

**UNIVERSITY OF BALAMAND**

**ISSAM FARES FACULTY OF TECHNOLOGY**

**Graduation Project Implementation (Tech391)**

**Cargo Hexapod**

**Submitted by :**

Louay El Masri (A1920068)

Mohamad Doreid Abdallah (A1910873)

**Guided by:**

Dr. Kamil Klaime

**Year**: 2021- 2022

***Acknowledgement***

We would like to thank everyone who helped us finish our project and contributed to its success. First of all, we convey our sincere gratitude for our supervisor Dr. Kamil Klaime. Without his proper guidance this project would’ve been a little success. We would also like to thank Dr. Rodrigue Imad, who was the first one to encourage us to implement this great idea for our final year project. Finally, we can’t forget Dr. Racha Berbari, Dr Ibrahim Shanbour, Dr. Georgina Ghazi and the rest of our instructors. After all they are the ones who helped us to gain all the experience needed to conduct this project and make it perfect.

***Abstract***

As senior Mechatronics Engineering students, we are required to fully design and implement a graduation project. We chose to make, as our project, a real life Hexapod Robot wirelessly controlled by a mobile app we fully developped. Basically, a hexapod is an 18 DOF[[1]](#footnote-1) six-legged robot. A hexapod has a lot of advantages against wheeled robots. However, the number one benefit that hexapod robots have ahead of wheeled robots is walking on rough terrains. For this solely reason, we chose to design a cargo hexapod that helps to carry luggage for military troops. Moreover, it is critical to incorporate the dynamic model in the design of the control algorithm and motion simulation to be able to control a robot manipulator as required by its operation.

The following report will go over all the little details of the design and implementation phases for this hexapod; from all the kinematic study needed, its locomentation and stability all over to the designing of the app and connecting it to the hexapod.

Table of Contents

[1. Chapter: Introduction 7](#_Toc104761564)

[1.1 Robotics 7](#_Toc104761565)

[1.2 Hexapod 8](#_Toc104761566)

[1.3 History 9](#_Toc104761567)

[2. Chapter: Background Theory 10](#_Toc104761568)

[2.1 Symbolic Representation of Robots 10](#_Toc104761569)

[2.2 Representing Positions 11](#_Toc104761570)

[2.3 Rotation Matrix 12](#_Toc104761571)

[2.4 Displacement Vector 14](#_Toc104761572)

[2.5 Roll, Pitch, Yaw Angles 15](#_Toc104761573)

[2.6 Homogeneous Transformations 16](#_Toc104761574)

[3. Chapter: Kinematics Modeling 17](#_Toc104761575)

[3.1 Kinematic Chains 17](#_Toc104761576)

[3.2 Forward Kinematics 18](#_Toc104761577)

[3.3 Inverse Kinematics 22](#_Toc104761578)

[4. Chapter: Gait Analysis 25](#_Toc104761579)

[4.1 Types of Gaits 25](#_Toc104761580)

[4.1.1 Tripod Gait 25](#_Toc104761581)

[4.1.2 Ripple Gait 26](#_Toc104761582)

[4.1.3 Wave Gait 26](#_Toc104761583)

[4.2 Hexapod Stride 27](#_Toc104761584)

[4.3 Stability Margin using Support Polygon 28](#_Toc104761585)

[5. Chapter: Implementation 29](#_Toc104761586)

[5.1 Components list 29](#_Toc104761587)

[5.1.1 The Brain 29](#_Toc104761588)

[5.1.2 Power Supply 30](#_Toc104761589)

[5.1.3 Servo driver 30](#_Toc104761590)

[5.1.4 Rest of the components 31](#_Toc104761591)

[5.2 Wiring 32](#_Toc104761592)

[5.2.1 VKLSVAN PCA9685 Driver/ raspberry pi: 33](#_Toc104761593)

[5.2.2 MG996R Servomotor/ VKLSVAN PCA9685 Driver: 34](#_Toc104761594)

[34](#_Toc104761595)

[5.2.3 Raspberry pi-camera-module-v2/ Raspberry pi: 35](#_Toc104761596)

[5.2.4 HC-SR04 Ultrasonic sensor/ Raspberry pi: 35](#_Toc104761597)

[5.2.5 Power supply Wiring: 37](#_Toc104761598)

[5.3 Programming 37](#_Toc104761599)

[5.4 Economical Study 38](#_Toc104761600)

[6. Chapter: Mobile Application 39](#_Toc104761601)

[6.1 Xamarin Forms 39](#_Toc104761602)

[6.2 MQTT 42](#_Toc104761603)

[6.2.1 Client 43](#_Toc104761604)

[6.2.2 Broker 43](#_Toc104761605)

[6.2.3 MQTT Connection 44](#_Toc104761606)

[7. Conclusion 45](#_Toc104761607)

[8. References 46](#_Toc104761608)

Table of Figures

[Figure ‎1‑1: Our Hexapod 8](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761425)

[Figure ‎2‑1: Symbolic representation of robots joints 10](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761426)

[Figure ‎2‑2: Two coordinate frame, a point P, two vectors 11](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761427)

[Figure ‎2‑3: Rotation about z0 by an angle 12](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761428)

[Figure ‎2‑4: Displacement vector example 14](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761429)

[Figure ‎2‑5: Roll, pitch and yaw angles 15](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761430)

[Figure ‎3‑1: Coordinate frames attached to an elbow manipulator 18](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761431)

[Figure ‎3‑2: Hexapod assembled, coordinate frames and displacement vector 19](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761432)

[Figure ‎3‑3: Hexapod leg parts 20](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761433)

[Figure ‎3‑4: Top and side view of a hexapod leg 23](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761434)

[Figure ‎3‑5: Clearer side view of robot leg 23](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761435)

[Figure ‎4‑1: Leg positions of the hexapod 25](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761436)

[Figure ‎4‑2: Footfall patterns of the different gaits 27](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761437)

[Figure ‎4‑3: Tripod gait support polygon 28](#_Toc104761438)

[Figure ‎5‑1: Raspberry Pi Model B 29](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761439)

[Figure ‎5‑2: Power Supply 5V 40A 30](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761440)

[Figure ‎5‑3: Servo driver VKLSVAN PCA9685 30](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761441)

[Figure ‎5‑4: MG996R servo motor 31](#_Toc104761442)

[Figure ‎5‑5: Raspberry-pi-camera-module-v2 31](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761443)

[Figure ‎5‑6: HC-SR04 Ultrasonic sensor 31](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761444)

[Figure ‎5‑7: Acrylic Parts 32](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761445)

[Figure ‎5‑8: Raspberry pi to servo driver wiring 33](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761446)

[Figure ‎5‑9: MG996R to the servo driver wiring 34](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761447)

[Figure ‎5‑10: Pi camera with raspberry pi 35](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761448)

[Figure ‎5‑11: HC-SR0 ultrasonic sensor with raspberry pi 35](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761449)

[Figure ‎5‑12 full wiring diagram 36](#_Toc104761450)

[Figure ‎5‑13: Power supply wiring 37](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761451)

[Figure ‎6‑1: Xamarin logo 39](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761452)

[Figure ‎6‑2: Second Page 41](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761453)

[Figure ‎6‑3: First Page 41](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761454)

[Figure ‎6‑4: Third Page 41](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761455)

[Figure ‎6‑5: Fifth Page 42](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761456)

[Figure ‎6‑6: Forth Page 42](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761457)

[Figure ‎6‑7: MQTT connection 44](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761458)

[Figure ‎6‑8: MQTT connection preview 44](file:///C:\Users\dorei\Downloads\Graduation-Project-Implementation%20%20(1).docx#_Toc104761459)

Table of Tables

[Table 1: DH parameters for one robot leg 21](#_Toc104744823)

[Table 2: Economical Study 37](#_Toc104744824)

# Chapter: Introduction

## Robotics

Robotics is a relatively new branch of modern technology that blurs the lines between engineering disciplines. The complexity of robots and their applications necessitates the study of mechanical engineering, electrical engineering, systems and industrial engineering, computer science, economics, and mathematics. There are two types of robotic systems now being researched: wheels robots and legs robots. In planar surfaces, the former performs more quick and gentle movements. They are, nevertheless, extremely sensitive to natural barriers and non-continuous touch surfaces. According to the United States Army, the wheel can only reach 50% of the world's population. Since they have more mobility and flexibility to irregular surfaces and barriers, such as climbing and descending stairs, the other one, often called as 'walking machines' are able to overcome the issues that the wheels' robots confront. It should be highlighted, however, that the process of modeling and regulating complex systems is far more challenging.

## **Hexapod**

Legged hexapod robots are six-legged mechanical walking machines. This category is fascinating for a variety of reasons. Indeed, they have greater mobility on uneven and irregular terrains, which is especially useful in human-hazardous areas like minefields, space or planets, nuclear power plants, or when the terrain must remain mostly undisturbed for research purposes. The hexapod robots have higher static stability while moving and standing. Hexapod robots must have certain critical properties, including stability, ground adaptation, and quick movement. However, modeling hexapod robots, on the other hand, is difficult due to the large number of legs with significant degrees of freedom. Designers must examine some of the most essential design concerns and constraints when designing legged robots, including the mechanical structure of the robot body, leg architecture, actuators and drive mechanisms, control architecture, power supply, and autonomy.

Figure ‑: Our Hexapod

## History

A number of legged mobile robot prototypes have been produced during the previous decade, ranging from biped to four and six legged robots. The early designs attempted to emulate animal movements, while later versions were more realistic and simple. Several published research works deal with the movement's control and modeling, but only a few studies present quantitative results of the system's motion characteristics, particularly when it comes to analyzing the mechanical efforts created throughout the movement.

The aforementioned factors justify the increased interest in locomotion robot research. The current work was done in this framework, with the goal of designing and building a hexapod robotic system. One of the qualities that makes studying legged locomotion systems both interesting and difficult is their mechanical complexity. In general, mobile robotic systems are mechanisms that can be investigated using classical mechanics (e.g., Newton-Euler technique, Lagrange-D'Alembert formulation, Denavit Hartenberg approach, and so on). A kinematic and dynamic analysis of mechanical systems is also designed to examine movement in terms of displacement, velocity, and acceleration, as well as the forces and torques created and transmitted. This allows for the identification of the more crucial scenarios and, as a result, the component design.

# Chapter: Background Theory

## 2.1 Symbolic Representation of Robots

The kinematic chain of robot manipulators is made up of links joined by joints. The most common types of joints are rotary (revolute) and linear (prismatic). Like a hinge, a revolute joint allows relative rotation between two links. Between two links, a prismatic joint allows for linear relative motion. They are represented by the symbols shown in Figure 2-1.

An RRR arm, for example, is a three-link arm with three revolute joints. Each link's interconnection is represented by a joint. The axis of rotation is denoted by

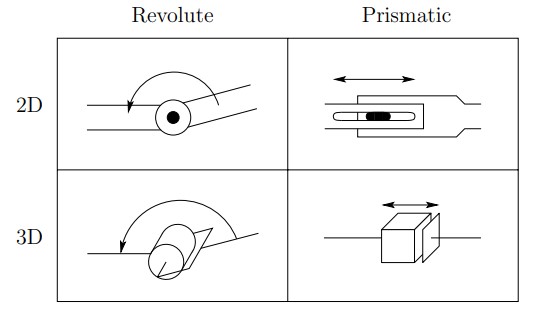


Figure ‑: Symbolic representation of robots joints

if the joint is the interconnection of links I and I + 1, the axis along which a prismatic joint translates by zi is called the revolute joint's rotation.

## 2.2 Representing Positions

All coordinate vectors must be defined with regard to the same coordinate frame in order to execute algebraic manipulations with coordinates.

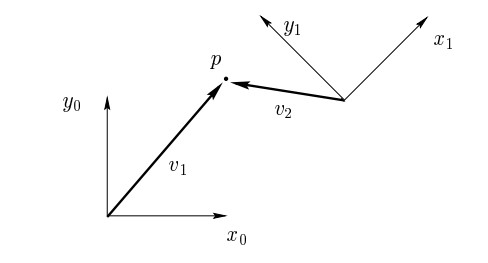


Figure ‑: Two coordinate frame, a point P, two vectors

Figure 2-2 shows two coordinate frames that differ in orientation. Because robot tasks are frequently stated using Cartesian coordinates, analytic reasoning is commonly used in robotics. Of course, a coordinate frame must be specified before coordinates may be assigned. Considering again Figure 2-2, we could specify the coordinates of the point P with respect to either frame o0x0y0 or frame o1x1y1.

## 2.3 Rotation Matrix

All vectors must be stated in the same coordinate frame in order to perform algebraic manipulations with vectors using coordinates. This is accomplished using rotation matrices.. In the n-dimensional Euclidean space, an n×n rotation matrix determines the orientation of one frame relative to another frame.

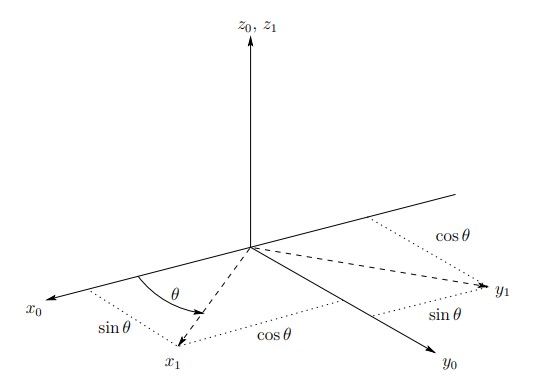


Figure ‑: Rotation about z0 by an angle

To specify the coordinate vectors of frame 1 with respect to frame 0 in three dimensions, the 3 × 3 rotation matrix is written as where the columns are the coordinates of the vectors x1, y1, and z1 expressed in frame 0.

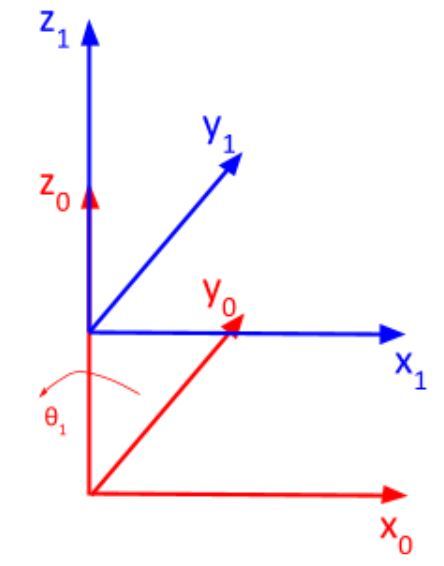
As a result, the rotation matrix R\_1^0 can be used to define the orientation of coordinate frame o1 x1 y1 z1 in relation to frame o0 x0 y0 z0 as well as to change the coordinates of a point between two frames. R01p1 represents the same location represented relative to the frame o0 x0 y0 z0 if a given point is stated related to o1 x1 y1 z1 by coordinate p1..

The basic rotation matrices representing rotations about the x, y and z-axes are given as:

## 2.4 Displacement Vector

Rotation manipulations are just half the puzzle. Translation transformation is just like a rotation one. An n×1 translation matrix (or displacement vector) specifies the displacement of one frame relative to another frame in the n-dimensional Euclidean space.

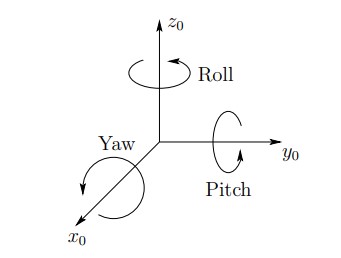
Figure ‑: Displacement vector example



However, finding displacement vectors is much easier than finding rotation transformations. A considerable quantity of kinematic and dynamic coupling between the links is frequently caused by rotational axes, resulting in an accumulation of errors and a more complex control challenge.

## 2.5 Roll, Pitch, Yaw Angles

A rotation matrix R is a sum of successive rotations around the principal coordinate axes x0, y0, and z0 in a certain order. The roll, pitch, and yaw angles are defined by these rotations. which are denoted by the letters φ, θ, ψ, and which are depicted in Figure 6.



We describe the rotational sequence as x y z, which means first a yaw about x0 through an angle ψ, then pitch about y0 through an angle θ, and finally roll about z0 through an angle φ.

Figure ‑: Roll, pitch and yaw angles

Since the successive rotations are relative to the fixed frame, the resulting transformation matrix is given by:

## 2.6 Homogeneous Transformations

Both positions and orientations have been represented. In this section, we combine these two concepts to define a rigid motion, and then we use the concept of homogeneous transformation to construct an efficient matrix representation for rigid motions. This transformation can be represented by the set of matrices of the form:

Where R denotes the rotation matrix and d the displacement vector. A homogeneous transformation is just a matrix representation of a stiff motion represented by the set of four matrices H of the kind provided in the equation above.

# Chapter: Kinematics Modeling

The forward and inverse kinematics of the hexapod are discussed in this chapter. Kinematics is the science of describing the without the use of a manipulator taking into account the forces and torques that cause the motion. As a result, the kinematic description is geometric. We start with *forward kinematics*, which involves determining the location and orientation of the end-effector[[2]](#footnote-2) given the values for the robot's joint variables. The goal of the *inverse kinematics* problem is to find the values of the joint variables given the position and orientation of the end-effector.

## 3.1 Kinematic Chains

As described in Chapter 1, a robot manipulator is made up of a series of linkages that are joined by joints. A revolute joint or a prismatic joint are examples of simple joints, whereas a ball and socket joint is an example of a more sophisticated joint.. The distinction between the two scenarios is that in the first, the joint has just one degree of freedom of motion: in the case of a revolute joint, the angle of rotation, and in the case of a prismatic joint, the amount of linear displacement. A ball and socket joint, on the other hand, has two degrees of freedom. In this project, we only used single degree of freedom revolute joints.

We rigidly connect a coordinate frame to each connection to perform the kinematic analysis. We attach oi xi yi zi to link i in particular. This means that when represented in the ith coordinate frame, the coordinates of each point on link I are constant, regardless of the robot's mobility. In addition, when joint i is actuated, link i and its associated frame, oi xi yi zi, also experience motion. The inertial frame, which is attached to the robot base, is designated as o0 x0 y0 z0.

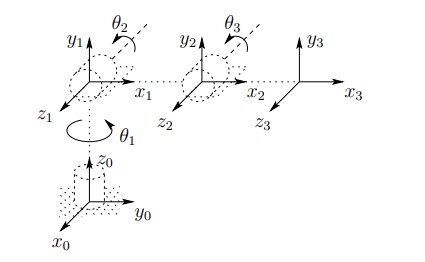


Figure ‑: Coordinate frames attached to an elbow manipulator

In the example of an elbow manipulator, Figure 7 depicts the concept of tightly attaching frames to links.

## 3.2 Forward Kinematics

The forward kinematics problem is concerned with the relationship between the various joints of the robot manipulator and the location and orientation of the tool or end-effector. The joint variables are the angles between the links in revolute or rotating joints, and the link extension in prismatic or sliding joints. In our case, to get the hexapod’s legs’ forward kinematics. Knowing the geometry of the robot legs, it is possible to develop a matrix mathematical model that, with the angles of each joint, calculates the position of the end of the leg that touches the ground (known as the end effector).

For the development of the mathematical model of the forward kinematics we used the Denavit-Hartemberg[[3]](#footnote-3) method. In this method, each joint is assigned to a Cartesian coordinate system (xi, yi, zi), which are related by Denavit-Hartemberg parameters (ai, αi, di, θi) in order to reference the position of the terminal element in the robot geometric center.

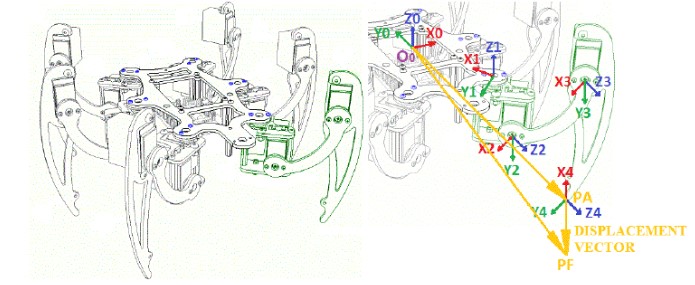


Figure ‑: Hexapod assembled, coordinate frames and displacement vector

In the Figure 8 it is possible to observe the design of the partially assembled robot, the coordinate systems assigned to each vertex, and the definition of the displacement vector, which is the vector between the current point and the future point of the terminal element. The index of Denavit-Hartemberg parameters starts at zero in the center of the robot and is incremented on each joint toward the terminal element.

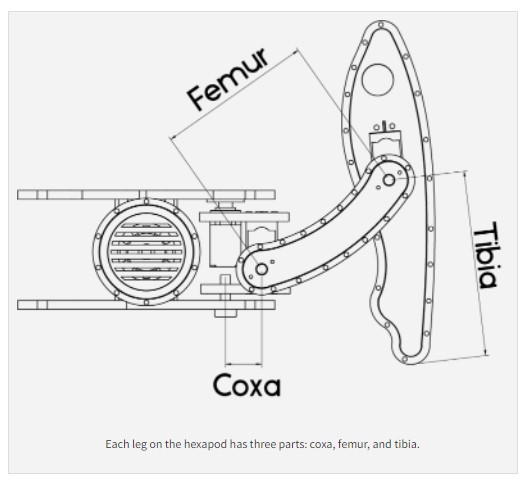
The Denavit-Hartemberg parameters represent:

ai – distance between the zi-1 and zi axes measured over the xi axis;

αi – angle between the zi-1 and zi axes measured in a normal plane to xi axis. The direction of angle is given by the right hand rule in the xi axis.

di – distance between the zi-1 and zi axes measured over the zi-1 axis;

θi – angle between the xi-1 and xi axes measured in a normal plane to zi-1 axis. The direction of angle is given by the right hand rule in the zi-1 axis.



The application of this method in a structure such as the legs of the robot, where all the joints are revolution joints, resulting in constant parameters ai, αi, di and just θ1, θ2, θ3 are variables, which are the angle of the joints.

So, with the angles of each joint, it is possible to calculate the coordinates that determine the position of the end effector (end of the leg) in the base coordinate system, which is located at the Coxa joint.

Figure ‑: Hexapod leg parts

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *i* | *ai* | *αi* | *di* | *θi* |
| 1 | COXA | π/2 | 0 | *θ1* |
| 2 | FEMUR | 0 | 0 | *θ2* |
| 3 | TIBIA | 0 | 0 | *θ3* |

Table 1: DH parameters for one robot leg

Now each link has his own homogeneous transformation Ai represented by:

The Global Homogeneous Transformation Matrix T is created by multiplying the three homogeneous translation matrices. It reflects the transformation of the end effector coordinate system to the base coordinate system and gives the forward kinematics of a robot leg.

By using Table 1, the T-matrix is thus given by:

The transformation matrix for each joint can be defined as the following:

## 3.3 Inverse Kinematics

We showed how to determine the end-effector position and orientation in terms of joint variables in the preceding section. However, this is not the case in real life situations; where you only have the initial coordinates of each leg’s tip and you want to make the robot move and walk based on this information. Well, this where inverse kinematics comes.

The inverse kinematic problem, from its name you can tell that it’s the opposite of the standard kinematics problem, so locating joint variables in terms of end-effector position and orientation is the problenm, which is a lot more complex than the forward kinematics problem.

The geometric methodology has been utilized to generate inverse kinematics equations, as well as the representation of 3-D design on 2-D, design into two figures, among other techniques to derive inverse kinematics equations. The first one provides a top view of leg motion in x, y coordinates, while the other one provides the Z axis.

Figure ‑: Top and side view of a hexapod leg

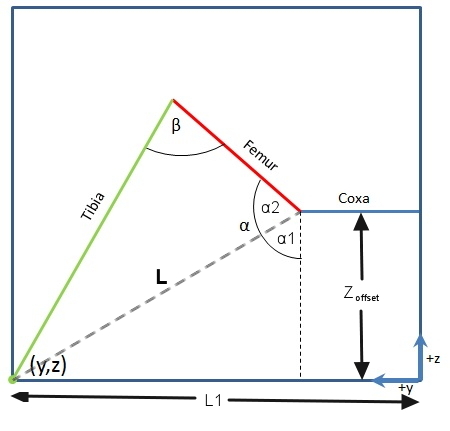
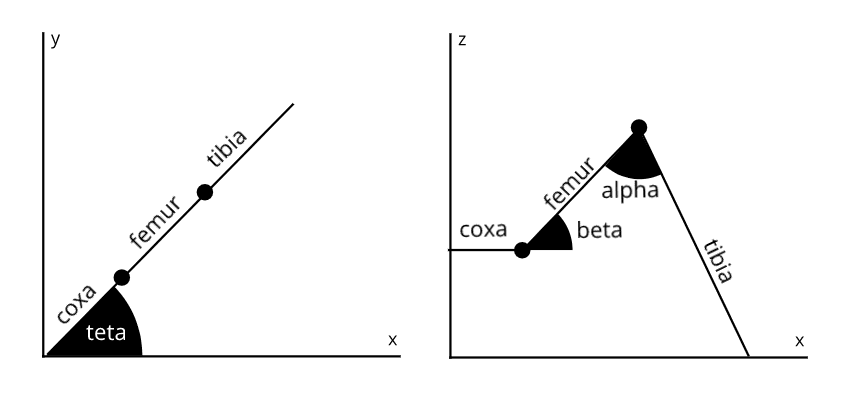


Figure ‑: Clearer side view of robot leg

The angle θ1 can be obtained by inverse tangent function:

The length of the leg is the hypotenuse of the triangle in figure 10, and it can be calculated by using Pythagorean Theorem.

Figure 10 represents the side view of a leg in y, z coordinates and θ2 and θ3, the resultant vector ‘L’ divides the total angle between the lines Tibia and L1-Coxa into two angle and its magnitude can be given by:

α1 the angle between L and the Zoffset can be found by inverse tangent function:

α2 can be calculated using the law of cosines in the triangle formed Femur, Tibia and L

And then the resulting angle α can be found by summing both α1 and α2 together.

β can also be calculated using the law of cosines in the same triangle mentioned earlier.

And thus we have all the angles of our joints.

# Chapter: Gait Analysis

A gait is a manner of traveling ahead with legs, and animals or robots with multiple degrees of freedom can use a number of gaits. Tripod Gait, Ripple Gait, and Wave Gait are the three major gaits utilized by hexapod robots.

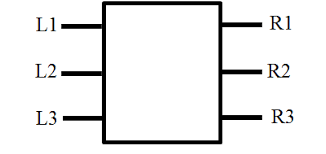


Figure ‑: Leg positions of the hexapod

## 4.1 Types of Gaits

### 4.1.1 Tripod Gait

When walking in the tripod gait, the hexapod always has three feet in touch with the ground : the front and back legs on one side and the middle leg on the other. Which side is up and which side is down is determined by the angle of the middle servo. It propels itself forward by pressing those feet back into the ground while propelling itself forward into the air with the other feet. After then, the hexapod shifts its weight to the other three feet and continues the process. It moves ahead by transferring its weight to the middle legs and then propelling the elevated feet forward. The foot fall pattern of this gait is shown in figure 13.

### 4.1.2 Ripple Gait

Three steps make up the ripple gait walking mechanism in a hexapod robot. It moves in a cyclic manner, with two legs swinging to take the next step and the remaining four maintaining ground contact and providing propulsion for the robot. In general, this is a slower pace than the tripod gait. The order in which the legs enter the swing phase to finish the cycle varies a lot. The sole criteria that must be met in order for this gait to remain stable is that the two legs in swing phase are not on the same side of the robot. The foot fall pattern of this gait is shown in figure 13.

### 4.1.3 Wave Gait

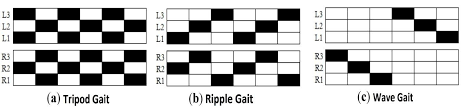
Since just a single leg is swinging at a time, the wave gait is the slowest of the three. This gait comprises a six-step cyclic pattern, in which one leg lifts off the ground to swing forward, while the other five legs remain in touch with the ground and produce propulsion. The foot fall pattern of this gait is shown in the figure 13 and produce propulsion.

Figure ‑: Footfall patterns of the different gaits

We wanted to implement all three gaits in our project; however, time wasn’t on our side, so we only implemented the tripod gait in our hexapod since it works well for both speed and stability.

## 4.2 Hexapod Stride

A leg's stride is one complete cycle in which the leg passes through both the swing and stance phases at a time. Swing and stance times are the lengths of time that the leg remains in swing and stance phases, respectively. The stride period is the complete time of a leg, which comprises one swing phase and one stance phase.

## 4.3 Stability Margin using Support Polygon

The support polygon of a system is an imaginary polygon constructed by linking all of the robot's legs that are in the stance phase, i.e. on the ground, at any given time. The robot's center of mass is then determined, and a stability margin is established. The robot is said to be unstable if the center of mass leaves the support polygon at any point, since the system loses its equilibrium and falls to the ground. Such situations must be avoided.

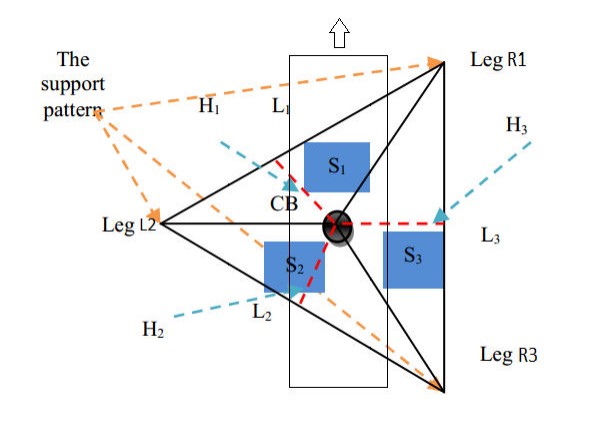


Figure ‎4‑3: Tripod gait support polygon

# Chapter: Implementation

## 5.1 Components list

### 5.1.1 The Brain

The hexapod was implemented on a Raspberry Pi Model B (Fig. 4) With a Raspbian Linux operating system installed on an 8GB SD card. The Raspberry Pi Model B is an integrated computer on a single board with a 700MHz ARM11 processor, 512MB RAM and 8 GPIOs (digital input/output).

A picture containing text, electronics, circuit

Description automatically generated

Figure ‑: Raspberry Pi Model B

### 5.1.2 Power Supply

 This hexapod has 3 servo motors in each leg, which means it has 18 motors in total. Each servo motor requires 500-900 mA to run and an operating voltage of 4.8-7.2 V. For this reason, we found that a 5V 40A power supply works well for this project

Figure ‑: Power Supply 5V 40A

### 5.1.3 Servo driver

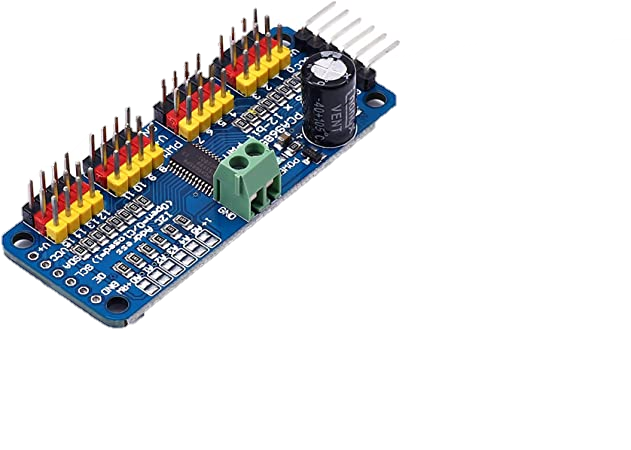
 As we said in the previous section, this hexapod has a total of 18 servo motors; and as it’s known, servo motors need PWM signals to function. However, the raspberry pi model 3B has only 3 PWM pins. For this reason, we need a servo driver that is capable of powering 18 servos. So as our final decision, we got ourselves 2 of Servomotor driver VKLSVAN PCA9685; each one is capable of powering 16 servos.

Figure ‑: Servo driver VKLSVAN PCA9685

### 5.1.4 Rest of the components

* MG996R servo package x18 (figure 18) used as joints for the legs.
* Raspberry-pi-camera-module-v2 (figure 19) to stream live footage.
* HC-SR04 Ultrasonic sensor (figure 20) to measure distance.

A picture containing shape

Description automatically generated

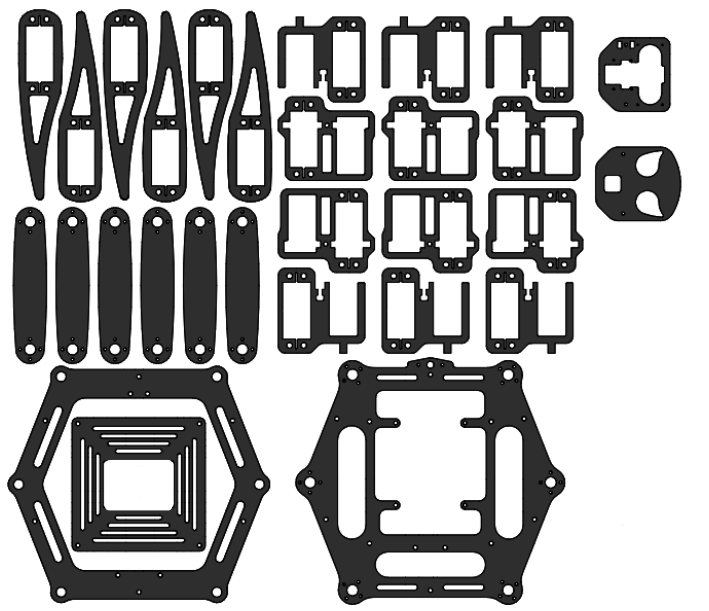
Figure ‎5‑4: MG996R servo motor

A picture containing electronics, circuit

Description automatically generated

Figure ‑: Raspberry-pi-camera-module-v2

Figure ‑: HC-SR04 Ultrasonic sensor



## 5.2 Wiring

Figure ‑: Acrylic Parts

### 5.2.1 VKLSVAN PCA9685 Driver/ raspberry pi:

* Raspberry 3V3 to Driver VCC
* Raspberry GND to Driver GND
* Raspberry SCL to Driver SCL
* Raspberry SDA to Driver SDA
* Servo Data wire to Driver PWM on channel 0
* Servo VCC wire to Driver V+ on channel 0
* Servo GND wire to Driver GND on channel 0

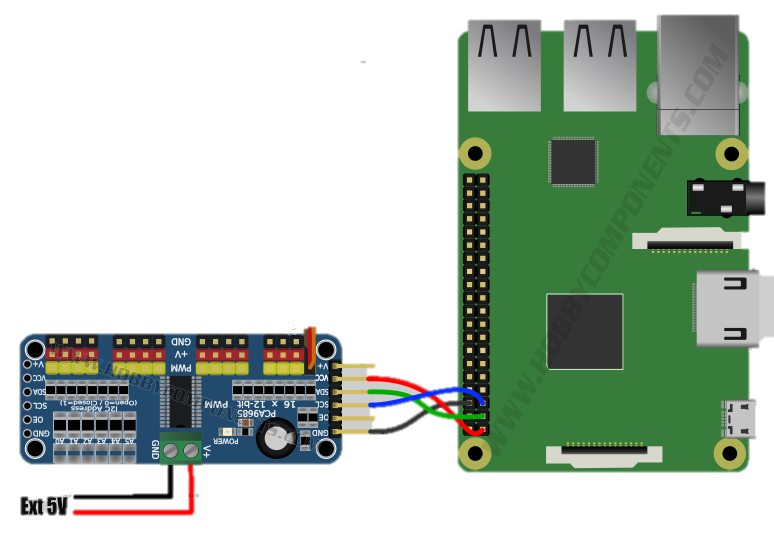
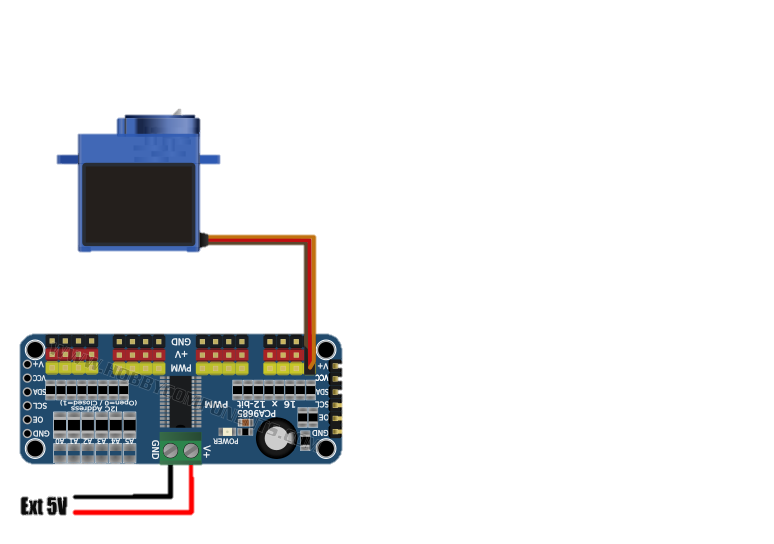
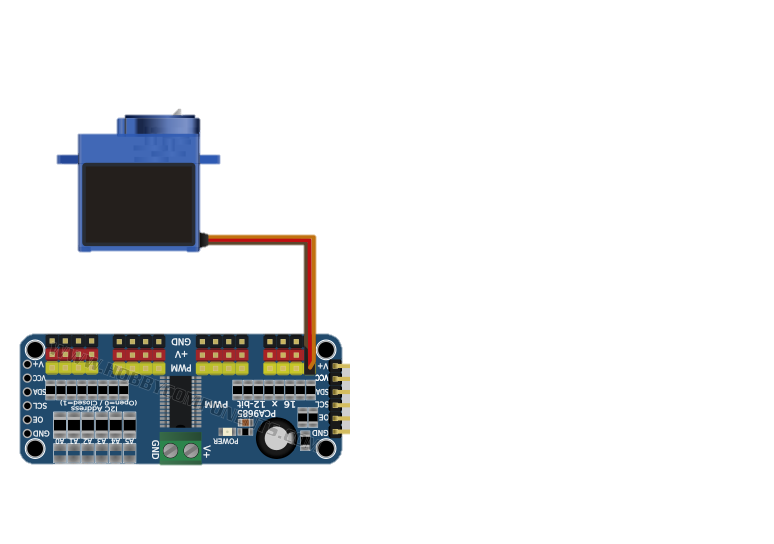


Figure ‑: Raspberry pi to servo driver wiring

### 5.2.2 MG996R Servomotor/ VKLSVAN PCA9685 Driver:

* GND to GND
* OE to OE
* SCL to SCL
* SDA to SDA
* VCC to VCC
* V+ to V+
* Solder A0





X9 X9



↑

Solder

Figure ‑: MG996R to the servo driver wiring

### 5.2.3 Raspberry pi-camera-module-v2/ Raspberry pi:

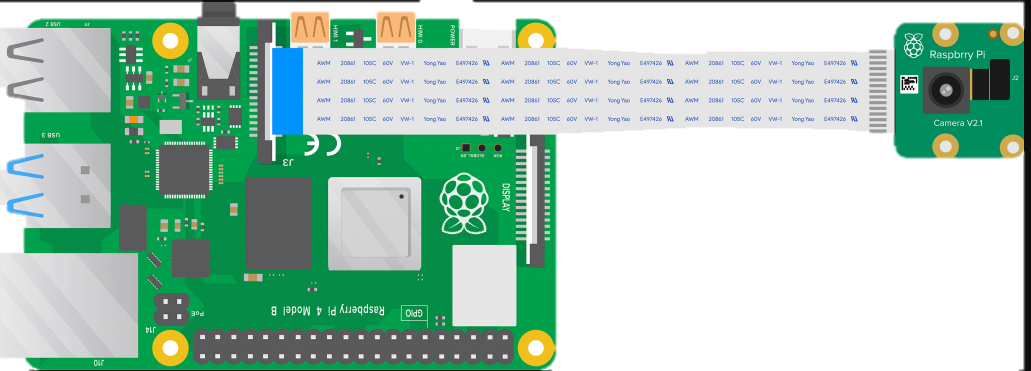


Figure ‑: Pi camera with raspberry pi

### 5.2.4 HC-SR04 Ultrasonic sensor/ Raspberry pi:

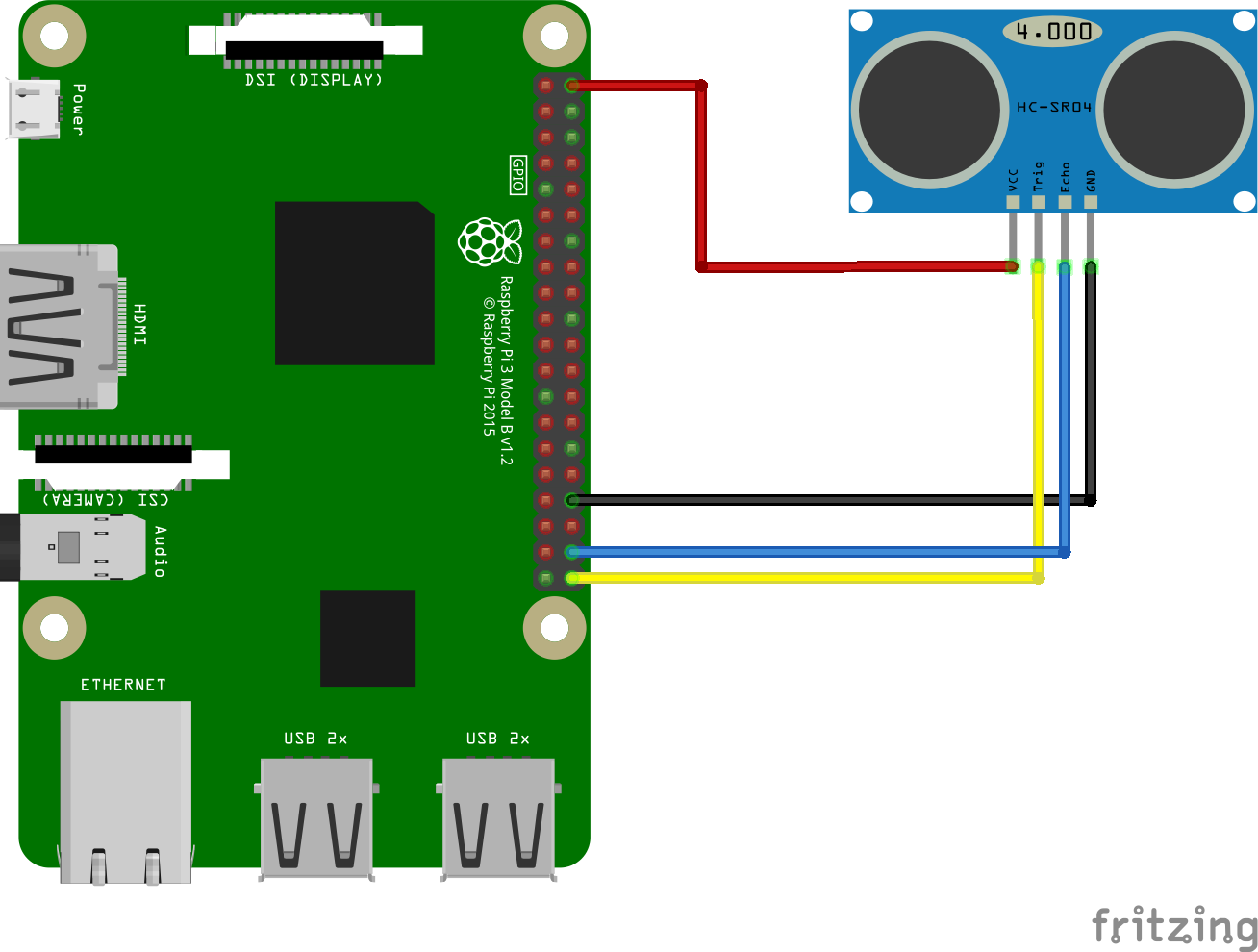
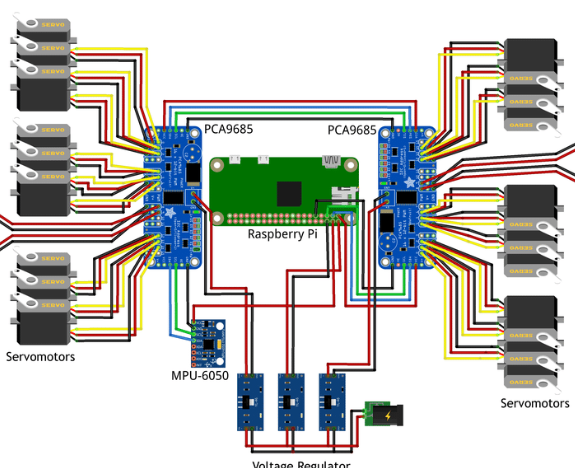
* VCC to 5v
* Trig to GPIO 21
* Echo to GPIO 20
* GND to GND

Figure ‑: HC-SR0 ultrasonic sensor with raspberry pi



Power Supply 5V 40A

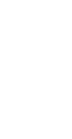
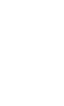


Figure ‎5‑12 full wiring diagram

### 5.2.5 Power supply Wiring:

A picture containing text, electronics, circuit

Description automatically generated

Figure ‑: Power supply wiring

## 5.3 Programming

Finally, for the programming part; since we’re very comfortible with python and it’s compatible with the raspberry pi, we used it as our main programming language. We used it for all our matrices calculations and hexapod movements.

For the app development, as I mentioned earlier, we used C# with the xamrin framework to develop it.

## 5.4 Economical Study

|  |  |
| --- | --- |
| Components | Price in USD |
| 18x Servomotors | 70 |
| Raspberry pi | 50 |
| Power Supply | 20 |
| Gyroscope Sensor | 3 |
| Ultrasonic Sensor | 2 |
| Pie Camera | 8 |
| 2x Servo Drivers | 17 |
| Plexi Body | 30 |
| Knots & Bolts | 10 |
| Wires | 2 |

Table 2: Economical Study



# Chapter: Mobile Application

There are many ways to control the hexapod wirelessly. We can do it through the bluetooth that can be attached to the arduino or raspberry pi, or through an already designed application. However, for this hexapod we decided to develop our own application that communicates with the hexapod through the mqtt module. All required details and information will be discussed in this chapter.

## 6.1 Xamarin Forms

We chose Xamarin forms as our mobile development framework for one very important reason. It’s an open source cross-platform framework from Microsoft for building iOS, Android, & Windows apps with .NET from a single shared codebase. Which means that we only write the code once and by its own it’ll be compiled into native IOS and Android apps, which saves a ton of time. The design layout for this mobile application will be shown in the next figures:

Figure ‑: Xamarin logo



When you open the app for the first time, the first page (figure 16) will be shown; for scurity measure the user is asked to enter the username and password required for the app, which are “user“ and “123“ respectively. After completing this step, the second page (figure 17) is prompted, and here the connection with the hexapod is established. The connect button is the one to establish this connection, and if the hexapod is not available to connect the coonect button won’t be showned, instead a refresh button is shown; from its name you can tell that its function is to refresh the app until the hexapod is ready or available to connect. After connecting to the hexapod, the third page is opened (figure 18), and this is where the user choses in which mode he wants to drive the hexapod. There are 2 modes. Manual and Automatic (not finished). Next, after choosing which mode the user desires, the final page (figure 19) and the most important one is prompted. This is where all the commands and controls happen. Here the user can find all the buttons required to move the hexapod, and make it walk forward and backward, turn right and left, pitching, yawing and rolling. All the pages have a log out buuton, this button, when pressed, takes the user back to the initial page and ask him again to log in.

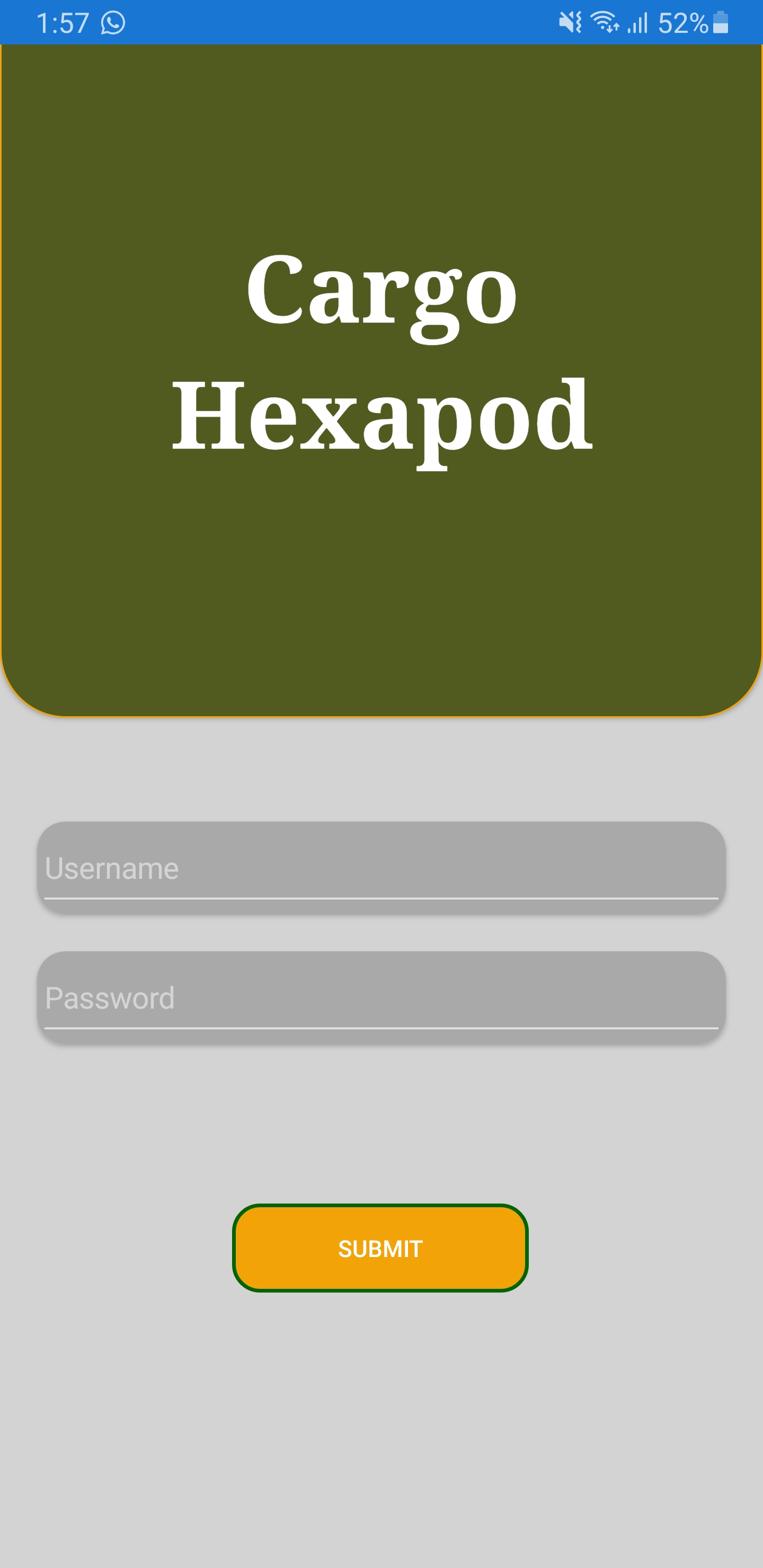


Figure ‑: Second Page

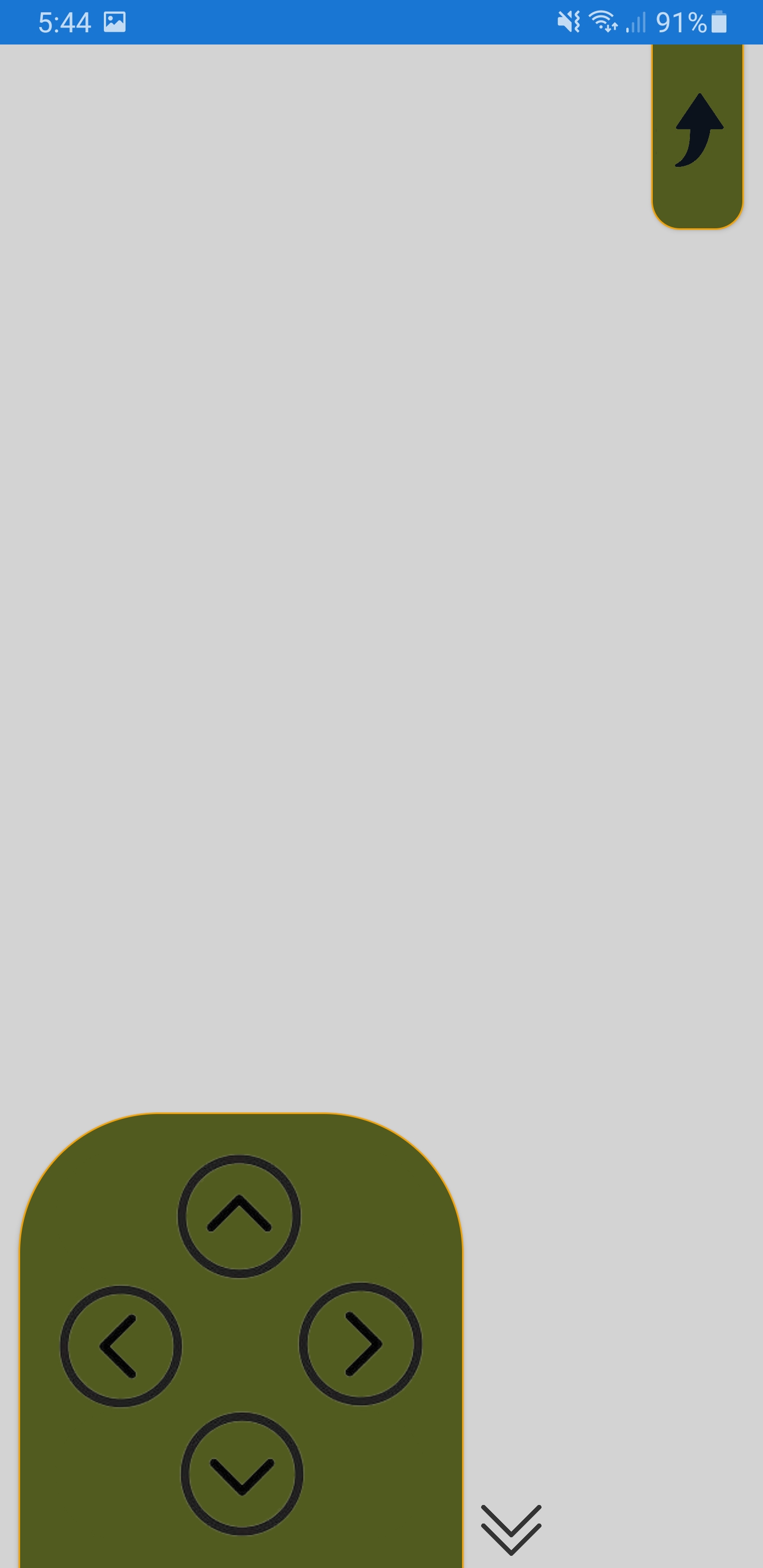


Figure ‑: First Page

Figure ‑: Third Page

Figure ‑: Fifth Page

Figure ‑: Forth Page



## 6.2 MQTT

MQTT is a low-bandwidth publish/subscribe protocol for connecting IoT devices with a tiny footprint. Unlike HTTP's request/response approach, MQTT is an event-driven protocol that allows messages to be pushed to clients. This architecture isolates clients, enabling for a highly scalable system that does not rely on data producers and data consumers. However, this protocol can’t be established without the broker. The broker is the one responsible to handle connections between clients.

Both the mqtt client and broker and more will be explained in this section.

### 6.2.1 Client

An MQTT clients are divided into 2 parts, subscriber and publisher; however, they are all referred as mqtt clients. The publisher and subscriber are to just to label whether the client is currently publishing messages or subscribed to recieve messages. Any device (from a microcontroller to a full-fledged server) that runs a mqtt library and connects to a mqtt Broker via the internet is considered a mqtt client. An MQTT client is essentially any device that talks MQTT across a TCP/IP stack.

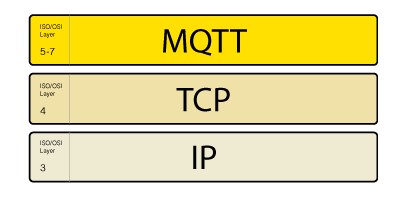
### 6.2.2 Broker

The broker is the brain of any publish/subscribe mqtt model. A broker can manage up to millions of MQTT clients connecting at the same time. It is in charge of receiving all messages, screening them, determining who has subscribed to each message, and then sending the message to those who have subscribed. All clients with persistent sessions, including subscriptions and missed messages, have their session data held by the broker (more details). The broker's other responsibilities include customer authentication and authorization.

### 6.2.3 MQTT Connection

TCP/IP is the foundation of the MQTT protocol. A TCP/IP stack is required for both the client and the broker.

Figure ‑: MQTT connection



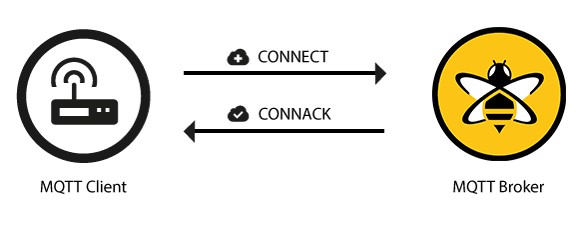
 The MQTT connection between a client and a broker is always established. Clients never connect with one another directly. To establish a connection, the client sends a CONNECT message to the broker. The broker responds with a CONNACK message and a status code.The broker maintains the connection open until the client sends a disconnect command or the connection breaks.

Figure ‑: MQTT connection preview

# Conclusion

## Results

The hexapod results can be divided into 2 parts, the hardware development and the kinematics analysis. Fortunately, both of these tasks were accomplished perfectly. All the body parts were assembled flawlessly to make a perfect hexapod.

Figure ‑: Hexapod image 1

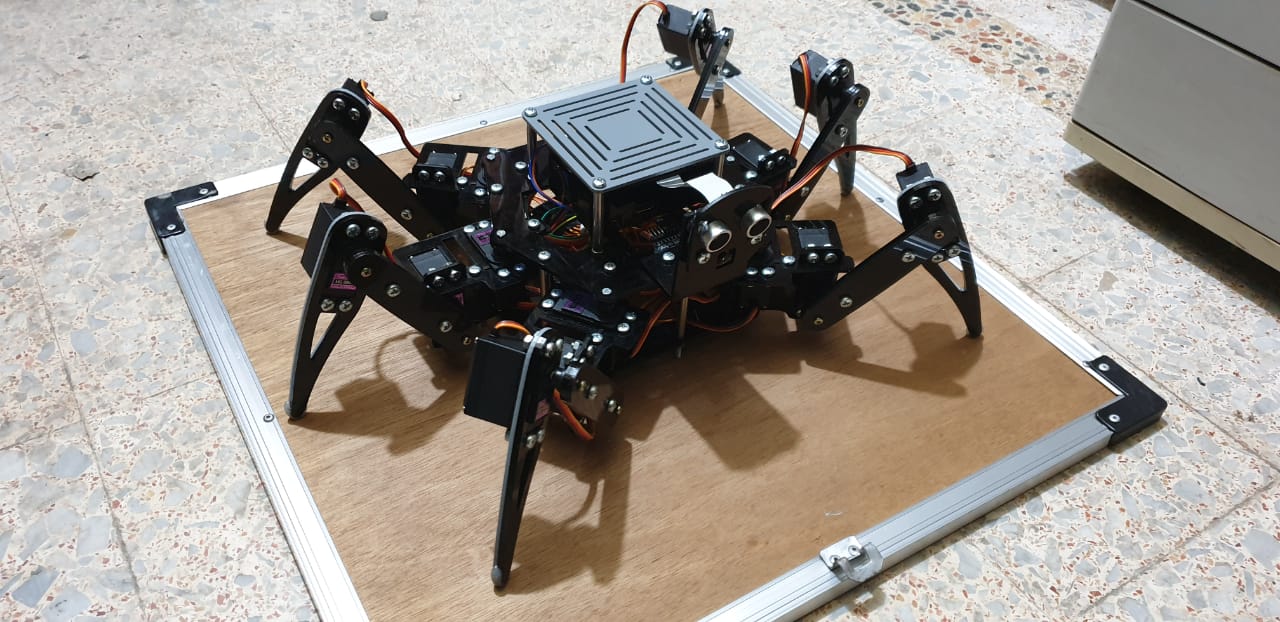


Figure ‑: Hexapod Image 2

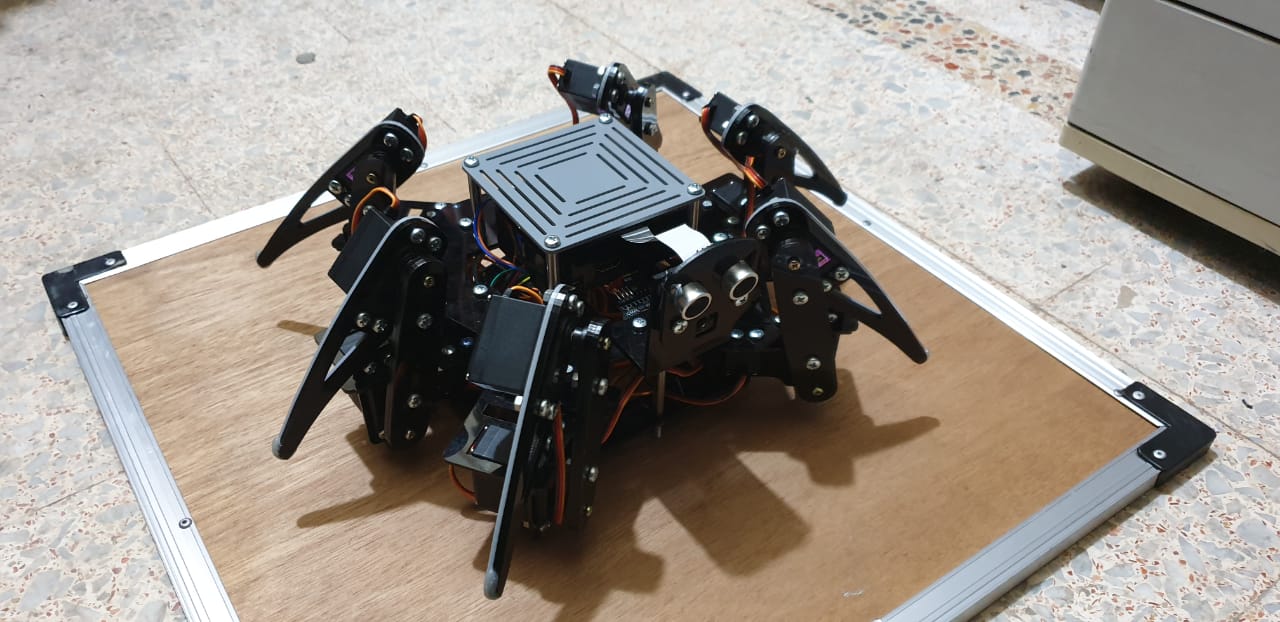
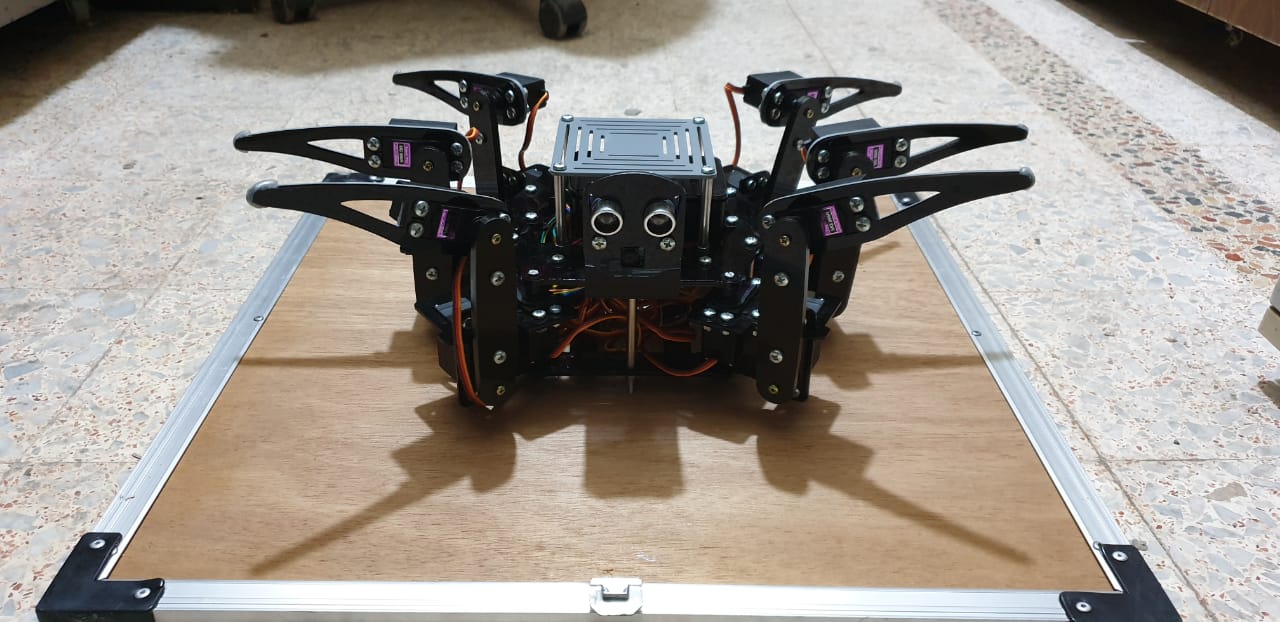


Figure ‎7‑3: Hexapod image 3

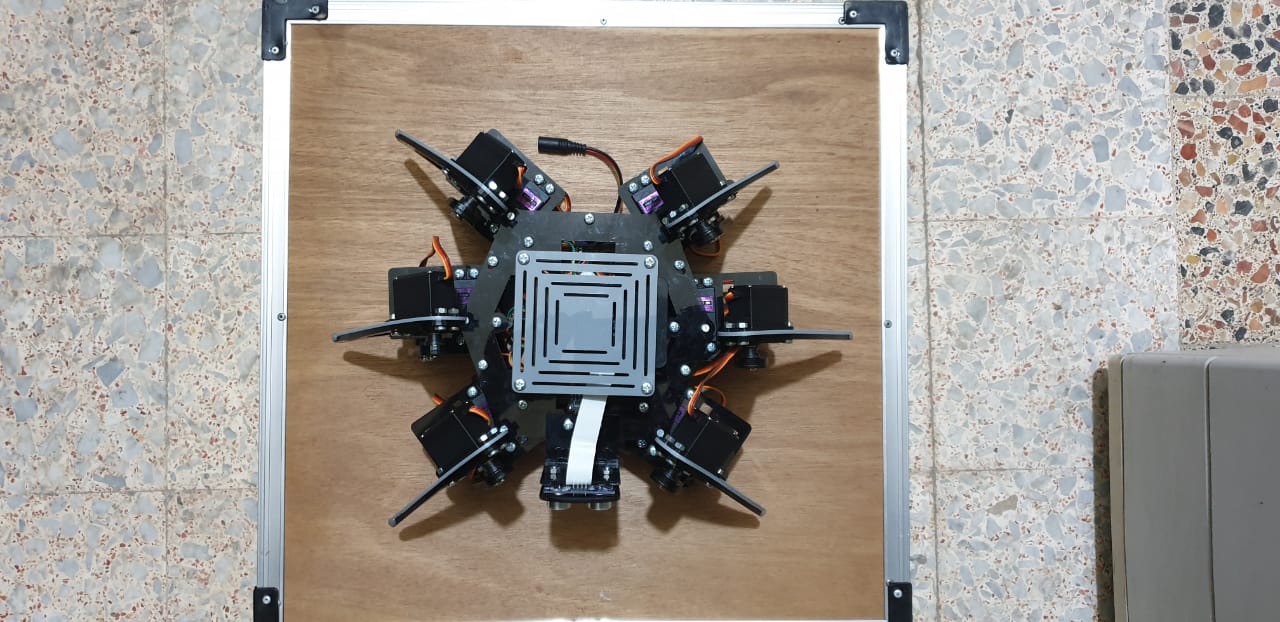


Figure ‑: Hexapod image 5

For the kinematics, we faced lots of problems, specifically in the inverse kinematics part. After all, it was inevitable to face these problems, here and in the other parts as well, especially when we didn’t take those parts at the university, we had to learn lots of those parts in our own. To check our inverse kinematics results, we did a graph that computes the results returned by the inverse kinematics function and computes them in the forward kinematics method, and then we compared the coordinates returned by this function to the real world coordinates. We did a graph for each leg to show those comparisons and the small error between both of those functions The blue represents the real position and the orange one represents our function result, keep in mid there’s real world callibration so of course those tiny errors were expected.

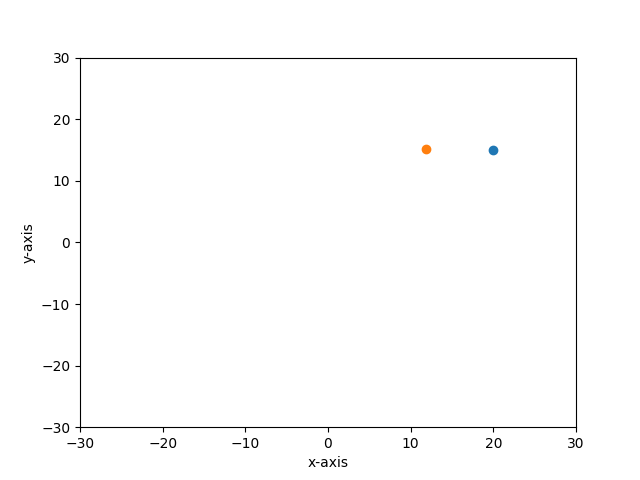
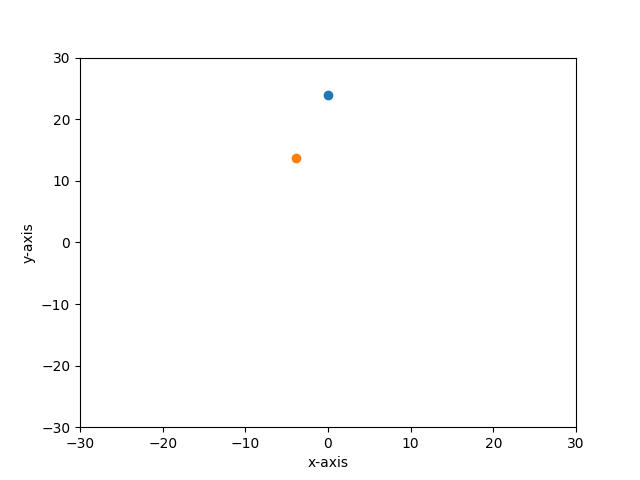


Figure ‑: Leg 2 error

Figure ‑: Leg 1 error

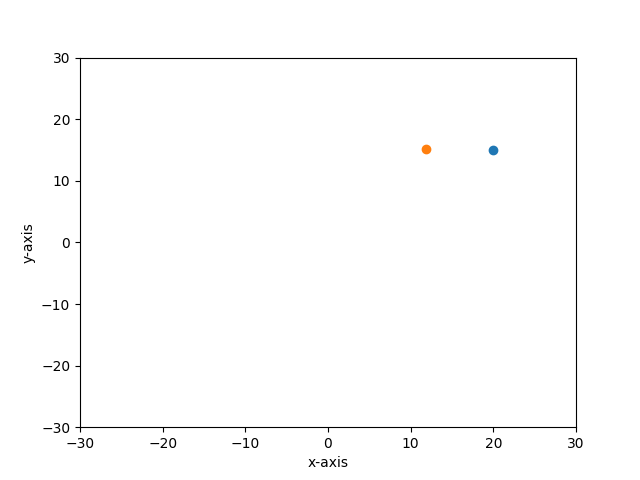
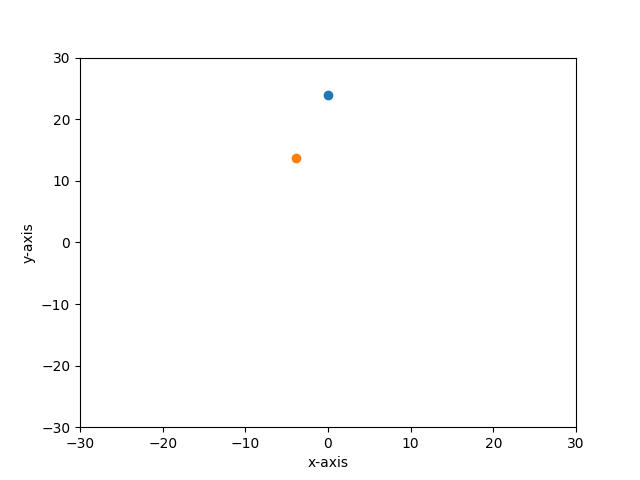


Figure ‑: Leg 4 error

Figure ‑: Leg 3 error

## Figure_1

Figure ‑: Leg 6 error

## Figure_6

Figure ‑: Leg 5 error

## Future Work

For everything mentioned before, it is all a study on the prototype and not the actual product. That is because in every business plan, the first thing to do after the ideation part is making a prototype that shows a small description of what the actual product delivers to the customer and getting the required feedback to modify or change any part to make sure of the efficiency and benefits of this product. The next step is of course working on the bigger project and providing the desired solution to the main problem.

# References

DARBHA, N.H. (2014). *AN OPTIMIZATION STRATEGY FOR HEXAPOD GAIT TRANSITION*. Jawaharlal Nehru Technological University

Høifødt, H. (2011). *Dynamic Modeling and Simulation of Robot Manipulators.* Norwegian University of Science and Technology

Chen, W., Ren, G., Zhang, J., Wang, J. (2012) *Smooth transition between different gaits of a hexapod robot via a central pattern generators algorithm*. DOI 10.1007/s10846-012-9661-1

Ferreira, N. M. F., Santos, F.N, Ferreira, J., Santos, J.C., Santos, F., Silva, M. (2018) *Education with robots inspired in biological systems. .* Knowledge Research Group on Intelligent Engineering and Computing for Advanced Innovation and Development

Letang, R.B., Mori, R.N., Campo, A.B. (2015). *DEVELOPMENT OF AN AUTONOMOUS HEXAPOD ROBOT WITH*

*DIGITAL IMAGE PROCESSING IMPLEMENTED ON RASPBERRY PI,* 23.

Spong, M.W., Hutchinson, S. & Vidyasagar, M. *Robot Modeling and Control* (First Edition)

Raheem, F.A & Khaleel, H.Z. (2014). *Static Stability Analysis Of Hexagonal Hexapod robot For The periodic Gaits.* University of Technology, Baghdad-Irag.

The HiveMQ Team(2019, July 17). MQTT Client and Broker and MQTT Server and Connection Establishment Explained - MQTT Essentials: Part 3. <https://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment/>

https://www.katranji.com/item/248356

1. A Degree of freedom, in robotics, is an independent joint that can provide freedom of movement of the manipulator, either in a rotational or translational (linear) sense. [↑](#footnote-ref-1)
2. In robotics, an end effector is a device or tool that's connected to the end of a robot arm where the hand would be. The end effector is the part of the robot that interacts with the environment [↑](#footnote-ref-2)
3. A commonly used convention for selecting frames of reference [↑](#footnote-ref-3)