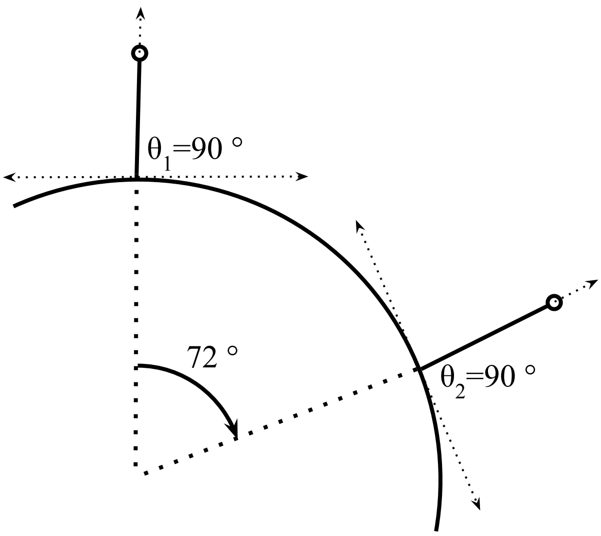


Figure 9. Top view of the robot showing individual Cartesian systems for individual legs.

Figure 9 illustrates that both leg 1 and 2 have a theta angle of $\theta = 90^o$, yet they are clearly not facing the same direction. This is because of this separate coordinate system that each of the legs have in order to do the inverse kinematic calculations. In order to do calculations on the whole robot, the centre of the robot is used as the origin.

If *leg* denotes the number of the desired leg, a leg's coordinate system can be rotated by using

$$(x', y') = (x \times cos(\gamma) - y \times sin(\gamma), x \times sin(\gamma) + y \times cos(\gamma)) \tag{13}$$

$$\text{where } \gamma = (leg - 1) \times 72^o. \tag{14}$$

For the robot to be able to walk, all five legs should be working together to move in a specific direction. The use of vectors work well since they can be added easily and contain both direction and magnitude information. The movement of each foot can be constrained for each of the bends (θ, ϕ, α) to limit the angle to a realistic value for the servo. These limitations can be seen in Figure 10 as dotted lines. The dashed line represents the vector for movement. The movement vector is limited by the boundary.

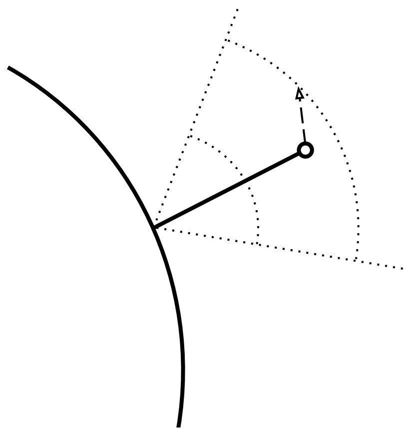


Figure 10. Top view of the robot showing the addition of vectors and leg boundaries.

All of the legs move together towards a point inside or on the edge of each individual perimeter. The direction of the movement of the robot can be calculated as

$$angle = arctan\left(\frac{y}{x}\right), \tag{15}$$

where x and y denote the input from the user interface. The magnitude can be found to be

$$magnitude = \sqrt{x^2 + y^2}. \tag{16}$$

The magnitude and direction can be used to check if the vector fits inside the perimeter.

All of the calculations above only account for translation, that is moving forward, backward, left, right or a combination of the above. Being able to move truly holonomically means being able to rotate with or without translating. The vectors used to calculate the trajectory of each leg works well for representing translation, but does not work so well for rotation. Rotation can instead be expressed as a scalar value, which has magnitude and a sign indicating the direction of rotation. A positive value indicates a clockwise rotation. When these rotation vectors are applied to the individual legs, it results in a different vector being added to each leg indicating the trajectory required by each leg to rotate the robot body. Figure 11 shows the effect of adding a rotation scalar to the movement. The dashed arrows indicate the trajectory of each individual leg.

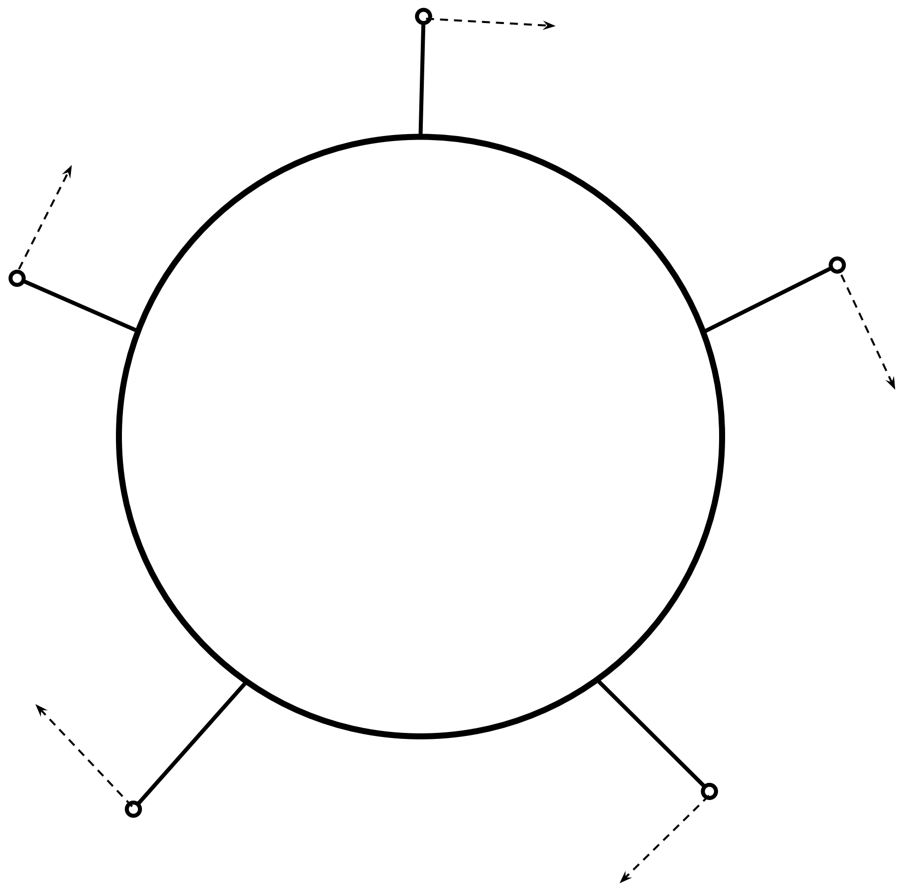


Figure 11. Top view of the robot showing the addition of vectors for rotation about the origin.

If each leg were instructed to follow the dashed arrows shown in Figure 11, the legs would fight each other for grip since the vectors will force them to spread out. Simply using the instantaneous trajectory as the movement vector will not work like it did in the case of translational movement. Instead the trajectory will constantly be changing to keep rotating the body while keeping the feet in their original positions. This change can be seen in Figure 12. The dotted lines represent this constantly changing vector. This can be calculated by rotating the foot position around the origin of the robot. It is important to note that although the legs will be spreading out as the robot body rotates, the feet should stay exactly where they are until the foot is out of the perimeter that the system allows. At this point the robot will lift up the leg and place it in the neutral position to allow for further rotation.

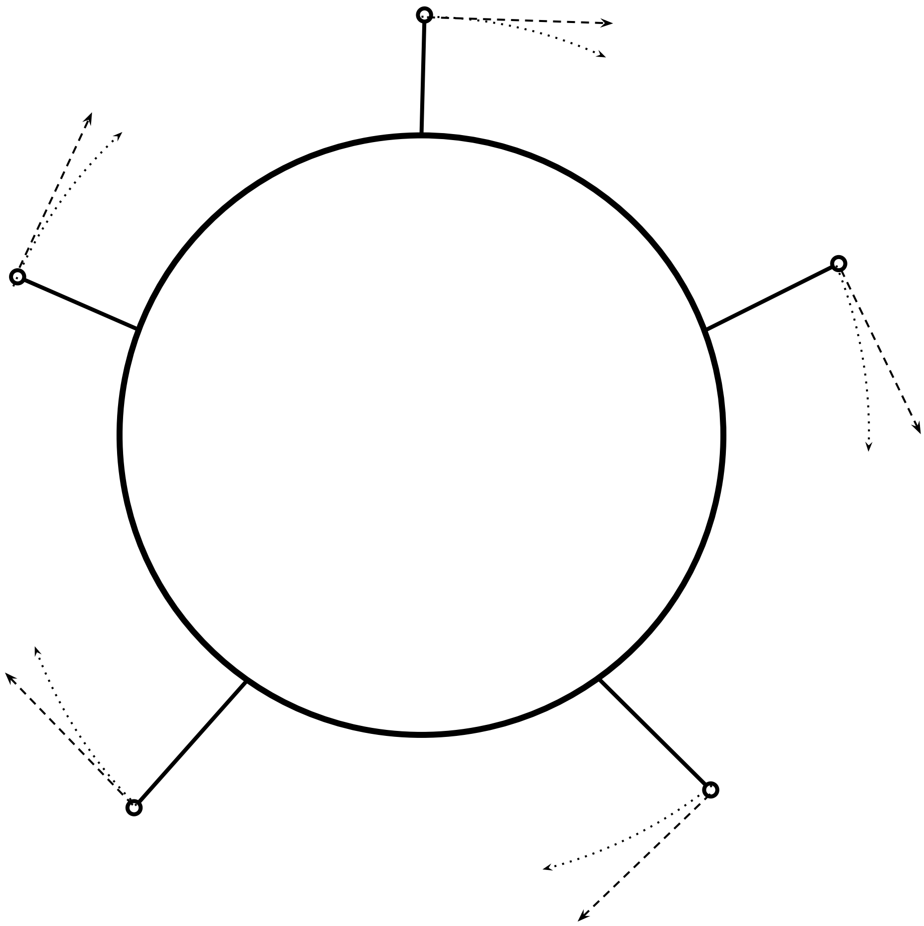


Figure 12. Top view of the robot showing the addition of curved approximation for rotation about the origin.

Since a curve cannot be added to the vector of translation, the curve will need to be broken up into smaller segments of straight pieces approximating the curved line. These straight pieces can then be added to the translation as vectors. Rotating the coordinates around the robot is not a simple task because of the shifting Cartesian systems. In order to rotate about a point that is not the origin of the plane, the coordinate system needs to be translated to move the origin, rotated around this origin and then shifted back to the original origin.

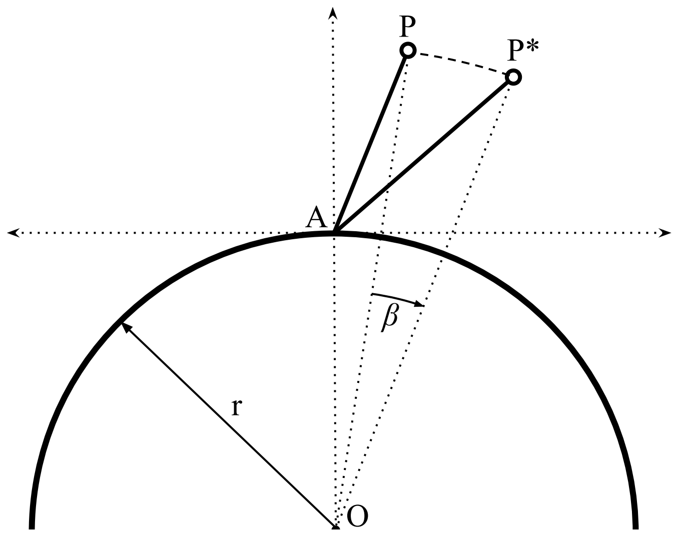


Figure 13. Illustration showing the procedure of rotation about the origin.

Figure 13 attempts to explain the robot rotation procedure. The following features appear on the sketch:

- Point O is the origin, the centre of the robot.
- Point A is the shoulder of the relevant leg.
- Point P is the position coordinate of the foot before any rotation.
- Point $P*$ is the position coordinate of the foot after rotation.
- β is the rotation angle.
- r is the robot radius.
- The dashed line is the rotation path of the foot.

When the robot needs to rotate, the procedure is therefore as follows.

1. Start with coordinates of point P, relative to point A which is the origin of the Cartesian system at this point.
2. Add the coordinates of P relative to A to the coordinates of A relative to O. The result is the coordinates of P relative to O.
3. Rotate through angle β using the rotation formula in Equation 13.
4. Move the origin back to A by subtracting A relative to O.

The result can be used to perform the Inverse Kinematic calculations as described in Equations 1 through 12

3.3 Simulation

In order to test the equations and design outlined in the Theory section above, the mathematics was implemented in a script written in the Python programming language and developed in the PyCharm IDE. The program consists of a GUI used to enter elementary commands, similar to that implemented in the Android application, as well as a window for plotting the result in a 3-dimensional Cartesian system. Some results showing the validity of the design equations in the previous section can be found below. All of the source code used to build this simulation can be found in Part 5 of this report on the attached optical disc.

The program flow of the software in the simulation is similar to that used for the final prototype software. More information on the software design flow can be found in Section 3.7. The main difference between the final implementation and the simulation is what is done once the angle values for theta, alpha and phi are calculated for each leg. In the final implementation the result is simply used to move the actuator. In simulation, however, the result is used to calculate the Cartesian position of each limb in order to plot it in the 3-dimensional system. It should be noted that the length of all of the limbs, as well as the robot body's radius was chosen as unity for simplicity. This is done since the purpose of the simulation is only to show the functionality of the IK. It should also be noted that although some of the legs may look crooked in the following figures, this is only a result of the viewing angle. When viewed directly from above, all limbs of a single leg align in an upright plane.

The basic user interface together with the Robot in its neutral position can be seen in Figure 14.

The orange markers found at each of the robot feet indicate the neutral position for each of the five feet. This is indicated to aid in showing the movement of the legs relative to this neutral position in Figure 14. The same applies the rest of the figures following in this section.

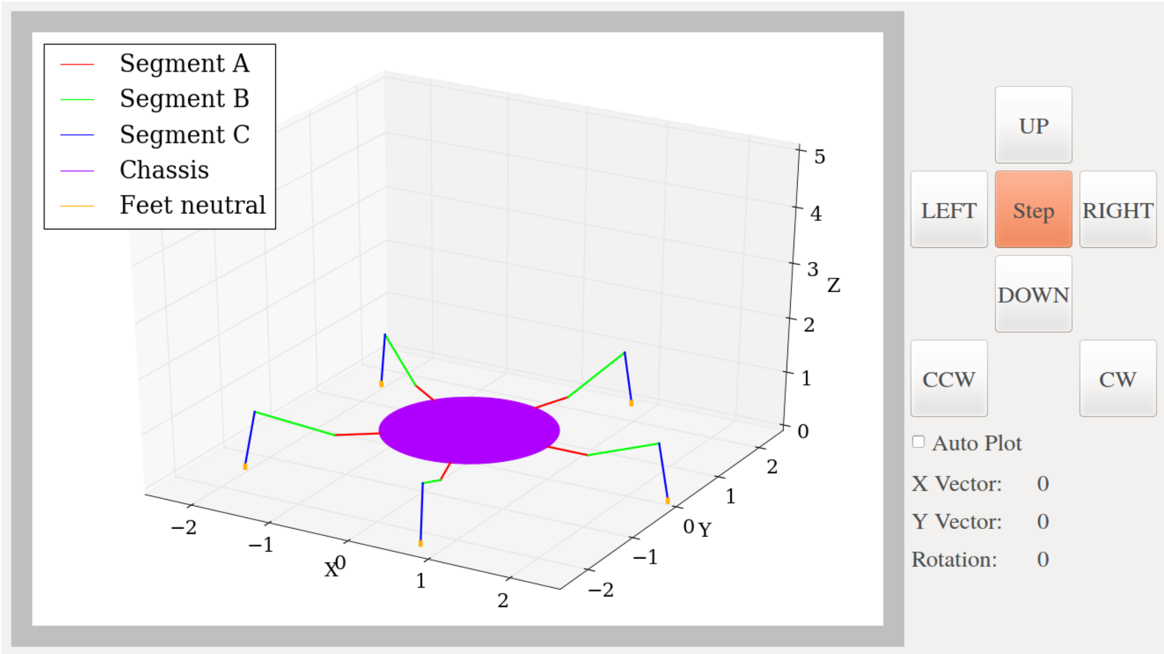


Figure 14. Simulation window showing the robot in its neutral position.

When the Y vector is changed to a positive value, the legs will all move in a negative Y-direction since the robot center is the origin. When the robot should move in a specific direction, the feet will move in the exact opposite direction to propel the robot forward. This is illustrated in Figure 15.

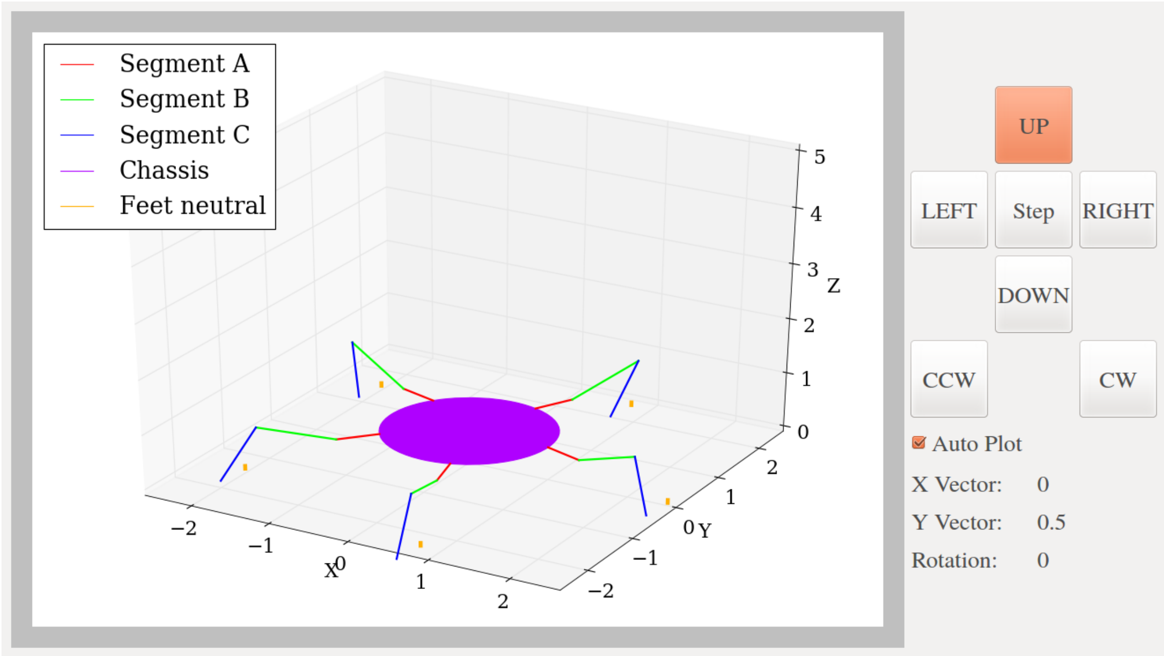


Figure 15. Simulation window showing the robot moving in a positive Y-direction.

The robot should also be able to translate in the X-direction without making a rotation first. This is illustrated in Figure 16.

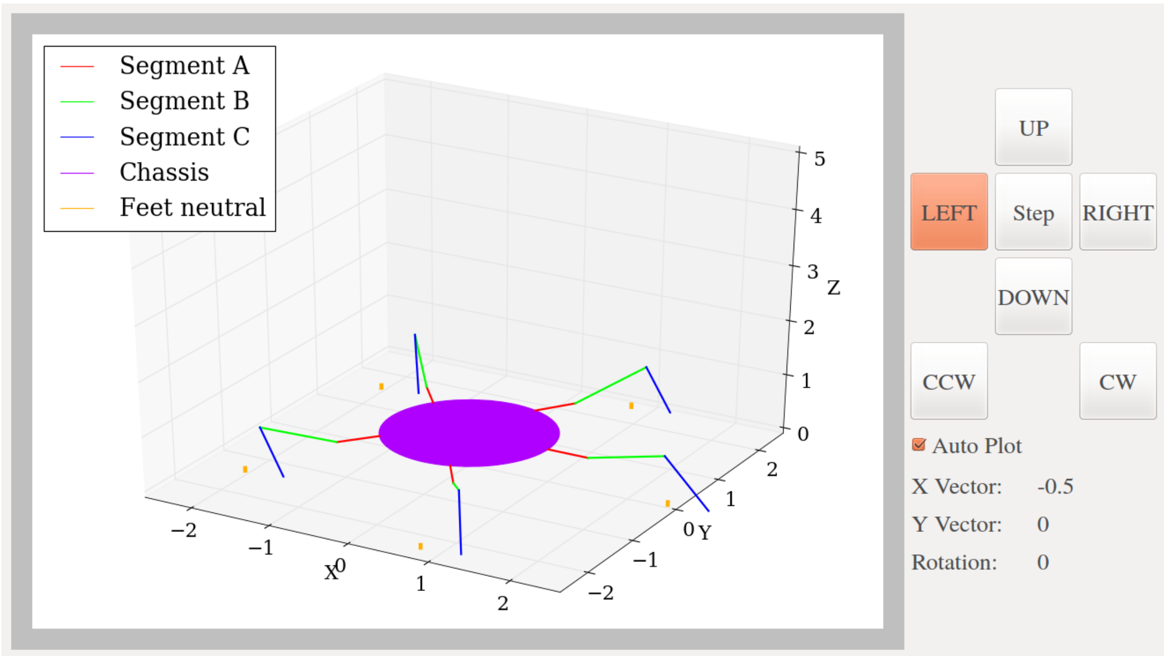


Figure 16. Simulation window showing the robot moving in a negative X-direction.

If the robot can instantaneously move in both the X and Y-directions, it should be able to move in a combination of the two without a problem. This ability is illustrated in Figure 17.

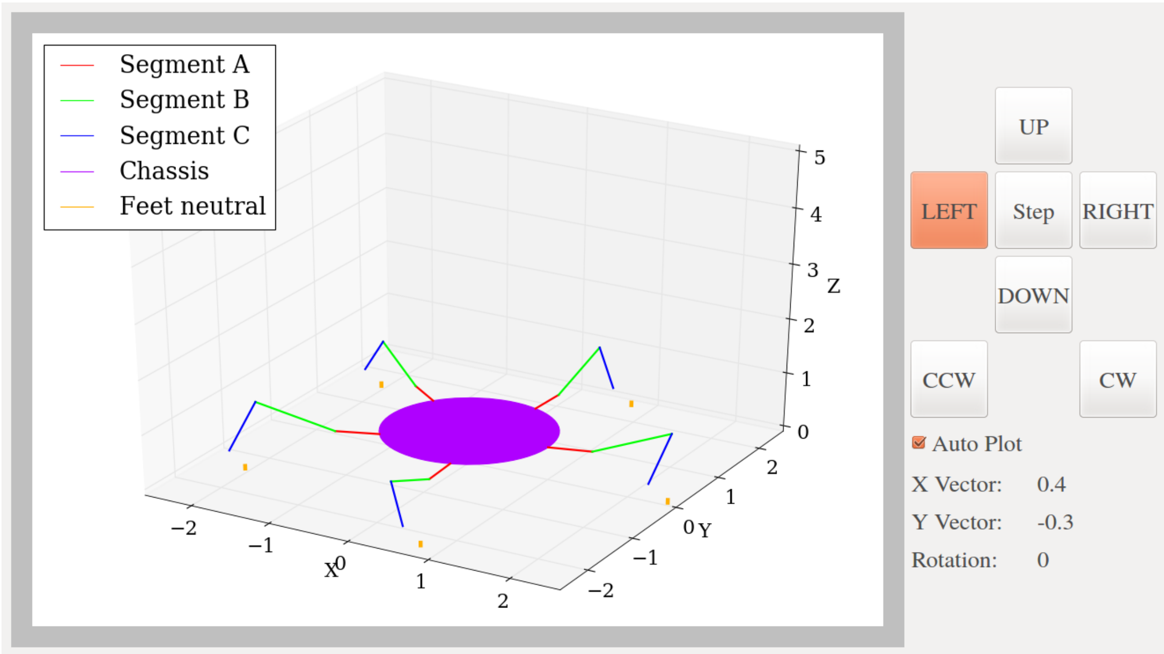


Figure 17. Simulation window showing the robot moving in a direction with a positive X and negative Y-component.

The translation appears to be working exactly as intended and designed for. The simulation is tested with a clockwise direction rotation in Figure 18.

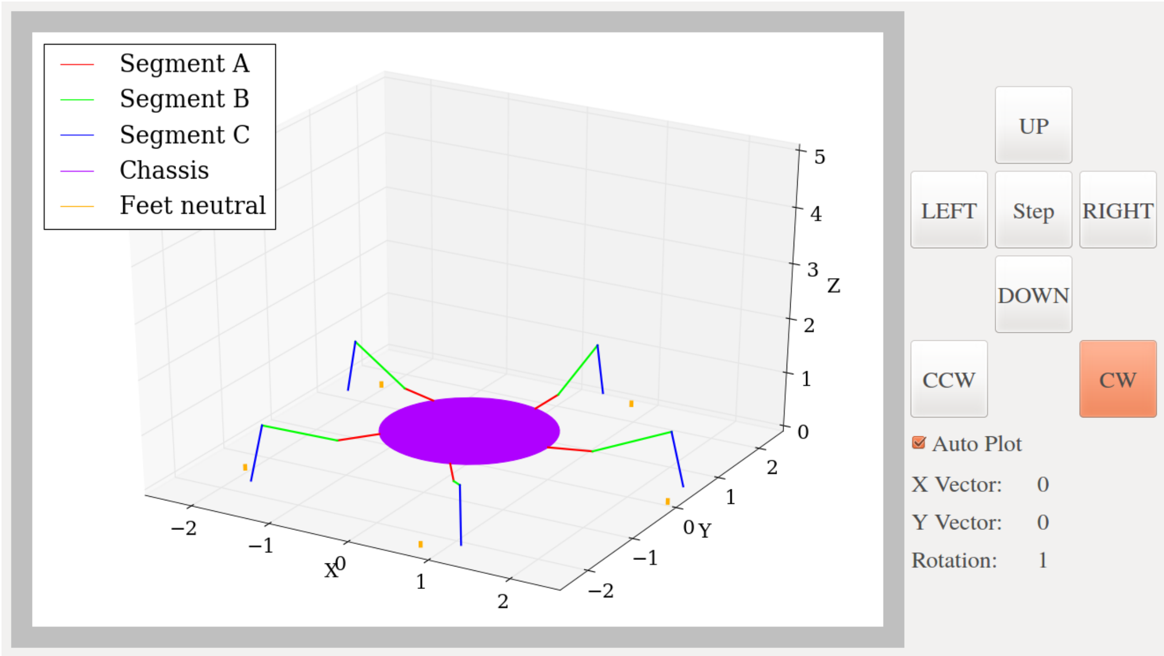


Figure 18. Simulation window showing the robot rotating in a clockwise direction.

Once the robot reaches the boundary illustrated in Figure 10 for any of its legs, it is necessary to lift the leg up, move it to a more suitable location and put it down in order to continue in the direction it is heading in. This ability is illustrated in Figure 19.

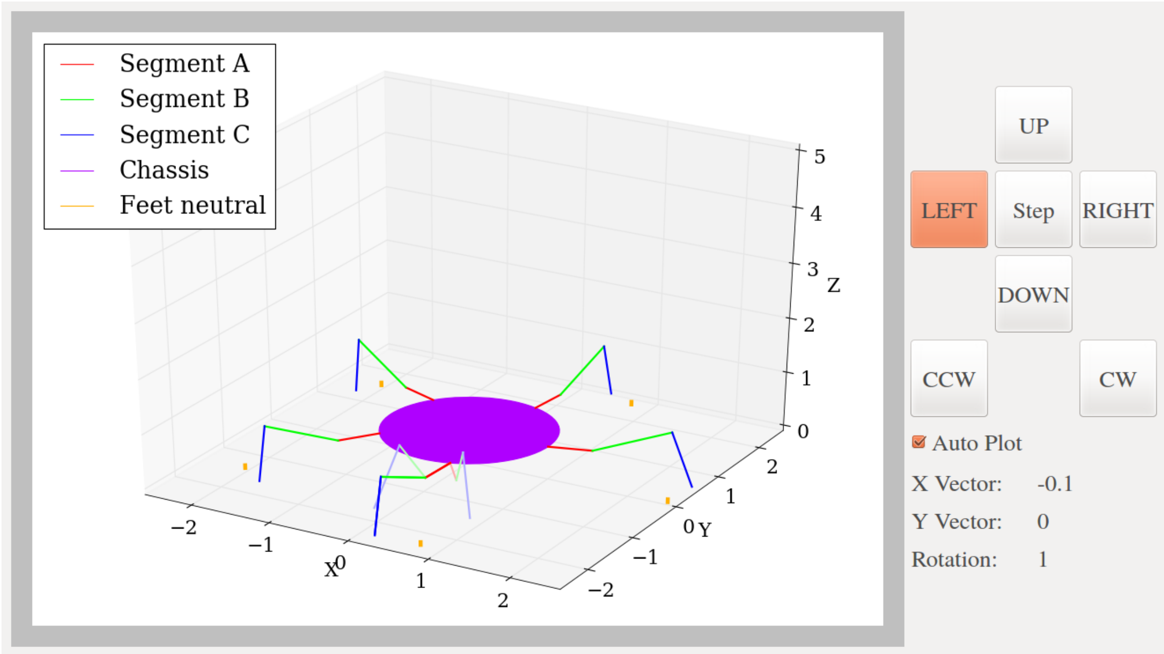


Figure 19. Simulation window showing the robot picking up a foot and placing it again.

In Figure 19, the closest leg appears three times, twice in dimmed (ghost) colours and then finally in the normal colours. What this attempts to illustrate is the resetting motion of the leg. Figure 18 shows the robot just before the resetting action. The right of the two "ghost" legs is right above the original location of the leg. This shows the position of the leg after it was lifted up. The left of the two "ghost" legs is the leg, still lifted, after being moved to a position directly over the new location. The leg plotted in full colour shows the final position of the leg. It is now able to start rotating further in the clockwise direction. Th robot will pickup and replace one leg at a time for all the legs that requires resetting and then continue.

3.4 Optimisation

The design of the algorithm used in the simulation works very well to systematically determine the 15 required servo angles. This is done from the three input values (X, Y, R) which ultimately represent the speed for the X-direction, Y-direction and Rotation. What the simulation fails to take into account is the practical time constraints necessary to make a robot work. In simulation the result was just plotted so all the movement could be done instantaneously. In practice, however, the servo motors used to control the limbs of the robot need time to adjust to the set position. These servo motors use traditional analogue proportional, integral, derivative (PID) control techniques. The derivative term means that a sudden, large change in setpoint would result in a large current surge as a result of the PID controller trying to get to the setpoint as soon as possible. If all 15 of the servo motors are updated at the same time with a large jump in position, the surge in current required from each individual servo motor would sum to a large surge, possibly causing a brief dip in the system voltage. The solution to this is to not make large changes in setpoint between updates to the servo motors, but rather smaller updates more frequently. The exact time between updates will be a function of a how long it takes the floating point arithmetic (FPU) unit of the microcontroller to do all the floating point math required by the IK algorithm once.

3.5 Electronic design

In this project, the user interface is implemented in the form of an android application. Commands are communicated to a microcontroller via a serial connection with a Bluetooth module. The microcontroller does all of the calculations and performs the necessary algorithms to determine what should be done with which servo and then communicates this to the relevant servo motor. The algorithm for movement also takes some inputs from the microcontroller into account.

The electronics to be implemented for this project is listed below.

- Power regulation and supply.
- Support electronics for the microcontroller.
- Digital input signal conditioning.
- Bluetooth module interfacing.
- Digital output driving where applicable.
- Servo power control switch

Each of these items are discussed separately below.

Power regulation and supply for the project can be divided into two parts. The first of these is the low voltage, low power part that supplies the microcontroller, Bluetooth module and support electronics. The second part is the higher voltage, high power part that supplies power exclusively to the servo motors. Both these supplies will be powered by Li-Ion 18650 cells because of their high power density. The nominal voltage of these cells are $3.7V$. The low voltage, low power supply can easily be powered from a single cell making use of a low dropout (LDO) linear regulator to provide the $3.3V$ rail required. A linear voltage regulator is not suited for the high power application, mainly for two reasons. The first of the two is that these regulators are rarely rated for use above $1.5A$. The second is that a linear regulator dissipates power in itself as heat in order to regulate voltage. This means that the voltage drop formed over the device to bring the output voltage down is dissipated in the device. The power dissipated can be quantified by

$$\begin{aligned} P &= V \times I && (17) \\ &= (V_{in} - V_{out}) \times I && (18) \end{aligned}$$

This is fine for low current applications but sufficient cooling quickly becomes a problem at currents in the Ampere range. A better solution for regulation at high power is making use of a step-down DC-DC converter. The efficiency of this topology is much greater than for linear regulators. Most are in excess of 90% if operated within the design limits. Since DC-DC converters can become very complex and it is far outside the scope of this project, a complete off-the-shelf module was implemented instead of designing and building one from first principles. Figure 20 shows the power supply layout for the robot.

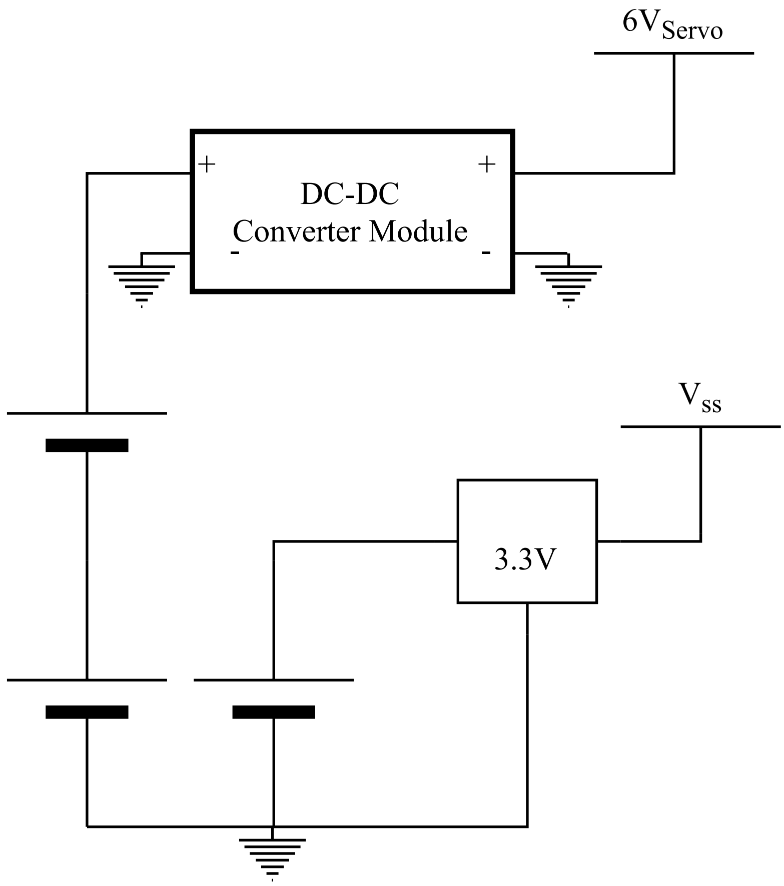


Figure 20. Diagram showing the power supply layout for the robot.

The support electronics for the microcontroller includes everything necessary for it to be able to start after power up and function normally. The application note provided by the manufacturer has detailed instructions and schematics on this and it was implemented as recommended for this project. This includes ceramic capacitors on all of the power supply pins, electrolytic capacitors on the power rails, a high frequency crystal oscillator, timing capacitors, a reset switch, various pull-up and pull-down resistors and a selector for pulling the BOOT pin high or low. More details on this can be fond in the technical documentation section of the report.

In order to protect digital input pins on the microcontroller, as well as debouncing input signals from switches, a small interface circuit is used between an input and a digital pin. This is implemented from prior knowledge gained in earlier modules. Figure 21 shows the schematic for this.

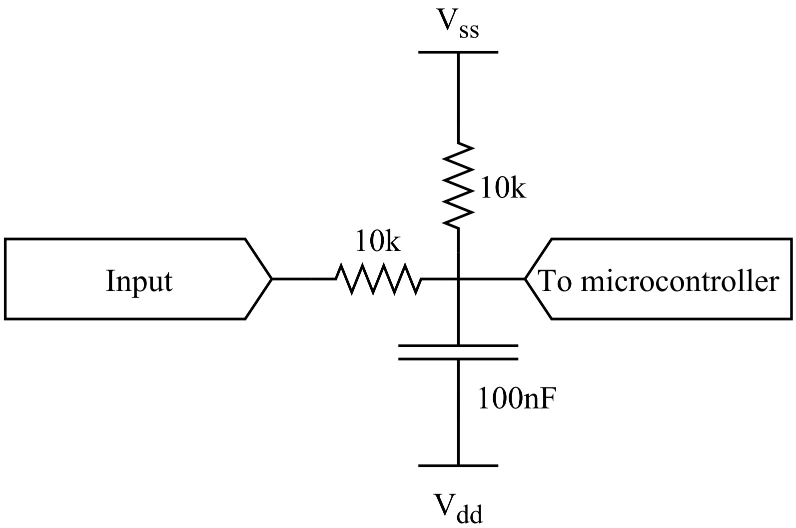


Figure 21. Diagram showing the digital input protection circuit for the robot.

The Bluetooth module used in this project does all of the Bluetooth protocol and decryption automatically and makes the data available through the serial interface. This means that the module is powered by the rails of the microcontroller and all that is further required to receive information is to connect the *TX* pin of the module to the *USART_RX* pin of the microcontroller.

In order to make the actions and decisions of the robot more apparent to the user, a red, green and blue (RGB) LED array is connected to the microcontroller. Different colours can then be used to indicate different states of the robot. Since the LEDs can't be powered directly from a digital output pin, a driver will have to be built to protect both the microcontroller and the LEDs. Figure 22 shows the schematic for this. Each LED requires a bipolar junction transistor (BJT) of type PNP. The PNP is required because the RGB LED has a common ground and therefore has to be switched on the positive side. The microcontroller is used to pull the base of the PNP high or low. The LED will then switch off or on respectively.

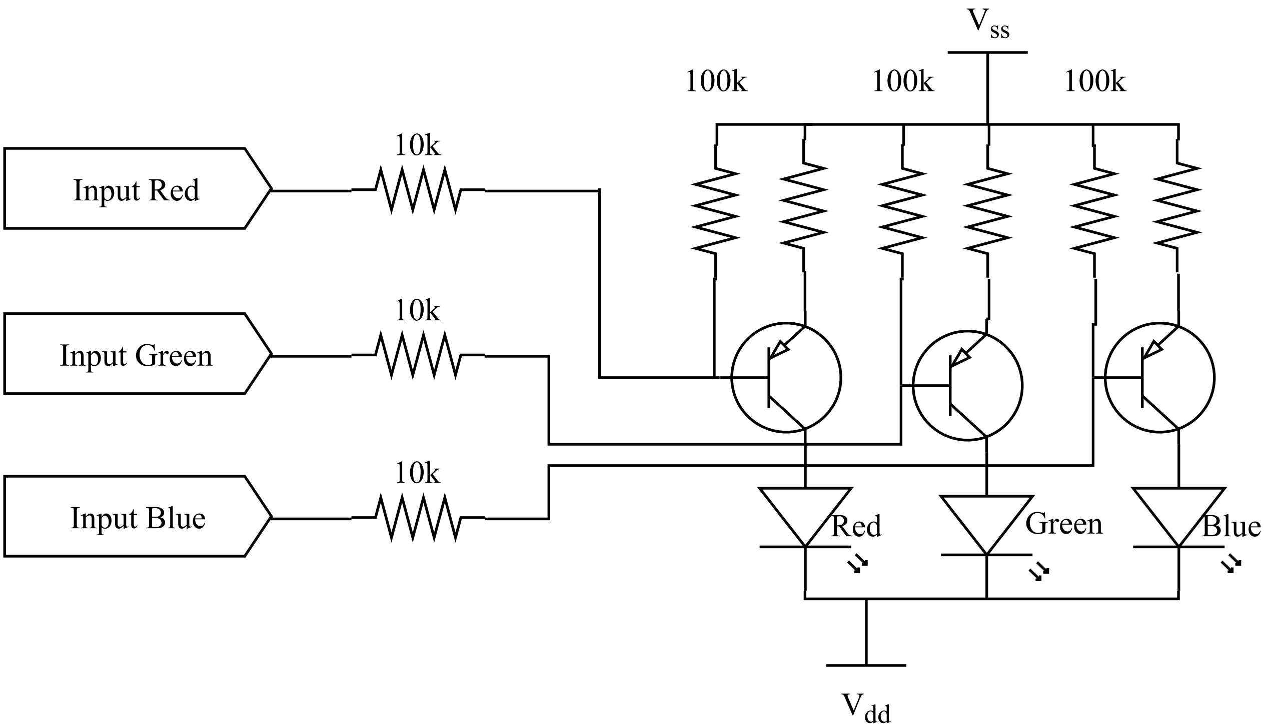


Figure 22. Diagram showing the driver circuit for the RGB LED array.

Servo motors constantly use power to hold the position they are in. When the robot is idle, waiting for a Bluetooth connection, it is unnecessary for the servo motors to be consuming current. The circuit illustrated in Figure 23 is designed to switch power to a rail dedicated to servo motor supply. It uses a P-channel MOSFET as well as a NPN BJT. The P-channel MOSFET is required to switch the positive rail with a low internal power dissipation due to a low drain-source resistance when switched on. The BJT is necessary to switch it on because the microcontroller pin on its own can't reach the 6V required to completely switch off. When the pin is low, the BJT is switched off, the MOSFET gate is pulled high by the resistor and therefore the MOSFET is off. When the pin is high, the BJT pulls the MOSFET gate low and the MOSFET is fully on, thereby turning the servo rail on.

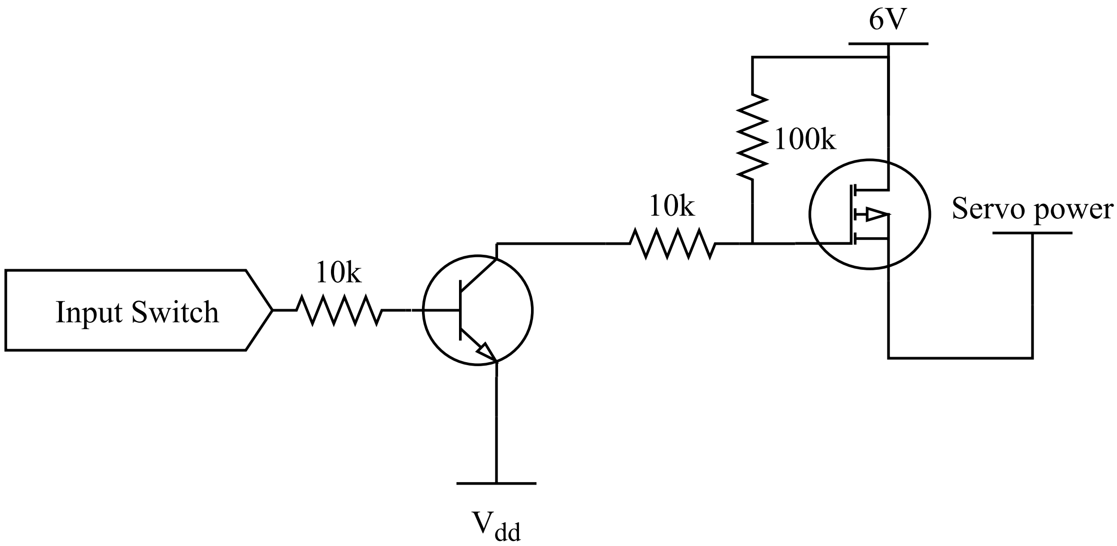


Figure 23. Diagram showing the driver circuit for the RGB LED array.

3.6 Electronic implementation

The microcontroller chosen is an STM32F576GTV6. This specific model was chosen because of its powerful FPU, high clock speed and the large pin count for possible expanding later. This unit is, however, only available in an LQFP100 package with 0.4mm pin spacing. This was soldered to a breakout board to make interfacing with the hardware easier. The breakout board was glued to a Veroboard where the rest of the electronics, discussed above, was implemented.

3.7 Software design

This section contains a summary of the development process for the software algorithm implemented in both the simulations discussed in section 3.3 and the final embedded software implementation.

If the robot attempts to lift all of the legs at the same time, the legs would not lift, the body would drop instead. The key to taking steps and therefore walking is lifting only one leg and moving it while the other legs remain in position. The trajectories calculated in section 3.2 will be used and broken into smaller steps which are executed in fixed intervals to maintain a specific speed.

Figure 24 attempts to explain the rough algorithm used to reset the legs and therefore the algorithm required for walking. This is the algorithm implemented in the Python program for simulation without the plotting functions.

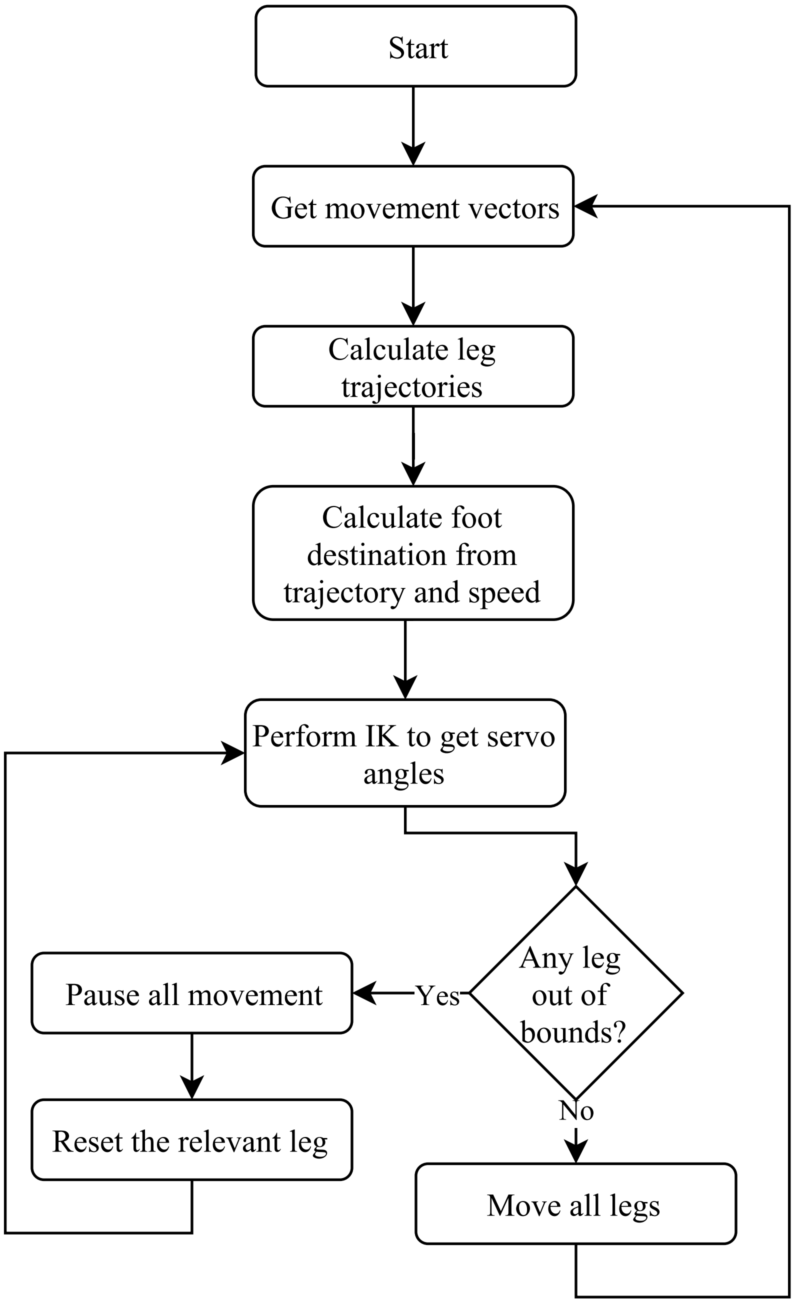


Figure 24. Flow diagram showing robot walking algorithm.

While this is the core of the algorithm, some additional functions need to be included to make provision for some conditions. These are:

- Power is turned on initially.
- Power is lost while in operation.
- Power is regained.
- Bluetooth is initially connected.
- Bluetooth is suddenly disconnected.

A more complete diagram illustrating the design implemented on the microcontroller can be seen in Figure 26. The colour of the block indicates the colour that the RGB LED on the robot would be in that state.

Servos are controlled with a square pulse with a specific width. The standard for hobbyist servo motors with a 180^o movement range is that a high pulse of $1ms$ equates to 0^o while $2ms$ results in 180^o. The period of this signal is $20ms$.

A typical set of control signals for an analogue servo motor can be seen in figure 25.

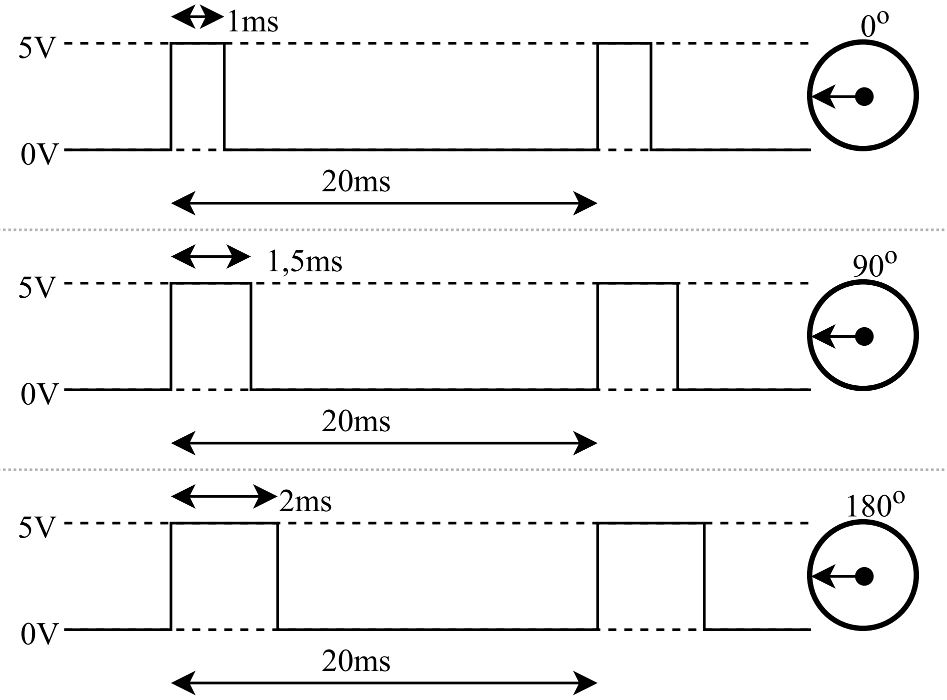


Figure 25. Illustration of the working of a traditional servo control signal.

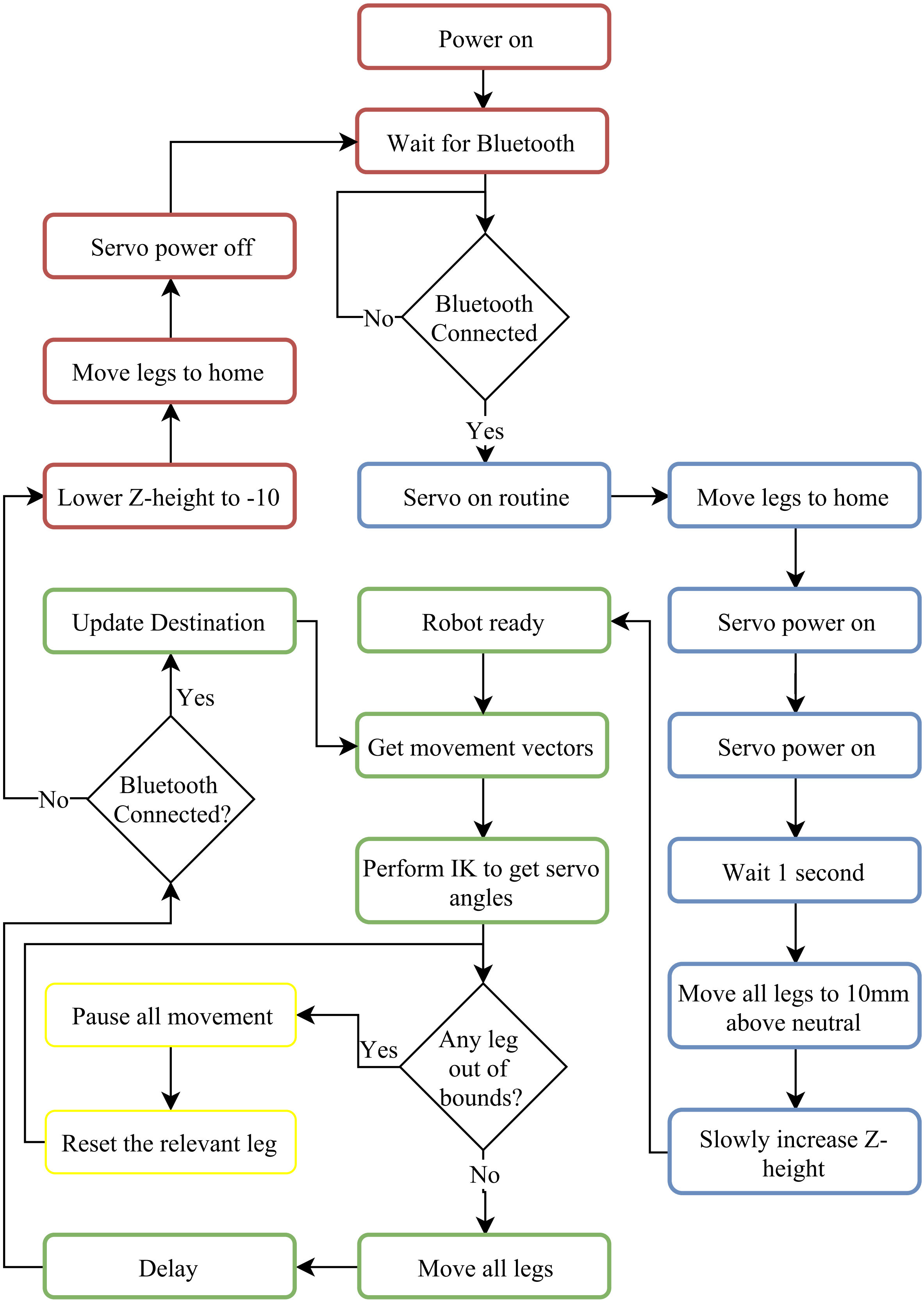


Figure 26. Flow diagram showing robot walking algorithm final implementation.

In order to control the 15 servo motors on the robot simultaneously, 15 independent control signals like the ones shown in Figure 25 should be generated. A common way to do this is to use the microcontroller's built-in pulse width modulation (PWM) module with a fixed frequency and variable duty cycle. While this is a valid approach for most cases, most microcontrollers don't have nearly enough independent PWM channels available to facilitate the 15 servo motors in this application.

Off-the-shelf modules that have a large number of PWM channels available and communicate with the microcontroller using the I2C, ISP or serial protocols are available but increases the electronic hardware components and power consumption.

The preferred solution in this case is using 15 normal digital output pins and two internal timers. One of the timers is set up to interrupt at $50Hz$, the period of the control signals. The second timer is reconfigured after each interrupt to have a new period. Figure 27 attempts to illustrate how this procedure works for the simpler case of 5 servos.

Servo	Angle (deg)	Timer (ms)
1	45	1.25
2	90	1.5
3	180	2
4	0	1
5	90	1.5

Table 5. Servo angles for the example explaining servo control using two timers.

Before starting the procedure, Table 5 is sorted in ascending order by the period in ms. The result can be seen in Table 6

Index	Servo	Angle (deg)	Timer (ms)
1	4	0	1
2	1	45	1.25
3	2	90	1.5
4	5	90	1.5
5	3	180	2

Table 6. Sorted servo angles for the example explaining servo control using two timers.

For the simplified case where 5 servo motors should be controlled, Table 5 shows example values to illustrate the procedure. The procedure is outlined using the event letters shown in Figure 27.

A Timer 1 interrupts. All servo pins are set high. Timer 2 period is set to the timer value for index 1 (a=1.25ms).

B Timer 2 interrupts. Servo for index 1 is set low (Servo 4). Timer 2 period is set to the timer value in index 2 - timer of index 1 (b=$1.25-1=0.25ms$).

C Timer 2 interrupts. Servo for index 2 is set low (Servo 1). Timer 2 period is set to the timer value in index 3 - timer of index 2 (c=$1.5-1.25=0.25ms$).

D Timer 2 interrupts. Servo for index 3 and 4 is set low (Servo 2 and 5). Timer 2 period is set to the timer value in index 5 - timer of index 4 (d=$2-1.5=0.5ms$).

E Timer 2 interrupts. Servo for index 5 is set low (Servo 3). Timer 2 is turned off for the remainder of the 20ms ($e=NULL$)

F Timer 1 interrupts. All servo pins are set high. Timer 2 period is set to the timer value for index 1 (a=1.25ms).

G Timer 2 interrupts. Servo for index 1 is set low (Servo 4). Timer 2 period is set to the timer value in index 2 - timer of index 1 (b=$1.25-1=0.25ms$).

H Timer 2 interrupts. Servo for index 2 is set low (Servo 1). Timer 2 period is set to the timer value in index 3 - timer of index 2 (c=$1.5-1.25=0.25ms$).

I Timer 2 interrupts. Servo for index 3 and 4 is set low (Servo 2 and 5). Timer 2 period is set to the timer value in index 5 - timer of index 4 (d=$2-1.5=0.5ms$).

J Timer 2 interrupts. Servo for index 5 is set low (Servo 3). Timer 2 is turned off for the remainder of the 20ms ($e=NULL$)

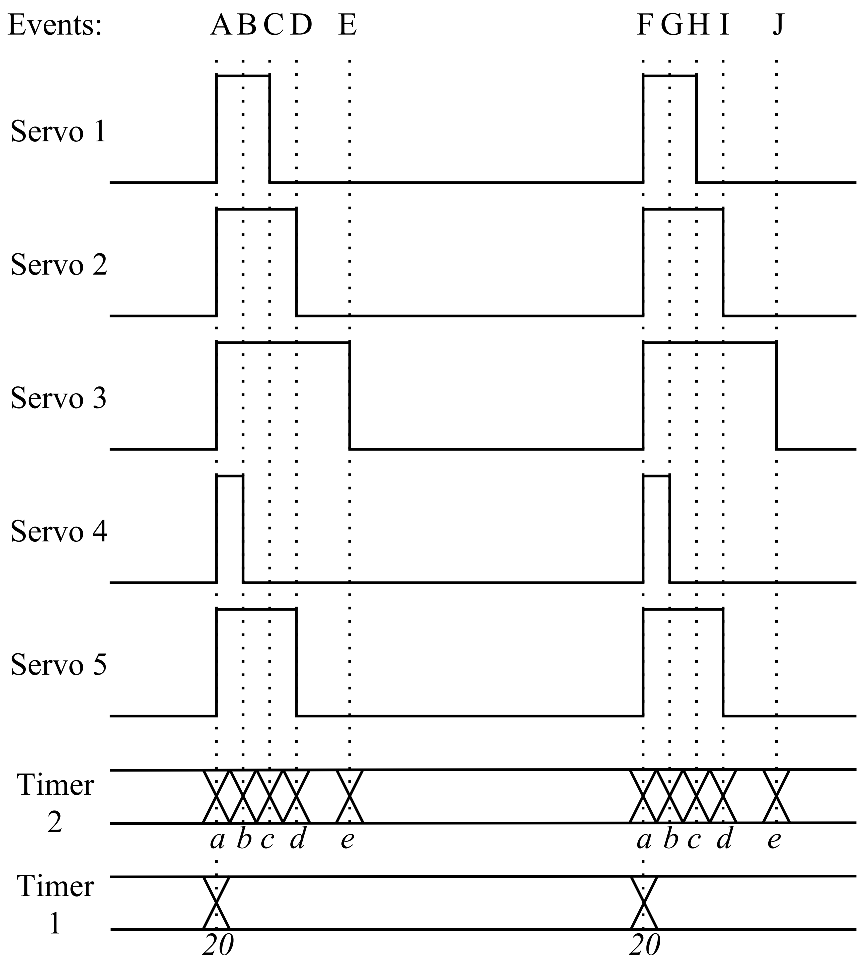


Figure 27. Illustration of how multiple servo motors are controlled using two timers.

3.8 Software implementation

For the implementation of the program on the microcontroller, the C programming language together with the KEIL microcontroller development kit, which is an integrated development environment (IDE), was used. The STM32 series microcontrollers come with an application called CubeMX that can be used to generate initialization code for a specific microcontroller. This was used to configure all of the peripherals used as well as the system clock settings.

The algorithms developed and refined in Python was then translated to C and implemented in the software. One important addition to the software used in Python is for the closed loop control system. In Python the legs would automatically be where they are instructed to be since this is a simulation. In the final implementation, sensors are added to the robot feet to sense that the leg makes contact with the ground. Some of the other functions implemented that were not included in the Python program are outlined below.

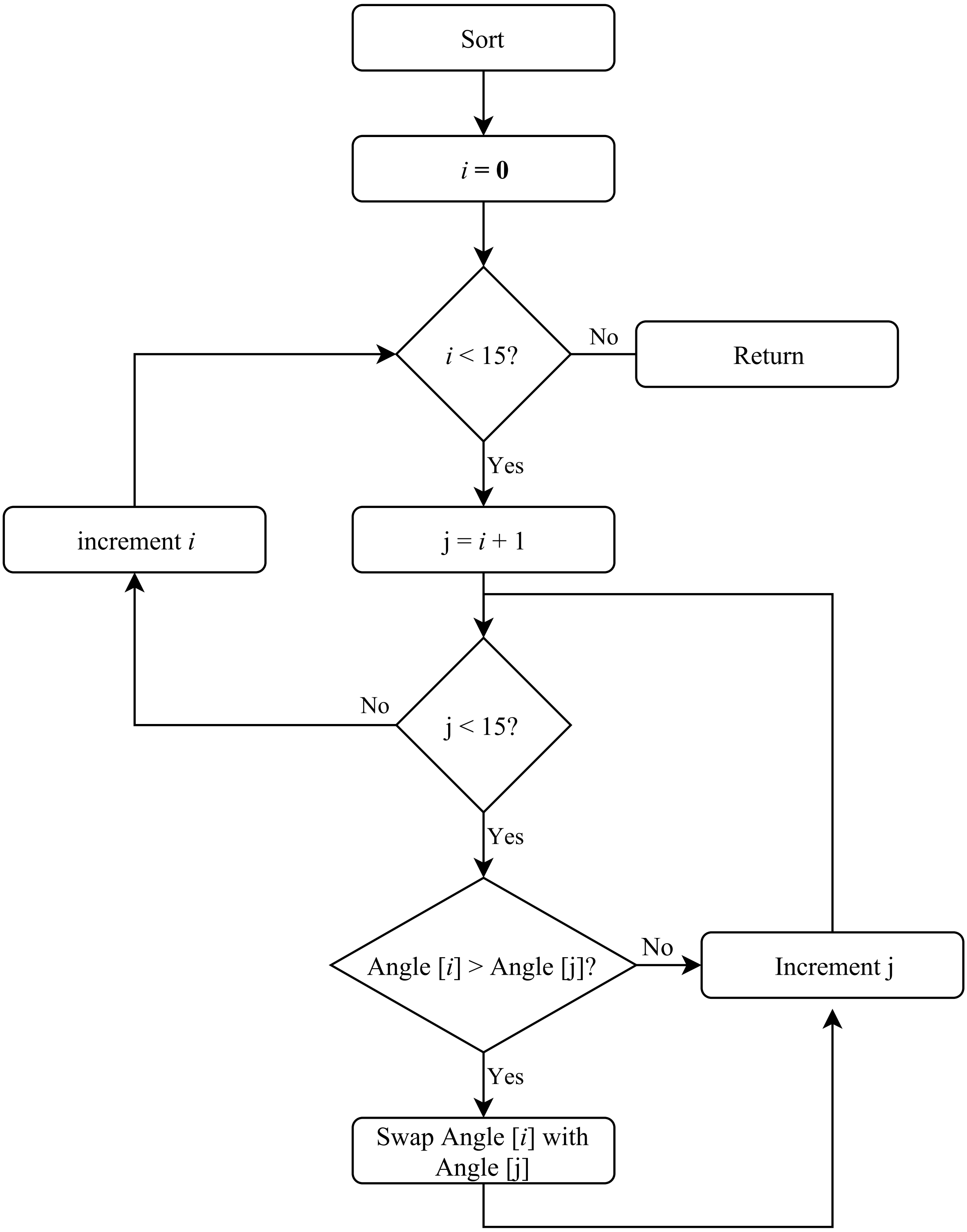


Figure 28. Flow diagram showing the functioning of the sorting required by the servo algorithm.

Figure 28 shows the functioning of the algorithm that sorts the servo times in ascending order. The sorting method used here is one of the simplest sorting methods, often referred to as bubble sorting. Since it is only 15 values being compared, it is not worth implementing a more complicated algorithm for the sake of efficiency

The timer interrupts are difficult to show in the main program flow because it is something that is constantly happening in the background. Figures 29 through 31 show the different timer interrupt service routines to show how this function works. It is important to not that timers 6 and 7 in Figure 29 correspond to timers 1 and 2 respectively in Figure 27

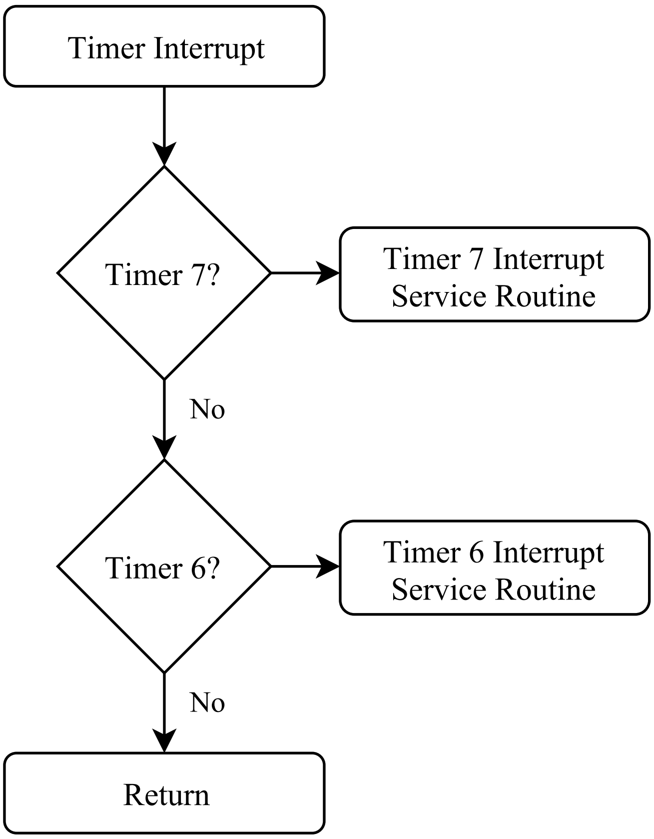


Figure 29. Flow diagram showing the functioning of the main timer interrupt service routine.

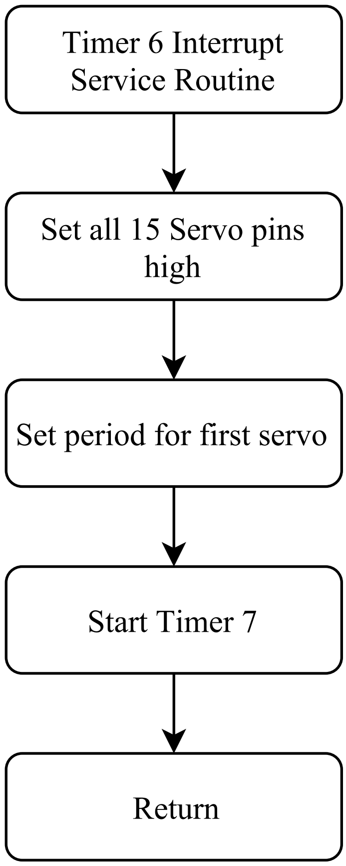


Figure 30. Flow diagram showing the functioning of the timer 6 interrupt service routine.

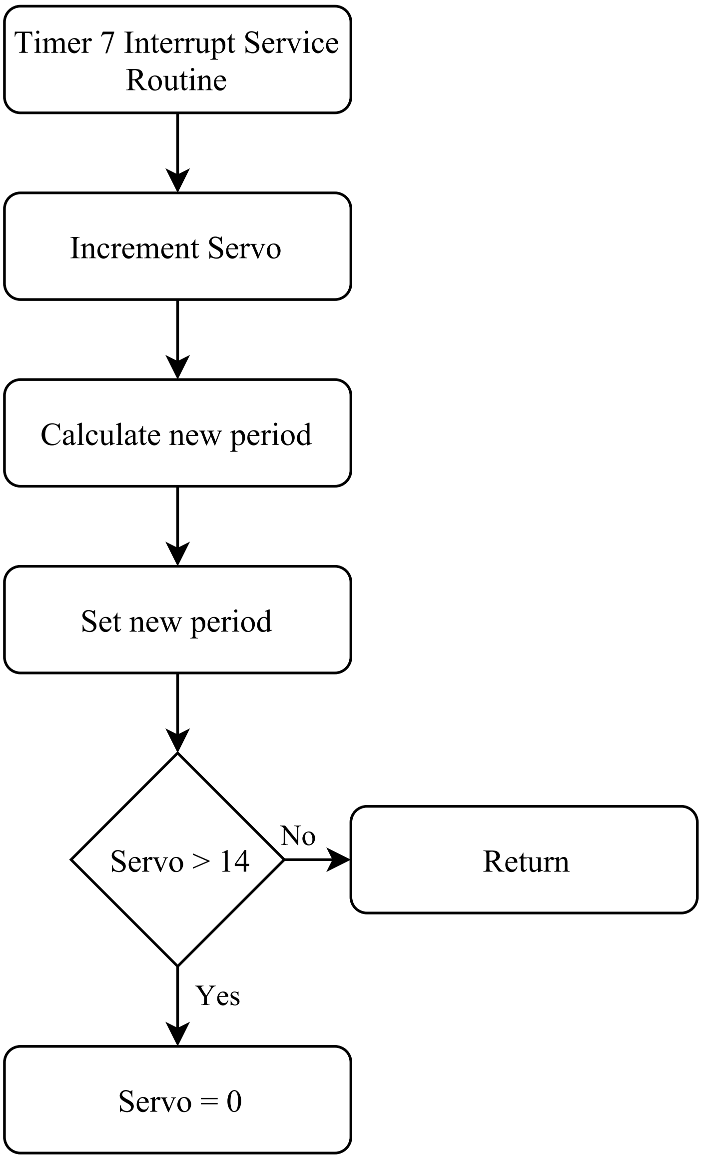


Figure 31. Flow diagram showing the functioning of the timer 7 interrupt service routine.

The communication with the Bluetooth module makes use of the serial protocol. Once a message is received as a string. It should be checked for validity and then decoded to get usable integer values from this. The android application always sends data in the same format. Each of the values for X, Y and R is represented by a single integer digit. This value can't be negative since the minus sign would take up another character and everything would be out of alignment. The solution is to represent each number as a value between -4 and 4, then add 5 to this number. The result is that a value of -4 would be represented by a 1 and a value of 4 represented by a 9. They are then inserted in a mask of the following format: *X_Y_R_** where the underscore is replaced by the appropriate character from 1 to 9 for each of the three values. If all of the values for X,Y and R are 0 from the UI, the resultant string would therefore be *X5Y5R5**. The decoding process is shown in Figure 32.

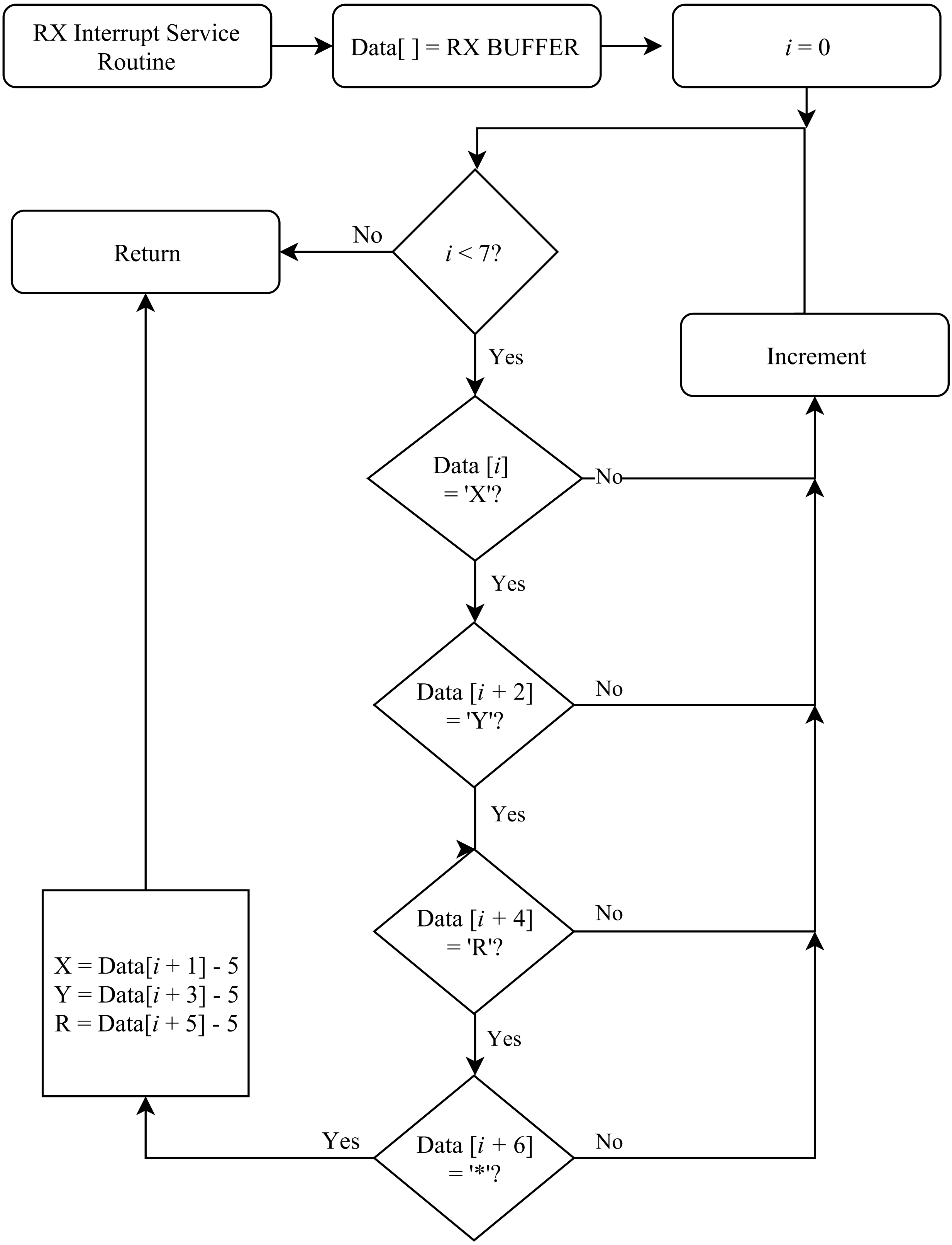


Figure 32. Flow diagram showing the decoding of data from the Bluetooth module.

3.9 Hardware design

This section covers the design of all of the mechanical hardware that was designed specifically for this project.

The first of the parts to be designed were the legs, this would determine the scale of the robot. The legs were designed in three different segments as discussed in the theoretical and mathematical analysis. In order to simplify the mechanical design and distribute the weight of the robot more evenly, the servos are mounted inside the legs. Because the legs are lifted by the servos, these should be as close as possible to the hip in order to minimize the leverage on the servo lifting the leg.

The annotations in Figure 33 that shows segment A is explained below:

1. Gear that connects to the servo motor located in leg B.
2. Hole for wire management.
3. Gear teeth that connect to the servo motor located in the base.
4. Centreline where a stainless steel shaft is inserted into bearings for horizontal movement.
5. Centreline where a stainless steel shaft is inserted into bearings for vertical movement.

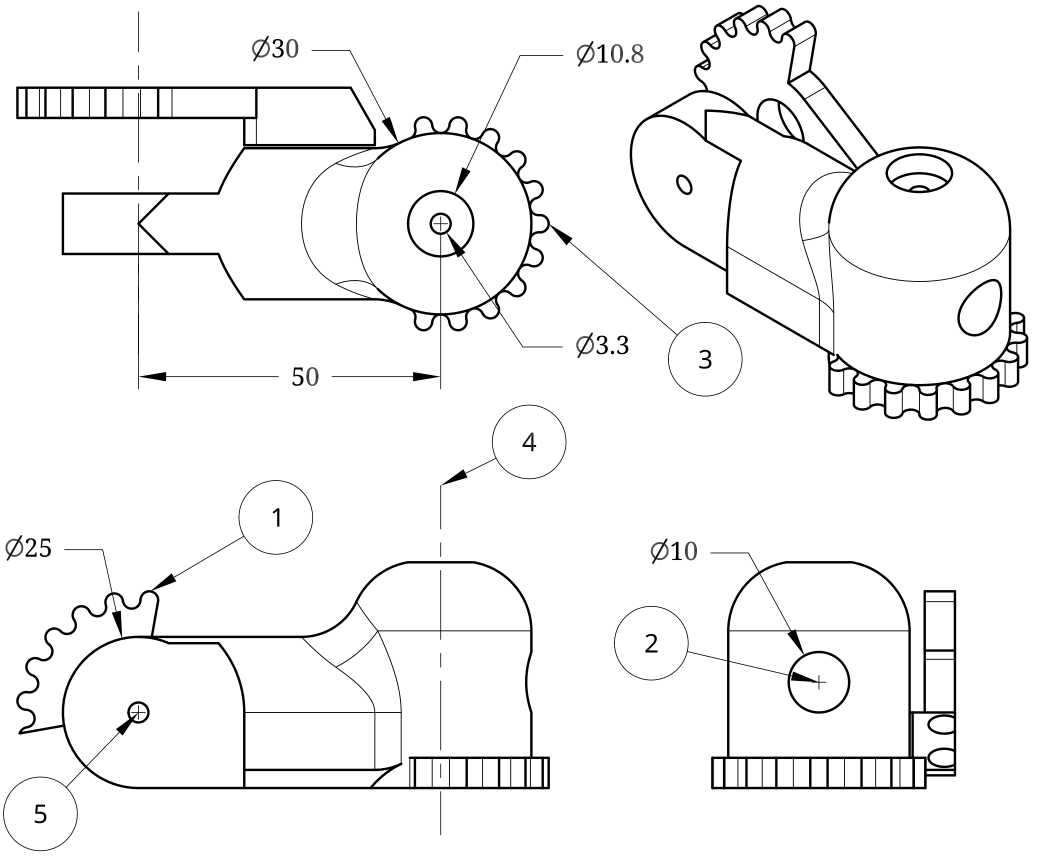


Figure 33. Illustration of how multiple servo motors are controlled using two timers.

The annotations in Figure 34 that shows segment B is explained below:

1. Cavity for the servo motor that drives segment C.
2. Cavity for the servo motor that drives segment B.
3. Ball with slot that connects to segment A.
4. Peg that goes in the ball with slot in segment C.
5. Centreline where a stainless steel shaft is inserted into bearings for vertical movement with segment C.
6. Centreline where a stainless steel shaft is inserted into bearings for vertical movement with segment A.

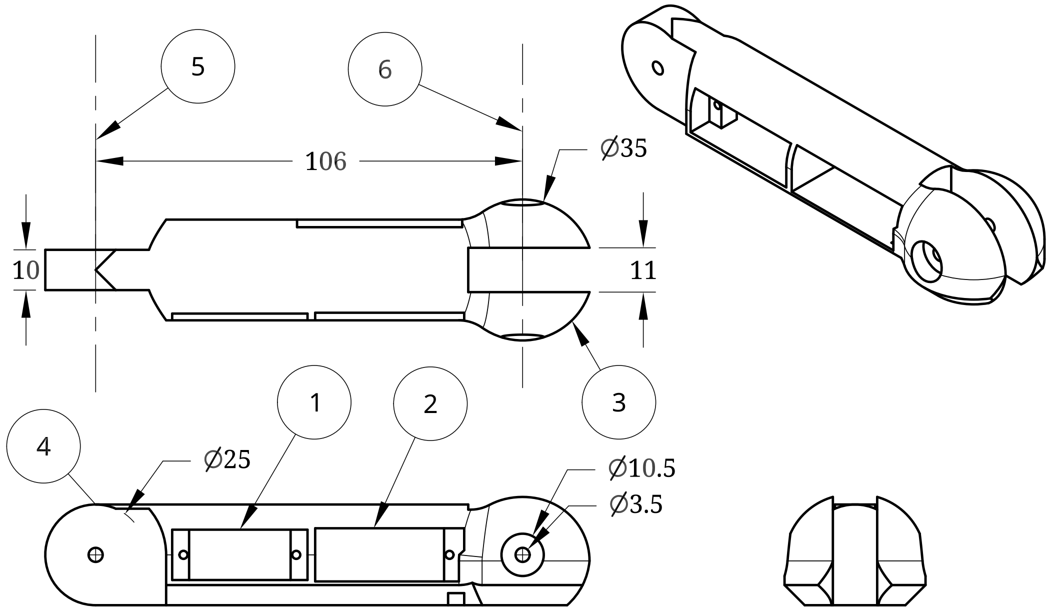


Figure 34. Illustration of how multiple servo motors are controlled using two timers.

The annotations in Figure 35 that shows segment C is explained below:

1. Centreline where a stainless steel shaft is inserted into bearings for vertical movement with segment B.
2. Flat part of the foot that makes contact with the ground.
3. Cut-out that allows for a wider range of movement.

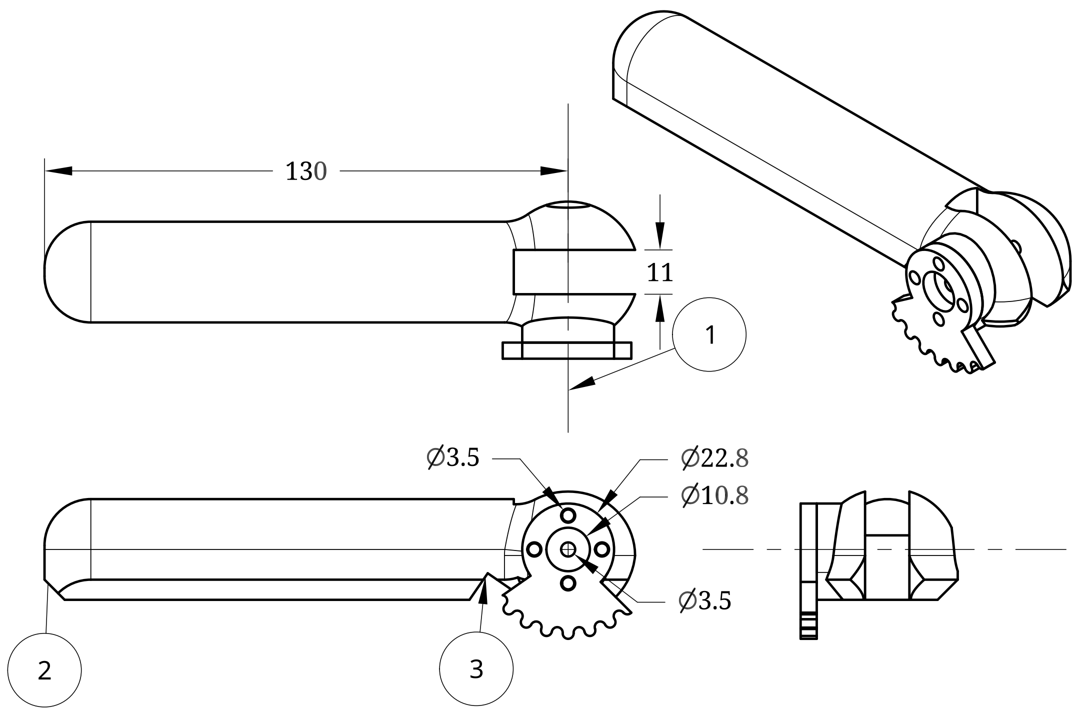


Figure 35. Illustration of how multiple servo motors are controlled using two timers.

The annotations in Figure 36 that shows the base is explained below:

1. Gear that connects to the servo motor located in leg B.
2. Hole for wire management.
3. Gear teeth that connect to the servo motor located in the base.
4. Centreline where a stainless steel shaft is inserted into bearings for horizontal movement.
5. Centreline where a stainless steel shaft is inserted into bearings for vertical movement.

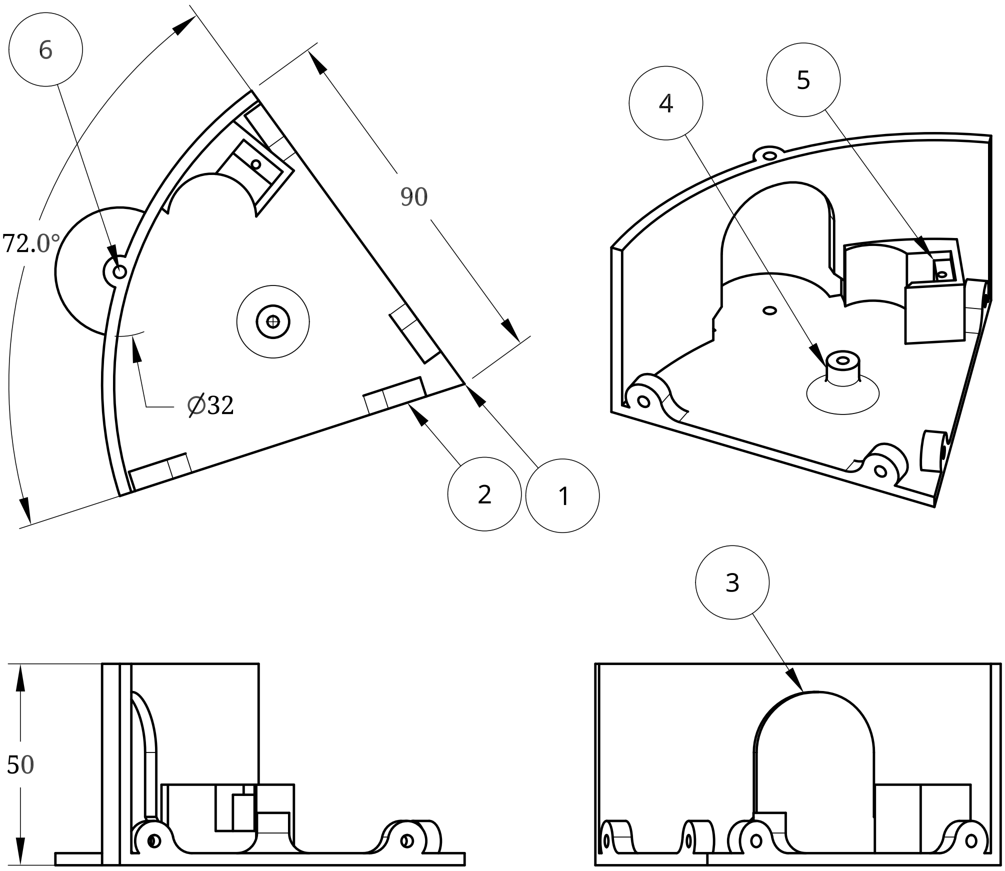


Figure 36. Illustration of how multiple servo motors are controlled using two timers.

3.10 Hardware implementation

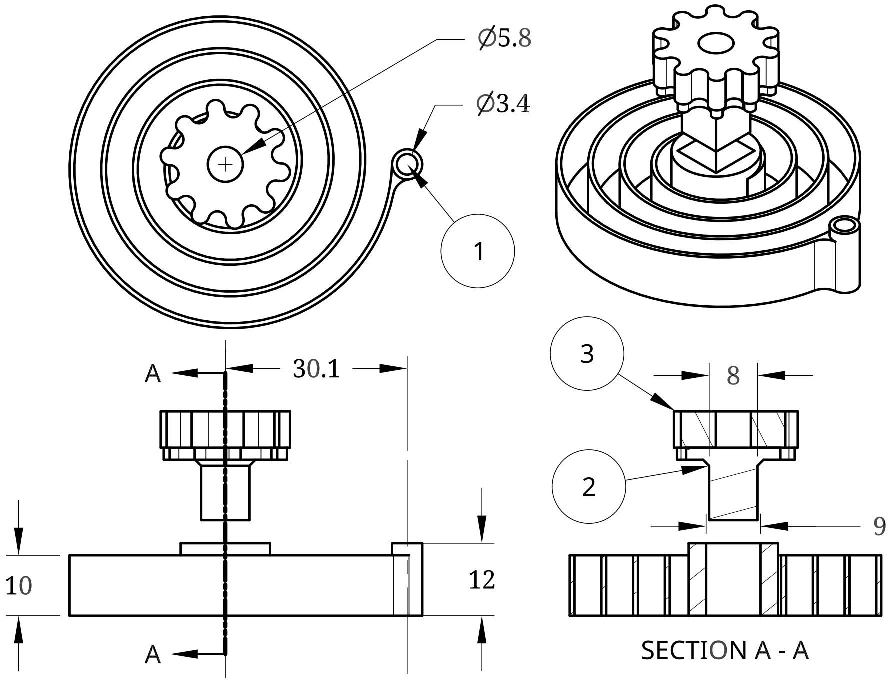


Figure 37. Illustration of how multiple servo motors are controlled using two timers.

3.11 Design summary

Task	Implementation	Task complete
Development of an IK engine in software	The mathematical analysis of the robot was used to create a generic model, which was then first implemented in Python and thereafter in the final robot implementation.	Completed
Development of a walking algorithm	A walking algorithm was developed and optimized in Python and then implemented in the final robot.	Completed
Development of a user interface	The user interface was developed for an Android smartphone using Android Studio.	Completed
Communication channel between user interface and robot	Bluetooth was used to establish a low bandwidth, low cost communication system between the smartphone and the robot. It communicates commands successfully.	Completed
Implementation of electronics on a PCB.	All electronics were implemented on a Veroboard instead of a PCB which is inexpensive and sufficiently robust for the purpose of this project. A PCB would not have made any functional difference to the end result and is not suited for making changes while prototyping. The only difference would have been aesthetically.	Incomplete
Implementation of inclinometer sensor	An inclinometer was planned to give the robot the ability to adapt to the exact angle of slanted surfaces but this was not implemented because the robot walks without a problem on the angles of surfaces specified for this project.	Incomplete

Table 7. Design summary

4. Results

4.1 Summary of results achieved

Description of requirement or specification (intended outcome)	**Actual outcome**	**Location in report**
Mission requirements of the product		
The robot should be able to move in a given direction in 30 degree increments from a stationary position.	The robot is able to move holonomically, therefore able to move in 30 degree increments.	
The robot should be able to rotate 90 degrees without translating the centre of the body by more than 5% of the body diameter.		
The time it requires to complete a manoeuvre should not vary more than 25% between smooth and rough surfaces.		
The robot should be able to walk at a speed of at least 100mm/s.		
The robot should be able to walk on a surface with a 10% incline without falling over.	The robot can successfully walk on a slanted surface with an incline of up to 10%	
Field conditions		
The robot should be in range of the wireless smartphone controller.	The robot works fine if it is within a range of 10m of the smartphone	
To prevent falling over, the robot should walk on surfaces close to horizontal.	The robot is able to work on slanted surfaces with an incline of less than 10%	
The robot should stay dry to protect electronics.	The robot was never tested in wet conditions	
The robot should work in normal temperature conditions for South Africa.	The robot never experienced any issues during the testing phase that relates to ambient temperature, relative humidity or any other atmospheric conditions	
Specifications		
The smartphone application should communicate commands from the user interface to the robot at a frequency of at least 10 Hz.		

4.2 Qualification tests

4.2.1 Qualification test 1 : Holonomic translation

4.2.1a Qualification test

Objectives of test/experiment
The objective is to prove that the robot is capable of moving in any direction from a standing position without first rotating. *Equipment used*
A canvas with lines in 30 degree increments is used to put the robot on. *Experimental parameters and setup*
The robot is turned on and placed in the centre of the canvas. The test is conducted in the field conditions listed in Table 8. *Experimental protocol*
The robot is instructed to move along one of the lines and then return to the centre by using the smartphone as remote control. Once back in the centre, the next line is followed. This process continues until all the lines have been followed.

4.2.1b Results and observations

Measurements
Figure 38 shows the robot on the canvas where the qualification test is being performed.

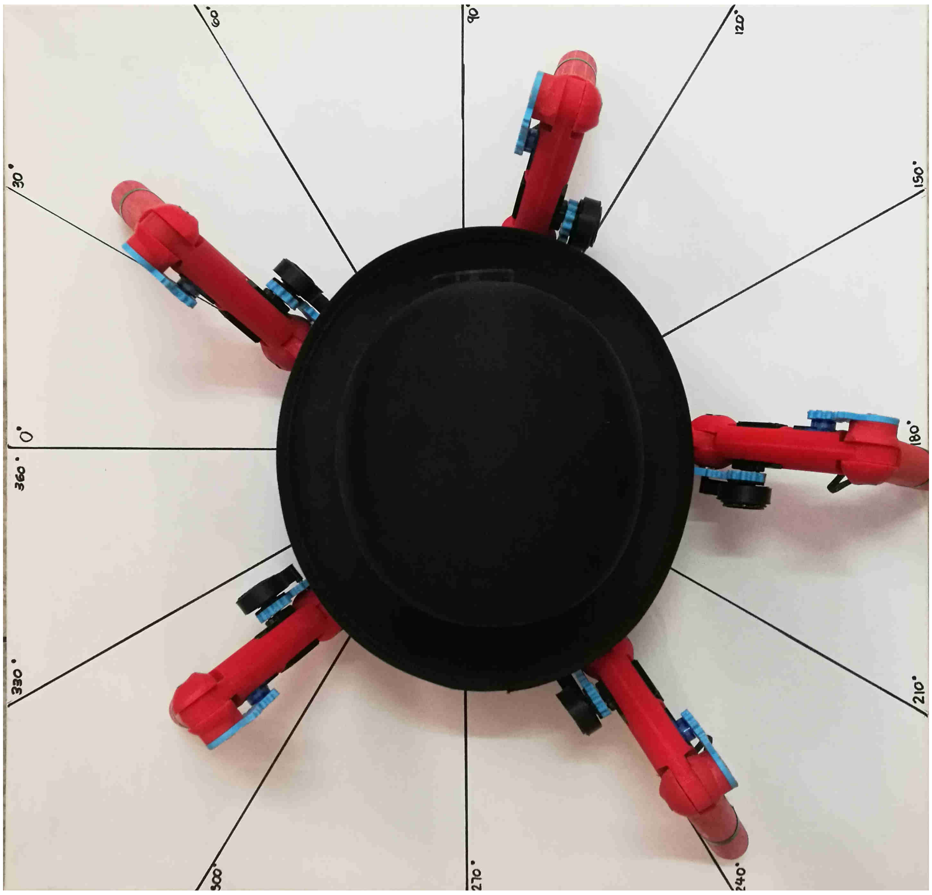


Figure 38. Top view of the robot on the canvas.

Description of results
The robot is able to successfully follow the lines on the canvas without ever rotating.

4.2.2 Qualification test 2 : Rotation without translation

4.2.2a Qualification test

Objectives of test/experiment
The objective of this test is to confirm that the robot is capable of rotating around its own axis without translating. *Equipment used*
The same canvas used in 4.2.1 is used for this qualification test. It has two circles, one has the diameter of the robot base, the second is 5% larger.

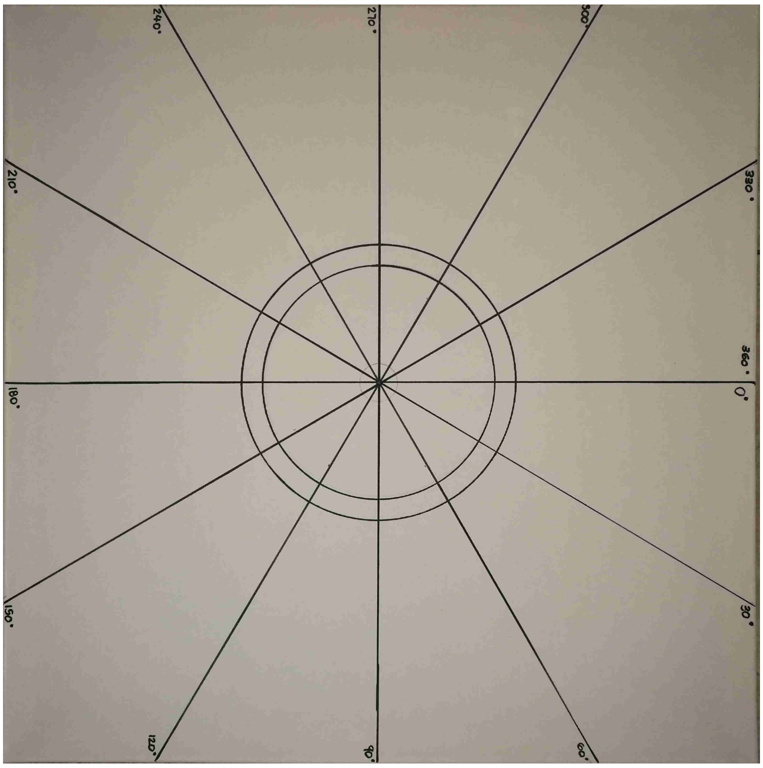


Figure 39. Canvas used for this experiment.

Experimental parameters and setup
The robot is placed inside the smaller circle and switched on. The test is conducted in the field conditions listed in Table 8. *Experimental protocol*
The robot is instructed to rotate from the smartphone application. After 90 degrees of rotation, the robot is instructed to stop and is then turned off. The robot should be within the bigger circle.

4.2.2b Results and observations

Measurements

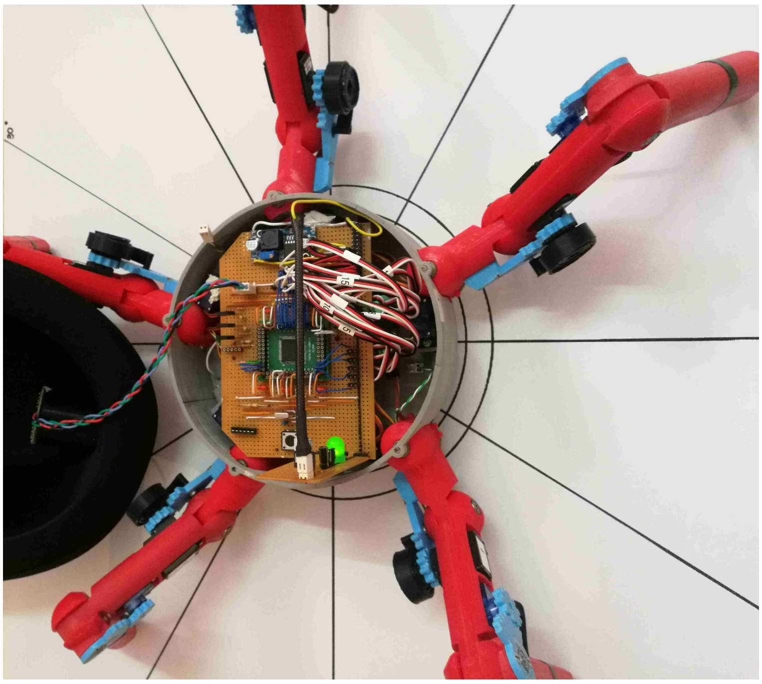


Figure 40. Robot inside the larger circle

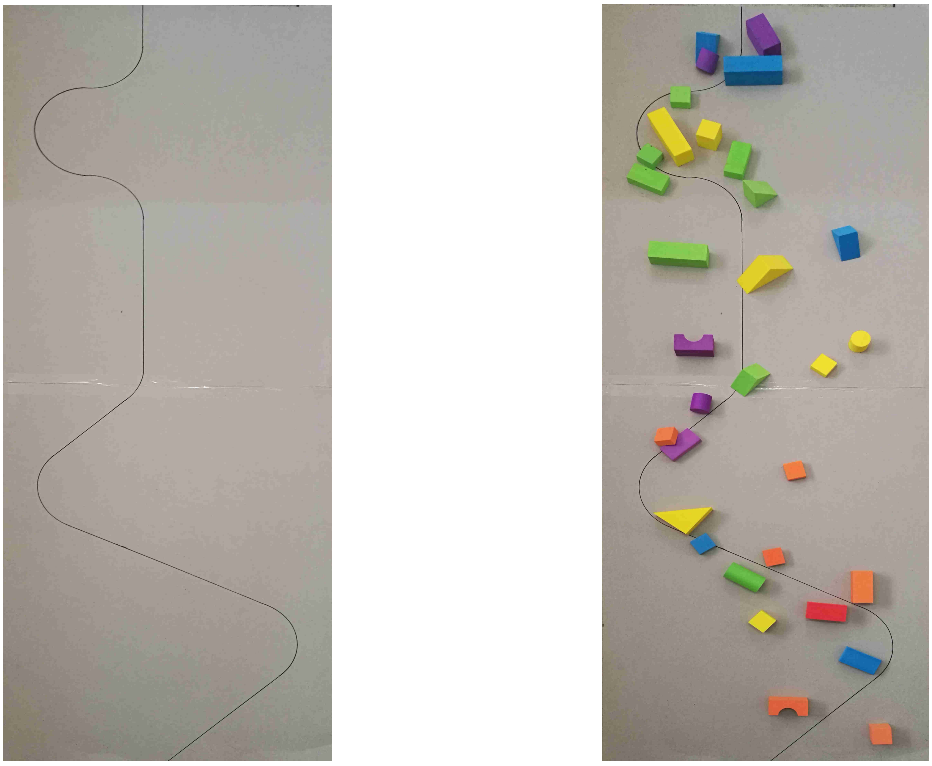
Description of results
The robot is still inside the larger circle after rotating 90 degrees.

4.2.3 Qualification test 3 :Speed on rough terrain

4.2.3a Qualification test

Objectives of test/experiment
The purpose of this test is to show that the robot functions reasonably well on rough terrain when compared to a smooth surface. *Equipment used*
A small test track developed for this purpose is used. In order to compare performance between the smooth and rough terrain, the time to complete the track is recorded with a stopwatch application on a smartphone. *Experimental parameters and setup*
A rough terrain is simulated by scattering small, loose obstacles (foam blocks) over the test track. This is a realistic representation of a wide variety of terrain conditions. For preparation the robot is placed on the start line of the smooth test track and switched on. The test is conducted in the field conditions listed in Table 8. *Experimental protocol*
The stopwatch is started and the robot is instructed to follow the line that forms the test track.

When the robot reaches the end, the time on the stopwatch is recorded. The test track is scattered with the small obstacles and the procedure is repeated.



(a) Test track without obstacles. **(b) Test track with obstacles.**

Figure 41. Test setup for proving that the robot can easily walk on rough surfaces

Figure 41 shows the test track for the two different tests.

4.2.3b Results and observations

Measurements

Course condition	**Time**
Smooth	5:12
Rough	6:20

Table 9. Results achieved for the test track

Description of results
The robot moves slower over the obstacles than on the smooth course.

4.2.4 Qualification test 4 : Speed

4.2.4a Qualification test

Objectives of test/experiment
The objective of this test is to prove that the robot can reach a speed of 100mm/s *Equipment used*
A linear track designed for this experiment is used to measure the speed. It has markings that is a fixed distance apart that makes it easy to measure the distance travelled. The time will be recorded using a stopwatch application on a smartphone.

Figure 42. Robot on the track for measuring speed.

Figure 42 shows the robot on the track mentioned above. *Experimental parameters and setup*
The robot is placed at the start of the track and switched on. The test is conducted in the field

conditions listed in Table 8. *Experimental protocol*
The stopwatch is started and the robot is instructed to move down the track as fast as possible. When the end of the track is reached, the time on the stopwatch is recorded.

4.2.4b Results and observations

Measurements

Distance	**Time**	**Speed**
1.5m	3:07	8mm/s

Table 10. Results achieved for the speed test track

Description of results
The robot moves much slower than anticipated.

4.2.5 Qualification test 5 :Slopes

4.2.5a Qualification test

Objectives of test/experiment
The objective is to prove that the robot is capable of walking on a slanted surface without falling over. *Equipment used*
A plank is used to create the slanted surface. A long ruler is used to measure the incline of the plank. *Experimental parameters and setup*
The plank is angled at 10%. This is ensured by making sure that the plank has a height of 10cm at the point that is 1m from the edge of the plank on the ground. The robot is put at the base of the plank. The test is conducted in the field conditions listed in Table 8. *Experimental protocol*
The robot is instructed to walk on the plank to see if it struggles or falls over at any stage in the process.

4.2.5b Results and observations

Description of results
The robot walks without any additional difficulty due to the incline. The robot was never close to falling over at any stage in the experiment.

4.2.6 Qualification test 6 :Slippery surfaces

4.2.6a Qualification test

Objectives of test/experiment
The objective is investigate the performance of the robot on a slippery surface *Equipment used*
A wooden floor is used as the slippery surface for the robot to walk on. *Experimental parameters and setup*
The robot is put on the wooden floor and turned on. The test is conducted in the field conditions listed in Table 8. *Experimental protocol*
The robot is instructed to walk on the wooden floor in both slow and rapid movements to investigate how the legged design functions on low traction environments.

4.2.6b Results and observations

Description of results
The robot with more difficulty on slippery surfaces than on surfaces withh higher traction.

5. Discussion

5.1 Interpretation of results

The results recorded in the previous section of this report indicate that the robot is able to function as a holonomic robot, therefore it is able to move in any direction without first rotating and rotate without translating. The results that measure the performance of the robot, however, show that it does not do it nearly as good as it was designed to do it. The maximum speed of the robot, for example, is far below what was aimed for originally. The fact that the robot is able to correctly execute commands shows that the theoretical design and algorithm is valid. In the case of the speed requirement, it is the mechanical design that was lacking. The robot was far heavier that planned for and the servo motors used struggle to carry the load. The only way to make the robot function is to slow all of the movements down. The test on slippery surfaces had unexpected results as well. The expectation was that the robot might struggle to walk because the legs would slip on the floor and it would slip when propelling itself forward. Instead the low friction causes the legs to want to spread outwards, causing more strain on the servo motors. The choice of servo motor implemented in the robot caused the robot to not work as good as it should have. If more powerful servo motors were implemented, the power consumption would have been significantly higher but torque would have been sufficient.

5.2 Aspects to be improved

The mechanical drive system for the legs could be improved. Torque at the critical joint is not sufficient and the full range of motion is not used. A solution that would not require a complete redesign would be to change the gear ratio at the critical joint to have a more powerful actuator with a smaller range of motion. The alternative solution would be to design the robot with larger and more powerful servo motors in the first place. This would, however, require a redesign of the power management system for the servo motors as well since larger servos require more current.

5.3 Strong points

The strongest part of this project is the inverse kinematics and motion planning algorithm. This is the case because it was designed for a parametric model which made it extremely flexible in the design phase. It was also thoroughly tested and optimized in the simulation UI in Python well before it was first implemented in the robot.

5.4 Under which circumstances will the current system fail?

The largest risk of failure of this robot is the strained servo motors. If any weight is added to the system, it becomes more difficult to move. This also places restrictions on the speed of the movements. The strained servo motors require a current closer to the stall current than the system was designed for. This means that the supply rail of the servo motors dip in voltage if the speed of motions is too high, causing the system to fail. The system would fail safe rather than violently. If a servo stalls, the batteries are protected internally against too high discharge current as well as over discharge and would simply shut down in either of these cases.

5.5 Design ergonomics

The user interface is designed to be as simple to use as possible and therefore only has the two joystick-like controls on the screen. Both can be used at the same time. The advantage of joysticks over a directional pad is that the user does not have to look at the display when using it.

5.6 Health and safety aspects of the design

The Lithium-Ion batteries chosen for implementation in this project have a built-in protection against over-charging, over-discharging, over-current and over-temperature. This makes the use of Lithium-Ion batteries much safer.

5.7 Social and legal impact and benefits of the design

There is no specific legislation that the product has to comply with. If the robot is able to we controlled remotely using longer range technologies, the methods used in this project could be used to get help or supplies to people in need in dangerous situations. This could be getting food or supplies to people in war zones or navigating tough terrain during search-and-rescue operations.

5.8 Environmental impact and benefits of the design

The product is designed to have as little impact as possible on the environment during use and when the end of its lifetime is reached. All of the plastic used for creating the mechanical parts are created using PLA plastic, which is bio-degradable. The Lithium-Ion batteries can be handed in for recycling at any drop-off point for used batteries found in most shopping malls. The stainless steel parts such as bearings and rods can be re-used in another project. The only part that poses a risk is the electronic waste. This includes the Veroboard, PCBs of modules and the electronic components. These should be dropped off somewhere where the recycling is specialized for the handling of electronic waste.

6. Conclusion

6.1 Summary of the work

The mathematical analysis of the generic robot body was done in such a way that the model could be changed parametrically to allow for greater flexibility in the design process. This was applied and developed in a Python simulation which eventually ended up to include a user interface and a plot of the robot body in action. Once the walking algorithm was implemented and optimized, the work was implemented in C to be used in the microcontroller that would control the robot. Algorithms were designed and implemented to control all 15 of the servo motors with as little hardware and software resources required as possible. An android application was developed to be a user interface and act as a remote control for the robot. Instructions are sent to the robot using Bluetooth, which are received by a Bluetooth module connected to the microcontroller with a serial connection. Legs and a body was 3D-printed to fit the servo motors and the robot was able to move its first legs. The movement of legged vehicles on slippery surfaces was investigated and the results were surprising. The investigation was somewhat limited by the available torque from the servo motors.

6.2 Summary of the observations and findings

The robot is able to walk but not in the way originally intended. Although holonomic motion is possible and those design goals have been met, the robot struggles to move properly. This is a result of the robot being much heavier than anticipated. The extra strain on the servo motors causes a high power demand from the servo motors which the DC-DC converter is not always able to supply. The result in the rail voltage is that dips occur when the robot requires strength for more than one leg at a time. The dip in voltage reduces the servo torque, making the situation even worse. A partial solution to this is to keep the robot as light as possible and severely limit the speed of all movements. This somewhat helps but the robot is no longer able to meet the mission critical specification for speed. The movement accuracy also suffers from the same problem.

6.3 Suggestions for future work

The work could be continued and largely improved by getting the robot to walk properly without exchanging the servo motors with stronger ones. This could possibly be done by changing the gear ratio on the outer two limbs to sacrifice some range for torque. This will allow the design to keep the low power consumption it was aiming for while being able to work as intended in the design.

7. References

[1] A. Hidayat, A. N. Jati and R. E. Saputra, “Autonomous quadruped robot locomotion control using inverse kinematics and sine pattern method”, *IEEE*, 2017.

[2] J. Oh, H. Bae and J.-H. Oh, “Analytic inverse kinematics considering the joint constraints and self-collision for redundant 7dof manipulator”, *IEEE*, 2017.

[3] P. Jain, “Odometry and motion planning for omni drive robots”, *IEEE*, 2014.

[4] J. Ollervides, J. Orrante-Sakanassi, V. Santibanez and A. Dzul, “Navigation control system of walking hexapod robot”, *IEEE*, 2012.

[5] A. Zagler and F. Pfeiffer, “Moritz a pipe crawler for tube junctions”, *Robotics and Automation*, vol. 3, 2003.

[6] A. Kakogawa and S. Ma, “Design of a multilink-articulated wheeled inspection robot for winding pipelines: Airo-ii”, *Intelligent Robots and Systems*, vol. 1, 2016.

[7] R. C. Arkin and W. F. Gardner, “Reactive inclinometer-based mobile robot navigation”, *IEEE*, vol. 2, 1990.

[8] J. H. Park and O. Kwon, “Reflex control of biped robot locomotion on a slippery surface”, *IEEE*, 2001.

[9] S. Dhawan, “Analogy of promising wireless technologies on different frequencies: Bluetooth, wifi, and wimax”, *Wireless Broadband and Ultra Wideband Communications*, vol. 2, 2007.