
UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,
TÎRGU-MUREȘ
SPECIALIZAREA AUTOMATICĂ ȘI INFORMATICĂ
APLICATĂ

ROBOT HEXAPOD
CONTROLABIL

PROIECT DE DIPLOMĂ

Coordonator științific:

Dr. Szántó Zoltán

ing. Fehér Áron

Absolvent:

Ciulei Daniel

2022

UNIVERSITATEA “SAPIENTIA” din CLUJ-NAPOCA Facultatea de Științe Tehnice și Umaniste din Târgu Mureș Specializarea: <u>Automatică și informatică aplicată</u>		Viza facultății:
LUCRARE DE DIPLOMĂ		
Coordonator științific: dr. ing. Szántó Zoltán, ing. Fehér Áron	Candidat: Ciulei Daniel Anul absolvirii: 2022	
<p>a) Tema lucrării de licență: Robot hexapod controlabil</p> <p>b) Problemele principale tratate:</p> <ul style="list-style-type: none"> - Probleme teoretice: geometria roboților mobili, planificarea mersului tip arahnidă - Probleme practice: proiectarea și realizarea unui sistem încorporat pentru reglarea robotului tip hexapod, PCB, proiectarea șasiului și imprimarea 3D <p>c) Desene obligatorii:</p> <ul style="list-style-type: none"> - Schema bloc a sistemului de comandă - Diagramele firmware-ului și ale programului Android <p>d) Softuri obligatorii:</p> <ul style="list-style-type: none"> - Programul de comandă a robotului - Programul controller sub Android <p>e) Bibliografia recomandată:</p> <p>[1] Delcomyn, Fred, and Mark E. Nelson. "Architectures for a biomimetic hexapod robot." <i>Robotics and Autonomous Systems</i> 30.1-2 (2000): 5-15.</p> <p>[2] Ricardo Campos, Vitor Matos, Miguel Oliveira, Cristina Santos. "Gait generation for a simulated hexapod robot: a nonlinear dynamical systems approach" 2010 DEI - Artigos em atas de congressos internacionais</p>		
<p>f) Termene obligatorii de consultații: săptămânal</p> <p>g) Locul și durata practicii: Universitatea Sapientia, Facultatea de Științe Tehnice și Umaniste din Târgu Mureș, laborator 243, respectiv on-line</p> <p>Primit tema la data de: 15.05.2021</p> <p>Termen de predare: 06.07.2022</p>		
Semnătura Director Departament Semnătura responsabilului programului de studiu	Semnătura coordonatorului Semnătura candidatului	

Model tip a.

Declarație

Subsemnata/ul Ciulei Daniel, absolvent(ă) al/a specializării Automatica și Informatica Aplicată, promoția 2022 cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Localitatea, Târgu Mureș

Data:

Absolvent

Semnătura.....

Extras

Robotica a parcurs un drum lung de la primul robot operat digital care a fost instalat pe o linie de asamblare. În zilele noastre roboții sunt atât de răspândiți încât îi putem găsi în aproape toate sectoarele piețelor, de la aplicații militare la sisteme agricole și electrocasnice. Inventatorii și-au dat seama că multe dintre aceste aplicații ar beneficia și chiar ar face roboții lor mai agili, prin imitarea naturii. De aceea unii roboți seamănă cu animale, plante sau sisteme care au evoluat pe Pământ. Această practică se numește biomimetism și a ajutat mulți ingineri să rezolve probleme complexe cu metode simple care se găsesc în natură. Acest concept ne-a dat și ideea de a crea un Hexapod care ar putea fi controlat de la distanță.

Scopul acestui proiect este de a crea un Hexapod, un robot cu șase picioare asemănător unei arahnide, pe care un utilizator îl poate controla printr-o aplicație pentru smartphone. Comunicarea între aplicația pentru smartphone și unitatea de control a Hexapod se va realiza printr-un protocol de rețea. Locomoția robotului va semăna cu mersul arahnidelor.

Structura mecanică a robotului este proiectată cu ajutorul software-ului Autodesk Inventor CAD, unde fiecare picior al Hexapod-ului va avea 4 grade de libertate. O placă de bază personalizată a fost proiectată în EasyEDA pentru a găzdui microcontrolerul robotului, un circuit de secvență de pornire/oprire și un circuit de măsurare a distanței în infraroșu pentru a determina dacă un picior atinge pământul sau nu. Firmware-ul MCU a fost programat în MicroPython, planificarea mersului și a traiectoriei robotului a fost creată cu ajutorul mediului MATLAB. Android Studio a fost folosit pentru crearea aplicației pentru smartphone, în timp ce protocolul de rețea MQTT, cu un broker MQTT care rulează pe un Raspberry Pi, este utilizat pentru comunicarea dintre robot și utilizator.

Cuvinte cheie: hexapod, robot, MQTT, design, control

**SAPIENTIA ERDÉLYI MAGYAR
TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR
SZÁMÍTÁSTECHNIKA SZAK**

IRÁNYÍTHATÓ HEXAPOD ROBOT

DIPLOMADOLGOZAT

Témavezető:

Dr. Szántó Zoltán

drd. Fehér Áron

Végzős hallgató:

Ciulei Daniel

2022

Kivonat

A robotika hosszú utat tett meg az első digitálisan működtetett robot óta, amelyet összeszerelősorra telepítettek. Napjainkban a robotok annyira elterjedtek, hogy a piac szinte minden szektorában megtalálhatóak, a katonai alkalmazásoktól a mezőgazdasági rendszerekig és háztartási gépekig. A feltalálók rájöttek, hogy ezek az alkalmazások közül sok előnyt jelent, sőt, még mozgékonyabbá is tenné robotjaikat a természet utánzásával. Ez az oka annak, hogy egyes robotok olyan állatokra, növényekre vagy rendszerekre hasonlítanak, amelyek a Földön fejlődtek ki. Ezt a gyakorlatot biomimikrinek nevezik, és sok mérnöknek segített összetett problémák megoldásában a természetben megtalálható egyszerű módszerekkel. Ez a koncepció adta az ötletet egy távolról irányítható Hexapod létrehozására is.

A projekt célja egy koncepció igazolás megvalósítása egy Hexapod, hatlábú robot, számára, amelyet a felhasználó okostelefonos alkalmazáson keresztül vezérelhet. Az okostelefon alkalmazás és a Hexapod vezérlőegysége közötti kommunikáció hálózati protokollon keresztül történik. A robot mozgása a pókfélék járására fog hasonlítani.

A robot mechanikai felépítését az Autodesk Inventor CAD szoftvere segítségével terveztem meg, ahol a Hexapod minden lábának 4 szabadságfoka lesz. Az EasyEDA-ban egyedileg elkészített alaplapot terveztem, amely befogadja a robot mikrokontrollerét, egy bekapcsolási/leállítási szekvencia áramkört és egy infravörös távolságmérő áramkört annak meghatározására, hogy egy láb érinti-e a talajt vagy sem. Az MCU firmware-ét MicroPythonban programoztam, a járás- és robotpálya-tervezést a MATLAB környezet segítségével készítettem el. Az okostelefonos alkalmazás létrehozásához az Android Studio, míg a robot és a felhasználó közötti kommunikációhoz az MQTT hálózati protokoll, egy Raspberry Pi-on futó MQTT brókerrel.

Kulcsszavak: hexapod, robot, MQTT, tervezés, irányítás

Abstract

Robotics has come a long way since the first digitally operated robot which was installed on an assembly line. Nowadays robots are so widespread that we can find them in almost every sector of the markets, from military applications to agricultural systems and household appliances. Inventors have realized that many of these applications would benefit, and even make their robots more agile, by imitating nature. That is why some robots resemble animals, plants or systems that evolved on Earth. This practice is called biomimicry and has helped many engineers in solving complex problems with simple methods that are found in nature. This concept also gave us the idea of creating a Hexapod that could be controlled remotely.

The goal of this project is to create a proof of concept for a viable solution for a Hexapod, a six-legged robot resembling an arachnid, that a user can control through a smartphone application. Communication between the smartphone application and the Hexapod's control unit is achieved through a network protocol. The locomotion of the robot will resemble the gait of arachnids.

The mechanical structure of the robot is designed with the help of Autodesk's Inventor CAD software, where each leg of the Hexapod will have 4 degrees of freedom. A custom-made motherboard was designed in EasyEDA to accommodate the robot's microcontroller, a powerup/shutdown sequence circuit and an infrared distance measurement circuit to determine if a leg is touching the ground or not. The MCU's firmware was programmed in MicroPython, the gait and robot trajectory planning were created with the help of the MATLAB environment. Android Studio was used for the creation of the smartphone application, while the MQTT network protocol, with a MQTT broker running on a Raspberry Pi, is utilized for the communication between the robot and the user.

Keywords: hexapod, robot, MQTT, design, control

Table of contents

Table of figures	11
List of tables	12
1. Introduction.....	13
1.1. Objectives.....	13
2. Literature review	15
2.1. Existing robots	15
2.2. Hexapod designs	17
3. Architecture	20
3.1. Mechanical design.....	21
3.1.1. Physical structure	21
3.1.2. CAD design.....	22
3.2. Hardware design.....	23
3.2.1. Motherboard schematic.....	23
3.2.2. Additional components	24
3.2.3. Communication protocols	26
3.2.4. Communication protocols	28
3.3. Kinematics and gait analysis.....	30
3.3.1. Forward kinematics	31
3.3.2. Inverse kinematics.....	33
3.3.3. Proposed gaits	35
3.3.4. Path generation for gait implementation	36
3.4. Software design.....	39
3.4.1. Software requirements	40
3.4.2. MCU firmware	41
3.4.3. MQTT network	42
3.4.4. Android application.....	43
4. Conclusions.....	46
5. Future development possibilities	48

6.	Bibliography	49
7.	Appendices.....	50
7.1.	Appendix A – Mechanical deisgn	50
7.2.	Appendix B – Hardware design	54

Table of figures

Figure 1: University of Rome's hexapod	15
Figure 2: LAURON and LAURON V	16
Figure 3: PhantomX Hexapod MK-IV	17
Figure 4: Hexapod leg types.....	18
Figure 5: Hexapod leg orientation.....	18
Figure 6: Hexapod joint configuration	19
Figure 7: Hexapod architecture	20
Figure 8: TowerPro MG90S.....	21
Figure 9: LiPo Battery Pack	24
Figure 10: XL4016 Buck Converter.....	25
Figure 11: PWM Driver board	26
Figure 12: I2C wiring schematic	27
Figure 13: SPI wiring schematic	28
Figure 14: Motherboard PCB.....	29
Figure 15: PCB 3D model top and bottom side view	30
Figure 16: Kinematic diagram of a leg.....	31
Figure 17: Vertical movement.....	37
Figure 18: Horizontal movement	38
Figure 19: Omnidirectional movement	39
Figure 20: MCU software architecture.....	42
Figure 21: MQTT network.....	43
Figure 22: Menu activity	44
Figure 23: Controller activity	45
Figure 24: Description dialog.....	45

List of tables

Table 1: Denavit-Hartenberg table for Hexapod leg	32
Table 2: Metachronal gait leg cycles.....	35
Table 3: Tripod gait leg cycles	36

1. Introduction

Since the creation of the first robots, humanity was amazed by their diverse applications and forms that they took. From household appliances or military applications, these automatons are slowly taking over more and more parts of our daily lives. We can definitely feel their presence wherever we are, and this will be felt to an even greater extent in the coming years.

Even though most of these robots first saw usage in the military, people have come to realize that they also serve a good way to have fun. That is why some robots are now being marketed as toys suitable for kids or, with a more modular design, for hobbyists or as household items. These are often found in radio-controlled (RC) vehicles such as cars, boats, drones, or in some sort of internet of things (IoT) applications with notable examples being line tracing robots and companion robots for the elderly and children.

Some of these robots are even used in reconnaissance and surveillance. Because of their agile build and robust design, the above-mentioned examples are an easy choice for exploring and venturing into hazardous environments or dangerous areas where humans would be prone to injury. Such automatons can be controlled from a safe distance or have them do the work all by themselves. More often than not, they usually have a build that mimics an animal.

Biomimicry served as a base foundation for plenty of robot designs that are now being considered to one day walk on other planets e.g., NASA's ATHLETE [1] six-legged lunar rover, or socially assistive robots such as Honda's ASIMO humanoid robot. Replicating nature has many advantages since we are taking ideas from creatures and systems that successfully survived many phases of natural selection, meaning nature has already done the heavy lifting for us.

These ideas and robots have inspired us to create our own Hexapod robot. A six-legged robot which is controlled by a smartphone application through a network protocol.

1.1. Objectives

The goal for this project is to create a proof of concept, a viable solution for a six-legged robot. As a consequence of budget and time constraints the manufacturing of a fully working legged mobile robot is not within the reach of this project. As for the demonstration of the concept, the robot will be suspended above ground for the user to be able to simulate the gaits and other features of the robot.

The Hexapod should have a robust design with 24 DoF, thus for each leg there will be 3 servomotors that will serve as revolute joints and one piston mechanism that will form a prismatic

joint. The control circuit of the robot will be placed in the middle of the body, as well as a battery pack below this hardware, since the Hexapod will be controlled remotely without any tethered connection. A shell should also be created for the robot to prevent damaging the hardware.

For the control unit of the Hexapod an electronic schematic will be designed with the help of an EDA software. Using this schematic, a PCB will be manufactured. This PCB, on which a MCU will be located, will serve as a type of motherboard for the robot.

A smartphone application will be created for the end-user to easily control the Hexapod. This application should resemble a controller with joysticks, directional pads and face buttons. On this controller, the user should also be able to see statuses of the robot, such as battery level, speed and orientation. It should also display warning messages to the user if there is something wrong.

For the Hexapod to receive commands from the user and send back status messages, a network architecture will be created. This will be accomplished with the help of an external computer running a software that will act as the middleman between the sending and receiving end. Essentially, this software will handle and transmit the messages between the user and the Hexapod.

To summarize, our objectives for this project are the following:

- Design a six-legged robot with multiple degrees of freedom (DoF) using a computer aided design (CAD) software
- Print the whole body of the Hexapod using a 3D Printer
- Design an electrical schematic for the control unit for the Hexapod
- Turn the schematic into a printed circuit board (PCB) that acts as the motherboard of the robot
- Program a microcontroller (MCU) that is found on the motherboard which serves as the control unit of the Hexapod
- Create a smartphone application that communicates with the MCU with which the end-user can control the Hexapod

Construct a network protocol with the help of an external computer that enables the communication between the smartphone application and the MCU.

2. Literature review

As with every existing robot that paved the way for future innovations, six-legged robots already existed in various forms. The first iterations of these robots were heavy and inflexible, but with advancements in computer technology these became much smaller and more agile.

2.1. Existing robots

The first computer controlled hexapod robot, that we know of, was created at the University of Rome in 1972 shown in *Figure 1*. While the robot had electrical drives as actuators, the design of the robot was unfortunately too rigid as it was incapable of operating on uneven ground. Even with these shortcomings, it was a success which paved the way for forthcoming designs.

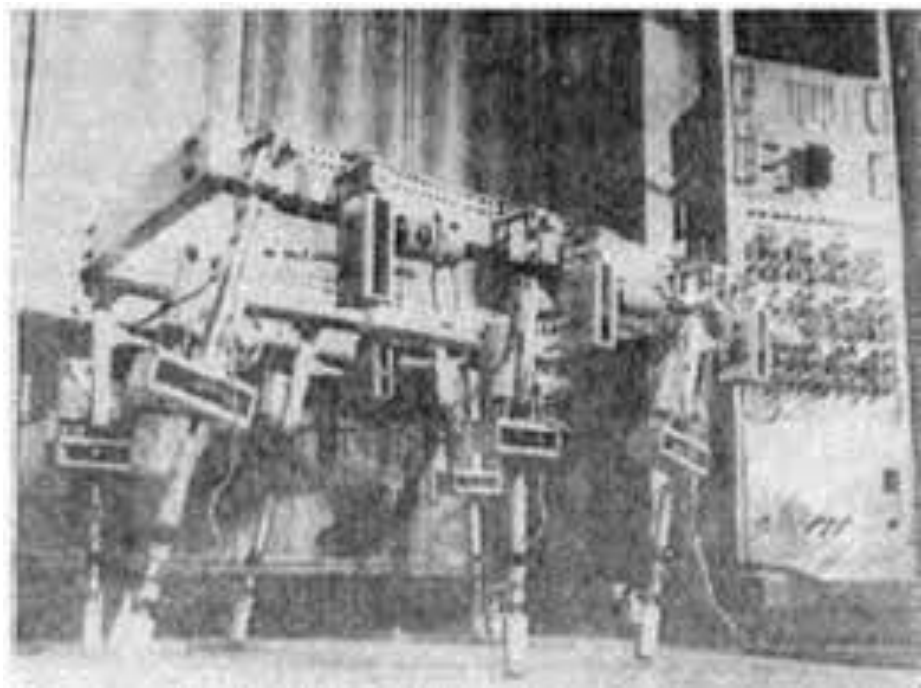


Figure 1: University of Rome's hexapod

(source: hexapodrobot.weebly.com)

LAURON is a hexapod robot that is still under development at the Forschungszentrum Informatik Karlsruhe in Germany. The project started out as research into the locomotion of biologically-inspired insectoids. Mimicking the movements of the stick insect *Carausius Morosus*,

the robot evolved into more than one generation, as the robot currently has 5 iterations. The first one being LAURON and the last LAURON V which is shown in *Figure 2*. Because the robot is intended to be operated from a safe distance in hazardous areas or in other extreme situations, the fifth generation of the robot has a robust mechanical design and multiple DoF which helps to maintain a stable motion even when walking through rough terrain. A camera is mounted on the front of the robot, with this the operator of LAURON can see the local environment and guide it safely.



Figure 2: LAURON and LAURON V

(source: fzi.de)

In terms of design and control, PhantomX Hexapod robot family created by Trossen Robotics is perhaps the most akin to the Hexapod that we've envisioned. Equipped with a Raspberry Pi 4 and DYNAMIXEL smart servos, the PhantomX MK-IV Hexapod (shown in *Figure 3*) is a research Robot Operating System (ROS) platform designed to be a highly versatile, reconfigurable and agile variant of the six-legged robots. With a height of 19cm and width of 61.5cm, this robot is much smaller than the previously discussed hexapods. It can be controlled by either tethered connections or remotely with any commercially available controller through Bluetooth or PlayStation 4 control options.



Figure 3: PhantomX Hexapod MK-IV

(source: trossenrobotics.com)

2.2. Hexapod designs

Noticeably, the most prominent feature of a hexapod robot is its legs. These limb designs come in different shapes and forms, being biologically-inspired or man-made models that are a mixture of synthetic and natural. This classification [1] is shown in *Figure 4*, biologically-inspired robots are trying to replicate the locomotion and gaits of certain animal classes, whereas non zoomorphic robots are artificially created by combining existing technologies with leg shapes found in animals. In mammal configuration the legs are situated under their body which means that the legs need to support the whole-body weight of the robot, but this also gives more stability to the robot. With an arachnid form the legs are found on both sides of the body with their knees poking upwards. Reptilian legs are akin to arachnid legs with the difference being that the knees are lowered to the base of the body. As with non-zoomorphic configurations, the legs can be a combination of the earlier mentioned leg types with man-made technologies e.g., wheels, suspensions or pistons.

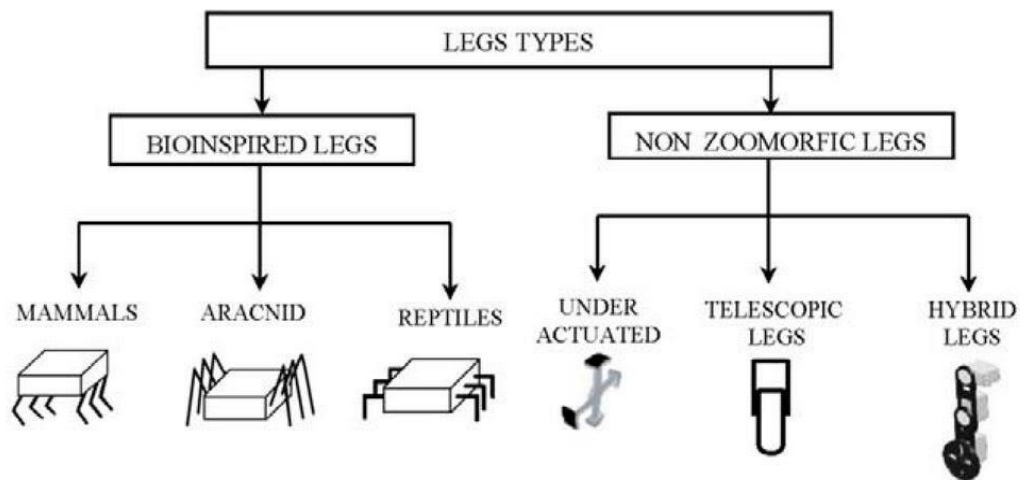


Figure 4: Hexapod leg types [2]

The leg orientation of a hexapod (shown in Figure 5) is another important aspect. In frontal configuration the robot moves perpendicular to the march of its legs, but in sagittal configuration the locomotion is parallel, as with the circular design the robot can move freely and rotate in any direction.

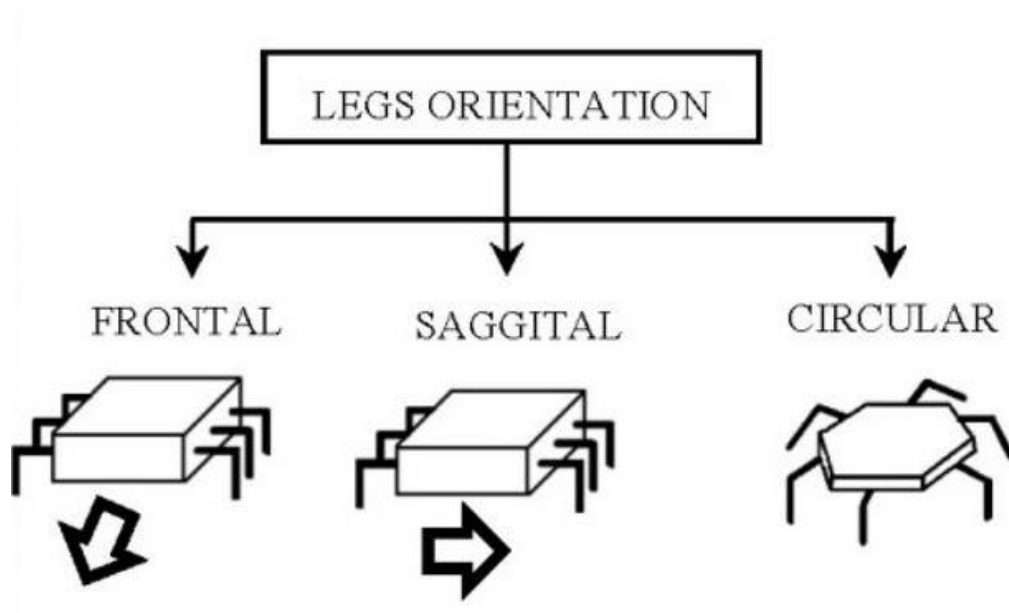


Figure 5: Hexapod leg orientation [2]

The last element to consider in hexapod leg design is the configuration of the joints (see *Figure 6*). In outward knee forms the knees are protruding away from the robot's body, whereas the inward knee forms are poking towards the body, and there can also be a merger of these two forms where one side of the robot has an inward configuration and the opposing side an outward configuration.

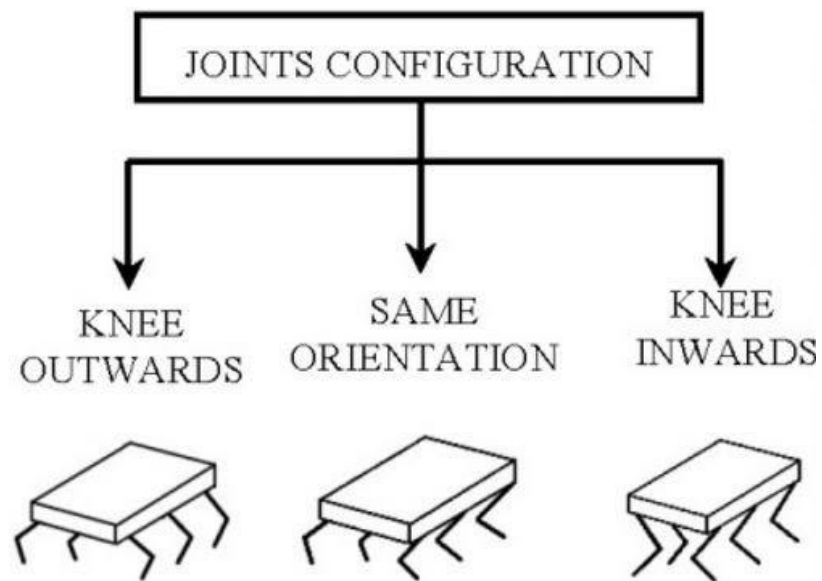


Figure 6: Hexapod joint configuration [2]

3. Architecture

To simplify the development process and design of the Hexapod the architecture seen in *Figure 7* was created. This bigger picture offers an overview of the whole blueprint where 3 main components can be found:

- User
- MQTT IoT Protocol
- Hexapod

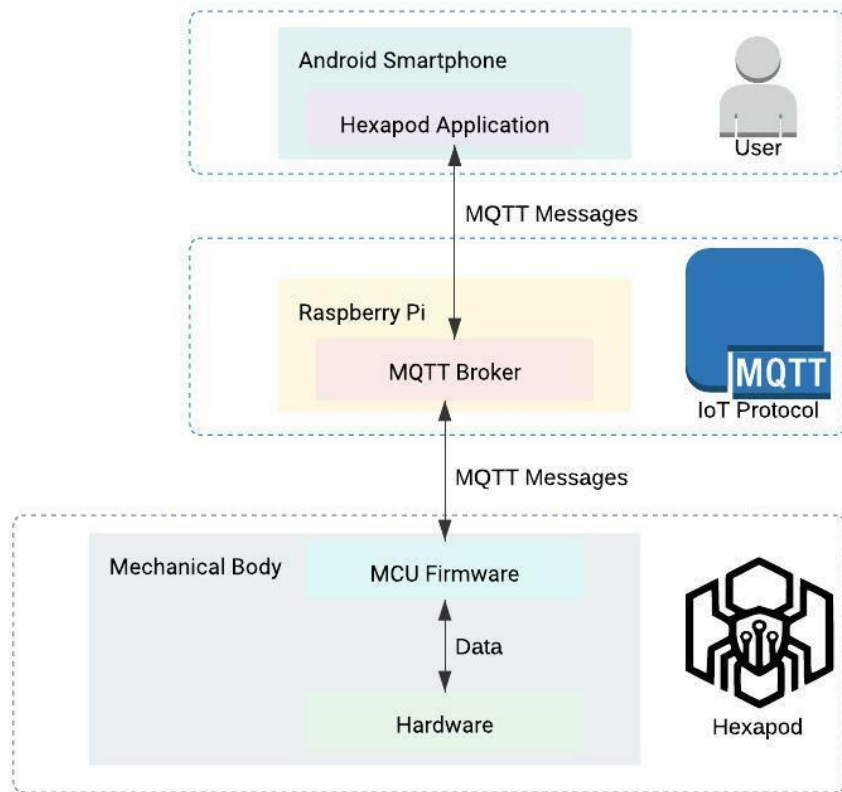


Figure 7: Hexapod architecture

The User is the one who will be controlling the robot through an Android smartphone that runs the Hexapod Control application. With the help of a Raspberry Pi single-board computer that runs a MQTT Broker the Hexapod's MCU can communicate with the Android application, and vice versa, this way the User can control the robot remotely. The last component on the list is the

Hexapod itself, which incorporates all the mechanical, electromechanical and hardware parts of the robot as well as the firmware that will run on the MCU.

3.1. Mechanical design

For the mechanical design of the Hexapod Autodesk's Inventor CAD software will be used. This software will speed up the design process of the body and will prove a great help in visualizing the 3D shapes of the robot.

3.1.1. Physical structure

The mechanical body of the Hexapod will be a combination of biologically-inspired figure and non-zoomorphic composition. To be more precise, an arachnid build with a telescopic leg design configured in an outward knee form.

The base frame will resemble a somewhat distorted rectangle shape with cut corners. Four servomotors will be placed on these ends, and two more in the middle between the ends on the longer side of the frame for a balanced posture.

Each leg of the Hexapod will consist of 3 sections, namely: coxa, femur and tarsus. The servomotors on the base will control the coxa of a leg, on the coxa another servomotor controls the femur, and the final one found on the femur will control the tarsus. Shown in *Figure 8* is the TowerPro MG90S servomotor that the Hexapod will use.



Figure 8: TowerPro MG90S

(source: towerpro.com.tw)

For the arachnid-telescopic leg design, a piston mechanism with a spring that will be placed between the femur and the tarsus of each leg. On the top of the piston casing an infrared (IR) sensor will also be placed, with this the MCU can later measure the distance between the piston and the sensor, thus determining if a leg is touching the ground or not.

The base frame of the Hexapod needs to have a width wide enough to install a PCB support in the centre of it. Below the PCB support needs to be an empty section where a lithium polymer (LiPo) battery pack will be placed, this can be accessed by a lid that will be found on the bottom of the robot's body.

As the motherboard and the servomotors of the robot will be powered by a LiPo battery pack a buck converter is needed. Thus, the body of the robot will also have a back section that will resemble the abdomen/hind body of a spider where the converter will be installed.

A shell can also be mounted on the body as a shielding component and displaying logos for the Hexapod.

3.1.2. CAD design

As stated before, the protruding ends on the base frame with rectangle areas is where the servomotors will be installed. In the centre of the frame is the section reserved for the PCB support and the LiPo battery pack. Slightly raised, at the back end of the base, is where the converter shall be set up (see *Appendix A*).

Moving onto the most vital part of the robot, the hybrid arachnid-telescopic leg of the Hexapod. The piston mechanism, along with the spring attached to it, is fixed between the femur and the tarsus of the six-legged robot. For the piston mechanism to move freely in its casing, the casing is retained on the femur, while the piston is fixed on the tarsus. The two body parts are held together by two joints on each side that will rotate freely when the leg is applying pressure on the ground. These joints serve as a connection between the two body parts and enables the linear movement of the piston.

The shell of the Hexapod is an optional component of the robot that will shield the hardware in case of an accident. A hole is found on the front of the shell where a button can be later installed to powerup/shutdown the Hexapod.

A rendered image of the Hexapod is also shown in *Appendix A*, created with the help of the Studio Environment that is provided by Autodesk's Inventor. This image shows the robot with its shell installed.

3.2. Hardware design

The hardware acts as the middleman between the robot's mechanical, electromechanical and its software components. Since the robot needs to be as compact as possible, the need of a motherboard that incorporates most of the control unit was created. EasyEDA, a web-based EDA tool suite which is tightly connected to the PCB manufacturer JLCPCB and the LCSC electronics part source companies, easing the production of the board, was used to create the schematic and the PCB of the motherboard.

3.2.1. Motherboard schematic

The motherboard of the robot consists of 3 different modules:

- Power Module
- Control Module
- Leg IR Module

The Power Module of the motherboard takes care of the powerup and shutdown sequence of the robot, power distribution to every part of the robot, checks the robot's LiPo battery level and generates the compulsory operating voltage for the Control Module and Leg IR Module (see *Appendix B*).

The supply voltage comes from the external LiPo battery that is connected to the XT90 female connector found on the motherboard's Power Module. The unregulated current goes through an NTD25P03LT4G p-channel power MOSFET, that turns ON when the user pushes the pushbutton found on the robot's body. This initiates the startup sequence of the robot that generates the 5V input voltage for the Control Module with the help of an LM1117 low-dropout linear regulator, while also supplying voltage to the external buck converter and PWM modules.

The Control Module consists of the robot's MCU, three axis accelerometer and gyroscope, and connectors that establishes the connection between the MCU and the external PWM modules, with which we can control the 18 servomotors of the Hexapod.

To reduce the chances of damaging the MCU and to easily access its INPUT/OUTPUT ports, the ESP32 is already mounted on an ESP32-DevkitC V4 development board. An MPU6050 acceleration and gyroscope sensor, which is mounted on a breakout board, is connected to the MCU. With the help of this sensor, we can later create an algorithm that handles the electronic stability control of the Hexapod.

For the Hexapod to know when its legs are pressed against the ground the IR Leg Module was created. This module is made up by a simple analog-to-digital converter that communicates with the MCU, and six connectors to which the external IR transmitter and receiver sensors are connected to.

When a leg of the Hexapod is pressed against the ground the piston will move towards the IR sensor, this creates a voltage difference at the receiver side of the sensor. The MPU6050 converts the raw voltage into a digital signal, which is transmitted to the MCU where the Hexapod can determine if the leg is touching the ground or not.

3.2.2. Additional components

External hardware is required for the robot to function properly, since it cannot be done just by a single motherboard. The battery pack unit that supplies the robot with electrical power is a Gens Ace Soaring 3300mAh 7.4V 30C 2S1P LiPo Battery Pack, as it is shown in *Figure 9*.



Figure 9: LiPo Battery Pack
(source: gensace.de)

The choice fell on this LiPo Battery Pack as our expectations for the Hexapod was that the user should be able to play around with the robot for at least 30-40 minutes. As the laboratory testing of the hardware have shown, the overall hardware should have an active current draw of 5A, and with a power unit that has a 3.3Ah capacity this would be achievable.

Figure 10 shows the XL4016 Buck Converter that will step down the 7.4V of the LiPo battery pack to 5V for the 18 servomotors of the Hexapod.

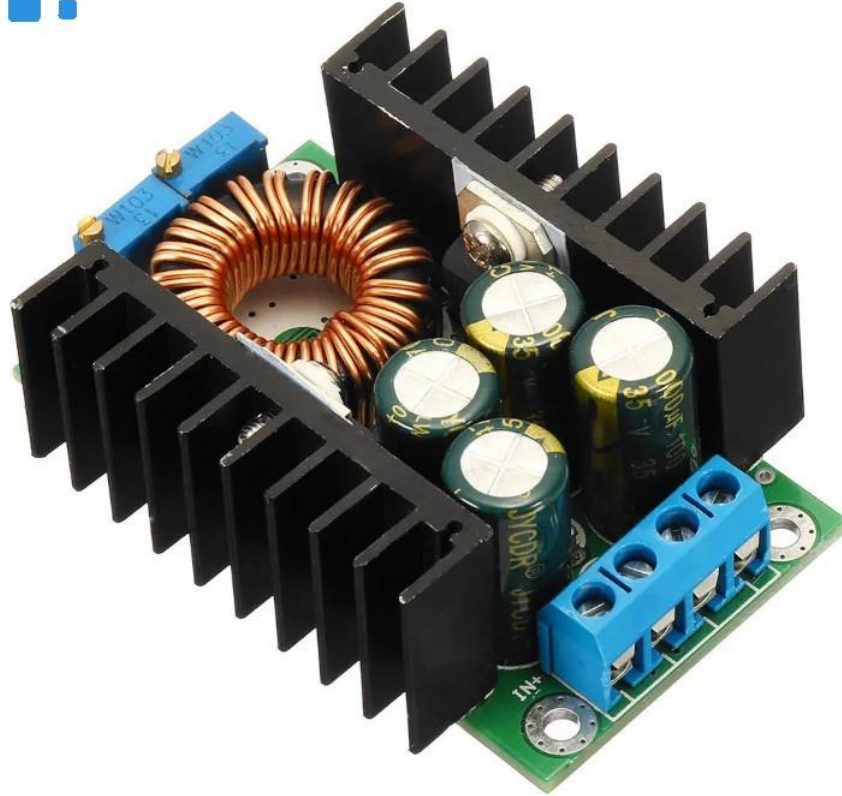


Figure 10: XL4016 Buck Converter
(source: optimusdigital.ro)

This converter [3] is found on the back of the Hexapod and is directly connected to the motherboard's Power Module. It has an output current that ranges from 0.2A to 12A. Since a single servomotor can wind up a stall current of up to 700mA, choosing a converter that can handle these temporary spikes was a must. The converter has multiple built-in features such as: thermal shutdown, output short protection and current limiting functions which will help prevent any damage that might occur during normal operation to the motherboard or servomotors.

Since the MCU cannot handle the generation of 18 PWM signals, the Hexapod uses two external Adafruit PCA9685 16-Channel PWM Driver boards (see Figure 11).

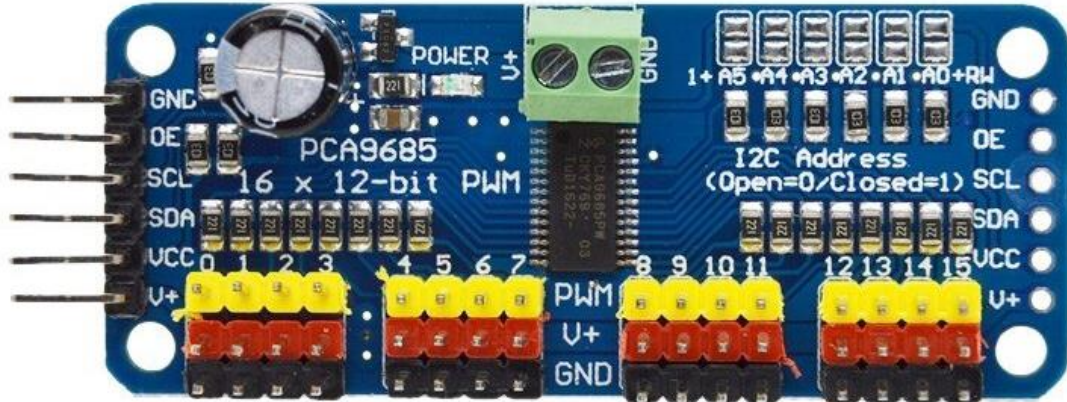


Figure 11: PWM Driver board
(source: learn.adafruit.com)

Considering that the Hexapod uses 18 servomotors for locomotion, we needed 2 boards to drive them all. The PWM Driver boards are physically placed in the centre of the Hexapod, above the motherboard and mirroring each other. The first board will drive 9 servomotors on the left side, with the second one driving the remaining 9 on the right side of the robot.

3.2.3. Communication protocols

The MCU uses two industry standard communication protocols to interface with the hardware components.

I2C, or Inter-Integrated Circuit, is a synchronous serial communication protocol that uses two bidirectional lines, the serial data line (SDA), and the serial clock line (SCL). *Figure 12* shows the simple wiring for this type of interface.

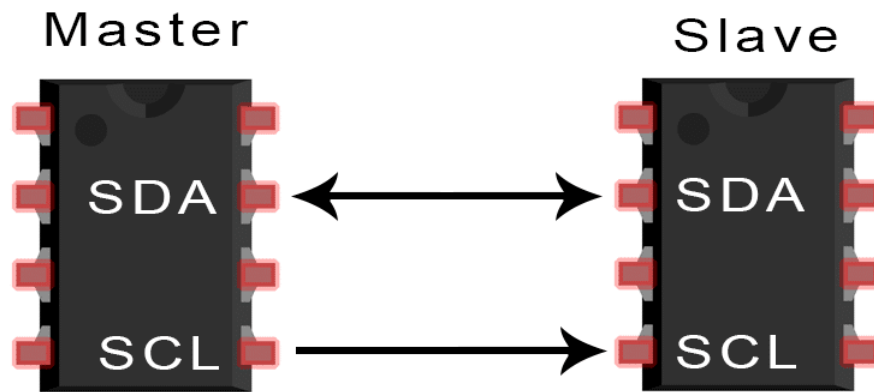


Figure 12: I2C wiring schematic

(source: circuitbasics.com)

With I2C [4], data is transferred in messages. Messages are broken up into frames of data. Each message has an address frame that contains the binary address of the slave, and one or more data frames that contain the data being transmitted. The message also includes start and stop conditions, read/write bits, and ACK/NACK bits between each data frame.

On the motherboard the MCU acts as the Master and the two external PWM Driver boards act as the Slaves. The I2C uses an open-drain/open-collector output which cannot be operated without a pull-up resistor. In our case, to conserve power, the PCA9685 [5] does not incorporate any pull-up/pull-down resistors, instead a pull-down resistor is found outside the IC on the PWM board designed by Adafruit.

SPI, or Serial Peripheral Interface, is a synchronous serial communication protocol that uses four logic-level lines: serial clock (SCLK), master out slave in (MOSI), master in slave out (MISO), chip select/save select (CS/SS). *Figure 13* shows the wiring for the SPI Protocol.

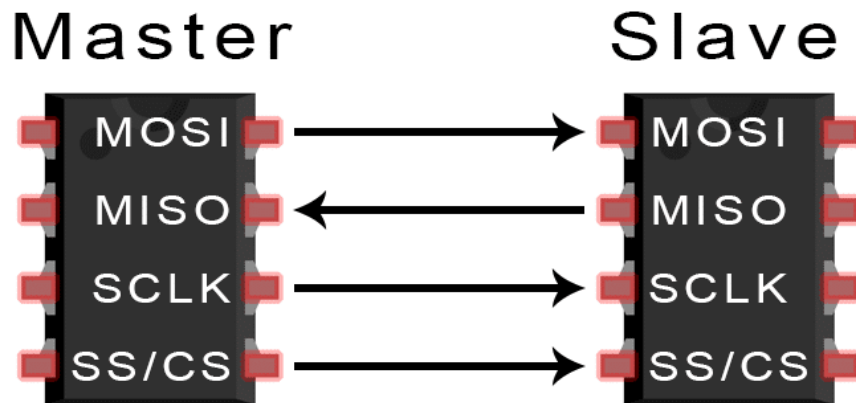


Figure 13: SPI wiring schematic

(source: circuitbasics.com)

The clock signal synchronizes the output of data bits from the master to the sampling of bits by the slave. One bit of data is transferred in each clock cycle, so the speed of data transfer is determined by the frequency of the clock signal. SPI communication [6] is always initiated by the master since the master configures and generates the clock signal, and in case of a single slave the SS line needs to be tied to the ground.

On the motherboard the MCU acts as the Master for the MCP3008 A/D converter.

3.2.4. Communication protocols

As stated earlier, keeping the motherboard compact and small in size was a priority when designing it. With the length of 11.2cm and width of 7.2 cm the PCB fits perfectly in the centre of the Hexapod.

To keep the design of the PCB as clean as possible, the power components are located on the left side and the logic components can be found on the right side of the motherboard (see *Figure 14*).

When designing the PCB, it was important to be mindful of the fact that high current will run through the motherboard because of the servomotors' stall current. Because of this the trace width on the Power Module ranges between 50mil-100mil and with a 1oz thick copper layer at 4 amps would mean a temperature rise of around 10°C which is acceptable as it would not result in

a significant voltage drop¹. The signal traces near the MCU and components such as the MPU6050 are kept at 20mil as these will not experience any high currents running through.

For debugging and other testing purposes, male pin headers were placed near the MCU, while other connectors such as those that will be connected to the IR sensors found on the Hexapod's legs are placed on the both sides of the motherboard to be easily accessed when wiring together every part of the robot.

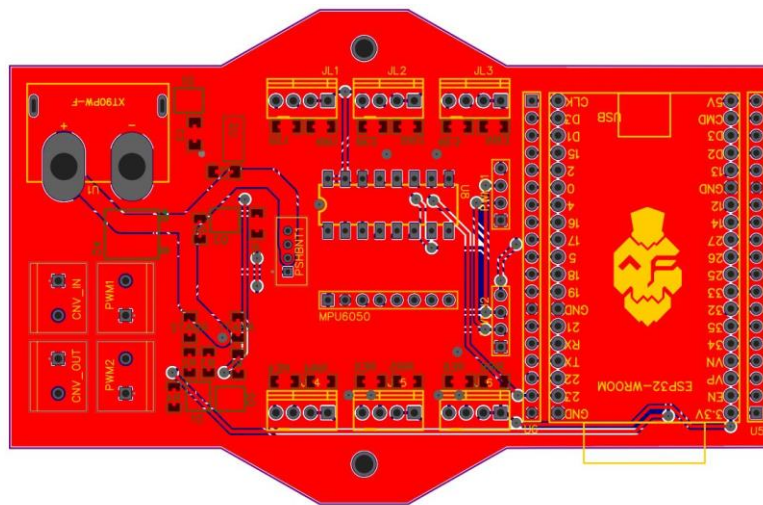


Figure 14: Motherboard PCB

Keeping in mind that the prototype of the motherboard will be created in the laboratory of a university the PCB was designed be double-sided. This way the SMD components will be mounted on the bottom side and the THT components will installed from the top side of the PCB as shown in *Figure 15*.

¹ PCB Trace Width Calculator - <https://www.melpcb.com/blog/pcb-trace-width-vs-current-table/>

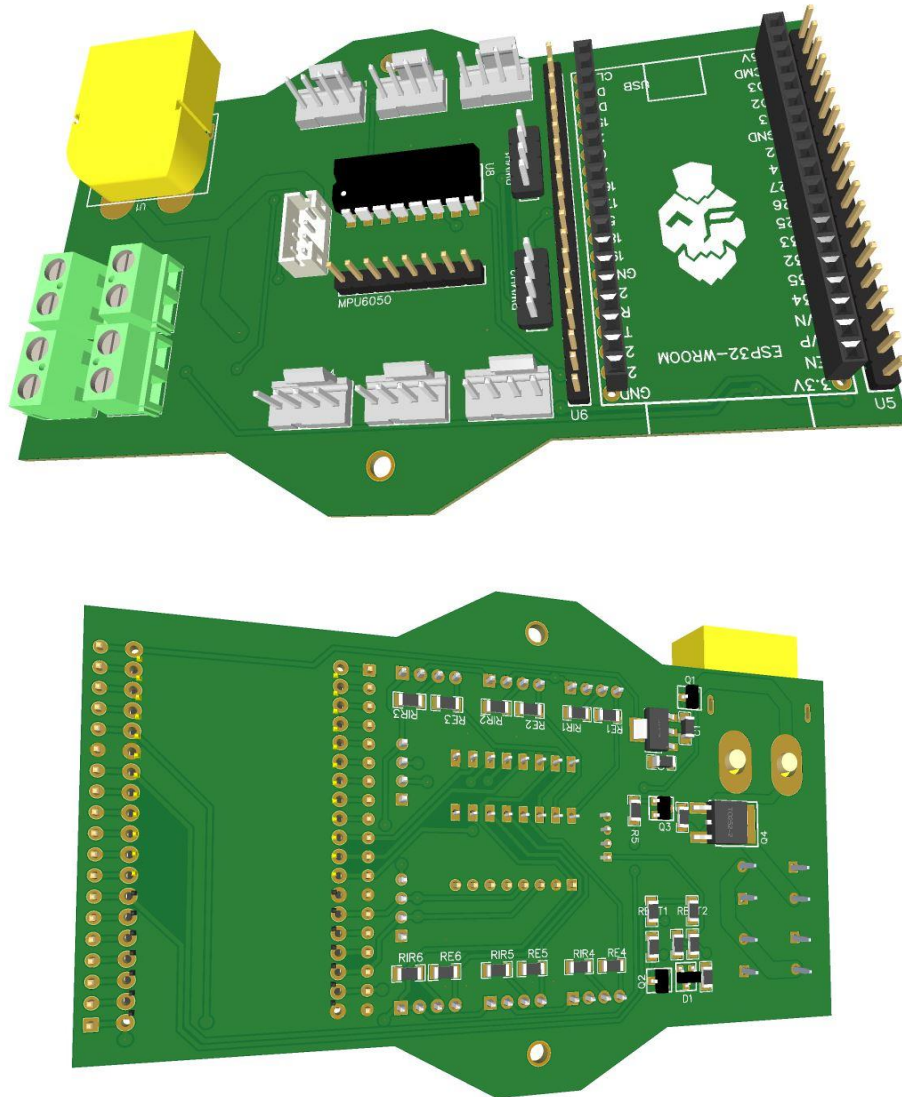


Figure 15: PCB 3D model top and bottom side view

With everything accounted for, the Gerber files [7] were generated by EasyEDA and ready to be used. The files were then sent to a PCB manufacturer in order to be turned into the designed motherboard.

3.3. Kinematics and gait analysis

We can describe the legs of the Hexapod as robotic manipulators, and since they are treated as manipulators, we can also draw a kinematic diagram of a leg. The servomotors and the piston

mechanisms, with the spring attachments, of the robot can be treated as active joints. These two types of joints are revolute and prismatic joints.

Shown in *Figure 16* is the kinematic diagram of a leg. The configuration of a leg is 3R1P, since this manipulator has 4 wrists described by the circular and translational movements: θ_1 , θ_2 , θ_3 and d . These wrists also have their limits, $\theta_1:(-45^\circ, +45^\circ)$, $\theta_2:(-30^\circ, +45^\circ)$, $\theta_3:(-45^\circ, +45^\circ)$ and $d: [0, 10mm]$. Knowing the movements of these wrists, and the length of the links between the wrists (a_1, a_2, a_3, a_4), we can calculate where the endpoint (P) of the leg is situated.

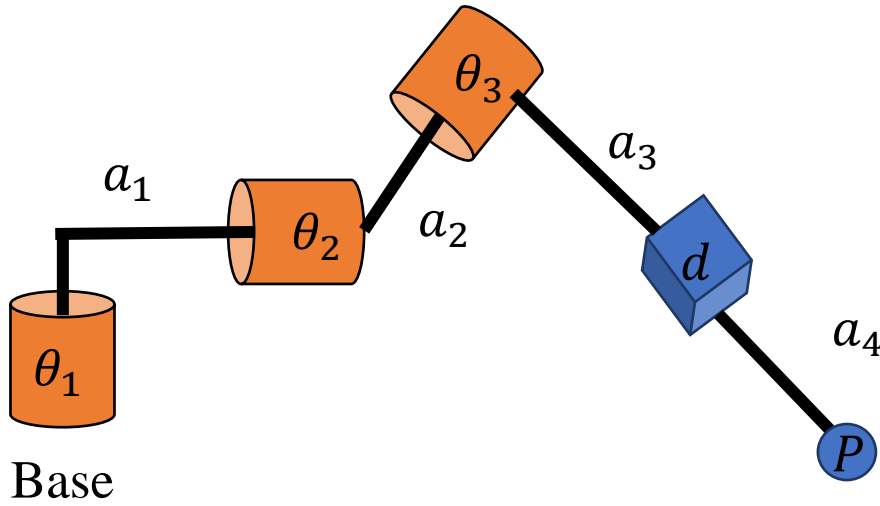


Figure 16: Kinematic diagram of a leg

3.3.1. Forward kinematics

For the leg to move in any direction the position of the endpoint needs to be known. Because of this, the Denavit-Hartenberg table [8] (shown at *Table 1*) was created. The above-mentioned table describes the transition from the base's coordinate system to the first joint's coordinate system and onto the next until the endpoint is reached. With this approach we can calculate the movement of each joint in order to move the endpoint to its desired position.

Links	θ_i	α_i	r_i	d_i
$0 \rightarrow 1$	θ_1	90	r_1	a_1
$1 \rightarrow 2$	θ_2	180	r_2	a_2
$2 \rightarrow 3$	θ_3	-90	r_3	a_3
$3 \rightarrow 4$	θ_4	0	0	$d + a_1$

Table 1: Denavit-Hartenberg table for Hexapod leg

As seen in above, there are a total of 4 system transformations that needs to be performed to reach the endpoint of a leg. These linear transformations [15] can be described in terms of transformation matrixes. Respecting the Denavit-Hartenberg convention, *Eq. 1* describes every transformation matrix, which will be the dot product of a translational matrix and rotation matrix along the z axis multiplied the translational matrix and rotation matrix along the x axis.

$$A_i = Trans_z(d_i) \cdot Rot_z(\theta_i) \cdot Trans_x(r_i) \cdot Rot_x(\alpha_i) \quad Eq. 1$$

The transformation matrices found in *Eq. 2* describe the transition from one joint's frame of refence to the next one.

$$\begin{aligned}
A_1 &= \begin{bmatrix} c\theta_1 & 0 & s\theta_1 & r_1s\theta_1 \\ s\theta_1 & 0 & -c\theta_1 & r_1c\theta_1 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
A_2 &= \begin{bmatrix} c\theta_2 & s\theta_2 & 0 & r_2c\theta_2 \\ s\theta_2 & -c\theta_2 & 0 & r_2s\theta_2 \\ 0 & 0 & -1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
A_3 &= \begin{bmatrix} c\theta_3 & 0 & -s\theta_3 & r_3c\theta_3 \\ s\theta_3 & 0 & c\theta_3 & r_3s\theta_3 \\ 0 & -1 & 0 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \quad Eq. 2$$

$$A_4 = \begin{bmatrix} c\theta_4 & 0 & s\theta_4 & 0 \\ s\theta_4 & 0 & -c\theta_4 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d + d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Because of the physical placement of the servomotor on the tarsus, the A_4 rotation matrix is needed to align the reference frame from the femur's servomotor to the one that is found on the tarsus. Consequently, the A_4 and A_5 matrices can be multiplied to form a simpler matrix shown in *Eq. 3*.

$$A_{4^*} = \begin{bmatrix} 1 & 0 & 0 & d \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{Eq. 3}$$

Thus, the transformation matrix from the leg's basepoint (0) to the endpoint (4) can be calculated with the matrix given in *Eq. 4*.

$$T_4^0 = A_1.A_2.A_3.A_{4^*} \quad \text{Eq. 4}$$

3.3.2. Inverse kinematics

With inverse kinematics we can calculate the angles of the joints at which they need to be in order for the leg to reach a specific endpoint position. This method is extremely useful, considering that for a leg to follow a path, the joints need to rotate accordingly in order for it to describe a given motion.

Eq. 5 represents the 3 distances along the x (r_1), y (r_2) and z (r_3) axis from the base of a leg to the endpoint.

$$\begin{aligned} r_1 &= \sqrt{x^2 + y^2} - a_1 \\ r_2 &= z \\ r_3 &= \sqrt{r_1^2 + r_2^2} \end{aligned} \tag{Eq. 5}$$

The 3 support angles found in *Eq. 6* are needed to calculate the joint angles of the femur and tarsus.

$$\begin{aligned} \varphi_1 &= \tan^{-1}\left(\frac{r_1}{r_2}\right) \\ \varphi_2 &= \cos^{-1}\left(\frac{a_3^2 - a_2^2 - r_3^2}{-2a_2r_3}\right) \\ \varphi_3 &= \cos^{-1}\left(\frac{r_3^2 - a_2^2 - a_3^2}{-2a_2a_3}\right) \end{aligned} \tag{Eq. 6}$$

Finally, *Eq. 7* describes the angles of the three revolute joints of a leg.

$$\begin{aligned} \theta_1 &= \tan^{-1}\left(\frac{y_1}{x_2}\right) \\ \theta_2 &= \varphi_1 + \varphi_2 \\ \theta_3 &= 180^\circ - \varphi_3 \end{aligned} \tag{Eq. 7}$$

3.3.3. Proposed gaits

Having a method with which we can calculate the position of a leg's endpoint, we can move on to describe the actual movement of the Hexapod. As the Hexapod is a biologically-inspired robot, the locomotion will be engineered by a set of gaits that will can be found in insects and other biological systems, namely the Metachronal gait, also known as wave gait, and the Tripod gait.

For the sake of simplicity, in the next tables the legs of the Hexapod will be coded as R_1 , R_2 , R_3 , L_1 , L_2 , L_3 , where the notations represent the right and left legs of the robot.

The Metachronal gait [9] (as Shown in *Table 2*) serves as a primitive instance of locomotion in biological systems. In this pattern of limb movement, the legs of a creature move in a certain direction with each leg given a specific timeframe for it to be positioned in that direction. The gait starts off on either the right or the left side of the body and the front leg taking the first step. The next step is taken by the middle leg on the same side. This is repeated in a circular pattern until the first leg on the opposing side has moved, when this happens a gait cycle is completed and the robot has moved in that certain direction.

R_1						
R_2						
R_3						
L_1						
L_2						
L_3						

Table 2: Metachronal gait leg cycles

For the Tripod gait **Error! Reference source not found.** (shown in *Table 3*) is a more complex form of locomotion, where the legs of the system need to be synchronized in order to move in a certain direction. Just as the Metachronal gait, the gait can start off on either side, with the first and last leg of one side and the middle leg on the opposite side taking the first step. The next step is taken by the middle leg from the same side and the first and the last leg on the opposite side. With every leg now moved in the same direction a gait cycle is considered completed.

R_1						
R_2						
R_3						
L_1						
L_2						
L_3						

Table 3: Tripod gait leg cycles

As seen in the Tables above, the Metachronal gait is 3 times slower than the Tripod gait, meaning the locomotion of the Hexapod is the fastest when the latter pattern of movement is selected, but it also uses more energy to move the system. There are other gait types (see [11]), but those are seldom seen in insect motion.

3.3.4. Path generation for gait implementation

Implementing the gaits for the legs of the Hexapod requires a path which the endpoint of a leg is able to follow. Because of the shape of the legs, the path that the endpoint needs to follow in order to for the Hexapod to move around is nothing else than a semicircle in 3D space.

To verify the equations for the robot's kinematics and to test the gaits discussed above, a MATLAB simulation was created to evaluate the joint movements of a leg with the actual lengths and their joint constraints.

Figure 17 shows the trajectory of the endpoint of a single leg for the vertical movement of the Hexapod. Coloured in light-blue is the path that was generated with the equations describing a semicircle in 3D space, along with the paths that each joint of the leg followed to enable the endpoint to reach its target. The coordinate systems units are given in mm.

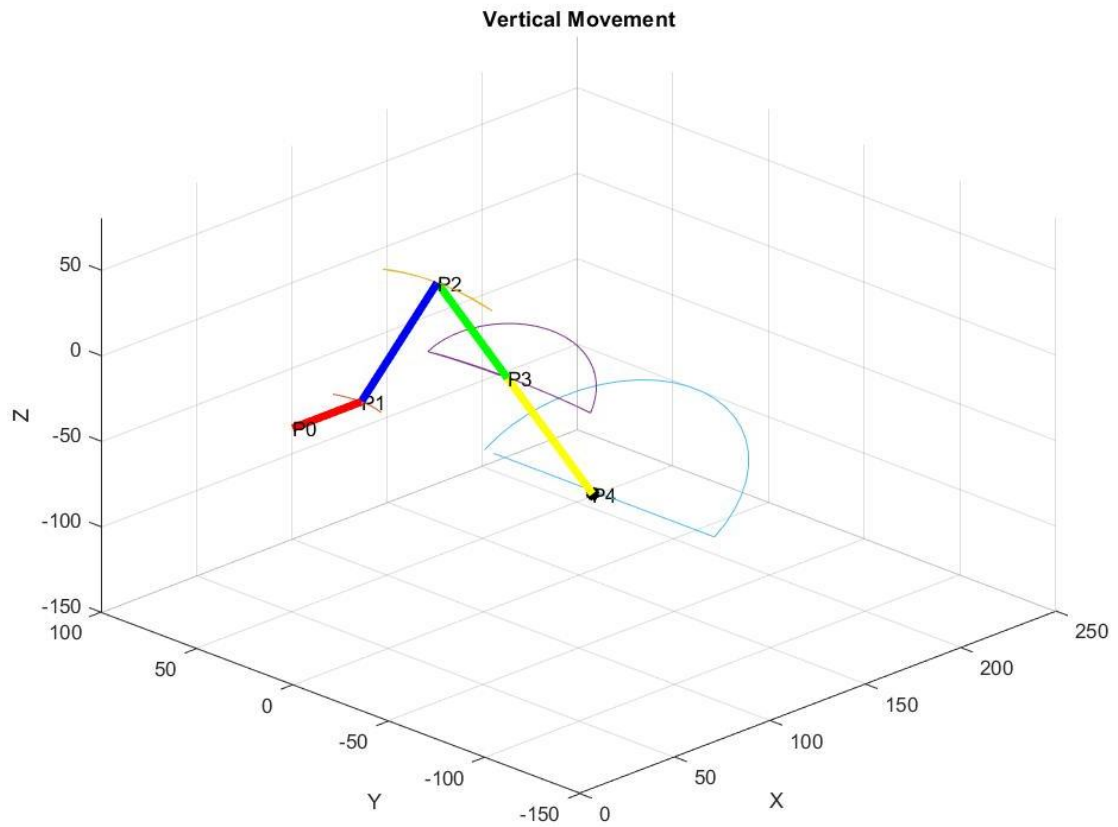


Figure 17: Vertical movement

By rotating around its axes and offsetting the centre of the semicircle it can also describe a horizontal movement of the Hexapod, this is shown in *Figure 18*.

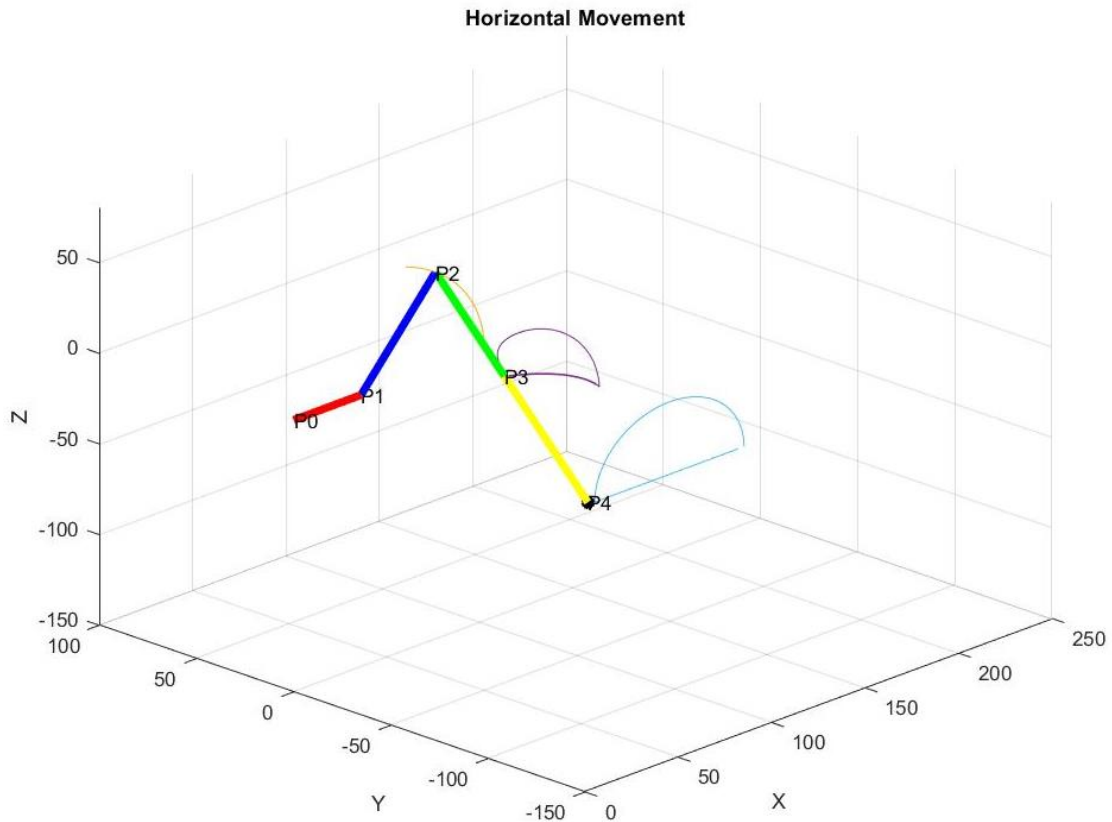


Figure 18: Horizontal movement

This method can also be used to manoeuvre the Hexapod in a direction that is slanted/oblique to the line of attack.

If this method were to be implemented in the firmware of the MCU to control the robot, the obvious downside is that the Hexapod can only advance in three distinct lines of attack, namely: vertically, horizontally or diagonally. By dynamically modifying the path with the control inputs from the user's interface, any leg can change its direction automatically which would result in an omnidirectional movement.

Figure 19 shows a possible solution for this problem. With the help of MATLAB's GUIDE² a simple user-interface was created to help visualize what an implementation of a joystick in the Android application would look like. This joystick would be used by the user to control the direction of the robot.

² <https://www.mathworks.com/help/matlab/ref/guide.html#d123e582394>

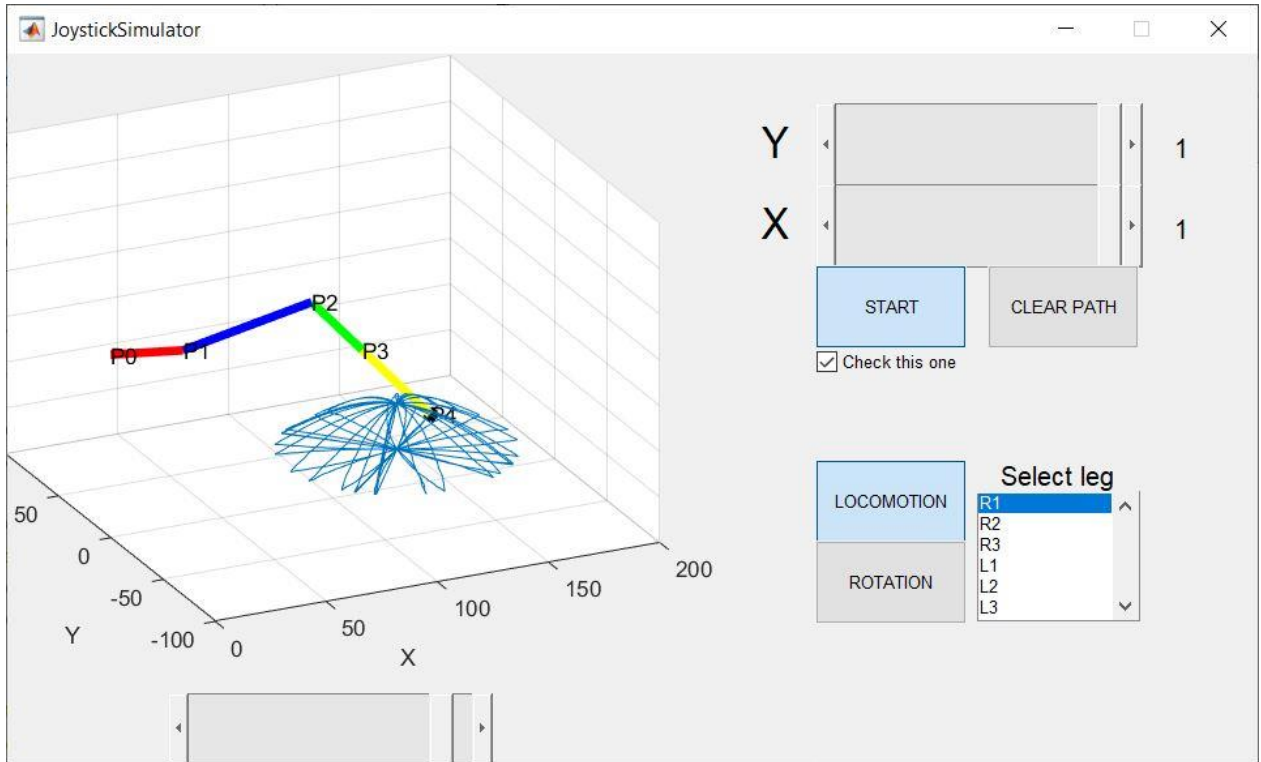


Figure 19: Omnidirectional movement

Here changing the sliders labelled with “X” and “Y” is simulating the movement of the joystick and calculating the angle given by the coordinates in Eq. 7 to determine the angle by which the semicircle is rotated around the Z axis. With the path now dynamically generated whenever a new user input is detected, the robot can calculate the angle of the servomotors with inverse kinematics which results in the amalgam of paths coloured in blue seen above.

This GUI can also be used for the simulation of certain rotations which the Hexapod is capable to do.

3.4. Software design

The software is an indispensable part of the Hexapod. Seeing that it can be found on every component of its architecture, this high-level element of the Hexapod acts as the firmware running on the MCU of the motherboard, handles the communication between the user and the robot on an external computer, and provides an interface on the smartphone of the user with which the robot can be controlled.

3.4.1. Software requirements

User requirements

- The user needs to ensure that the Hexapod is in the immediate proximity of its MQTT network and has stable ethernet connection
- The user needs to make sure that the Hexapod's LiPo battery is at an optimal level so that it can function properly
- For the Android application, the user needs to make sure that its smartphone is on the same network as the Hexapod

System requirements

- ESP32 microcontroller
- Raspberry Pi 3 Model B
- Android Smartphone with at least Android version 11

Functional requirements

- The user should be able to power up the Hexapod simply by long pressing the button found on the front of body
- Likewise, the user should be able to shot down the Hexapod simply by long pressing the button found on the front of body
- If there are any errors regarding the connection status of the robot or the Android application the user should be notified via a popup dialog window
- The user should be notified if the Hexapod's battery level drops below a safe shutdown level
- If the Hexapod's battery level drops below the safe operating level, then the robot should engage into its shutdown sequence to prevent damaging the battery

Non-functional requirements

- Latency between the Android application and robot should be kept at minimum when the user is controlling the robot

- It should not matter if the robot is powered up first or the application as the MQTT broker can retain messages and forward it to its subscribers
- If the user abruptly closes the application without a proper shutdown sequence of the robot, the robot should stay idle until the user reconnects or shuts it down by the frontal button

3.4.2. MCU firmware

The ESP32 runs MicroPython, which is a full Python compiler and runtime interpreter that runs on the bare-metal. MicroPython [12] is a lean and efficient implementation of the Python 3 programming language that includes a small subset of the Python standard libraries which are optimized to run on microcontrollers and in constrained environments.

Flashing the MicroPython compiler and the Python project files on to the MCU is done using esptool³, a Python-based, open-source, platform-independent utility to communicate with the ROM bootloader in Espressif chips. This is done by establishing a serial connection between the computer (typically a COM port on a OS running Windows) and the MCU with the help of a USB cable.

Figure 3.13 shows the Python files that are found on the flash memory of the ES32 with respect to the relationship between them.

Ensuing the startup flow – the first and second stage bootloader – the compiler firstly looks for the boot.py file and runs it. This file imports the required standard libraries installed with the MicroPython compiler which are needed by the application to establishes wireless network connection with the configuration parameters specified inside the code. If no errors occurred during this stage the compiler runs the main program.

The main.py Python file handles the looping of the control algorithm that runs on the MCU after a successful startup sequence. This Python file firstly imports the umqtt.py, this library is a MicroPython implementation with which the MCU can establish connection to the MQTT broker and then publish/subscribe messages.

³ github.com/espressif/esptool

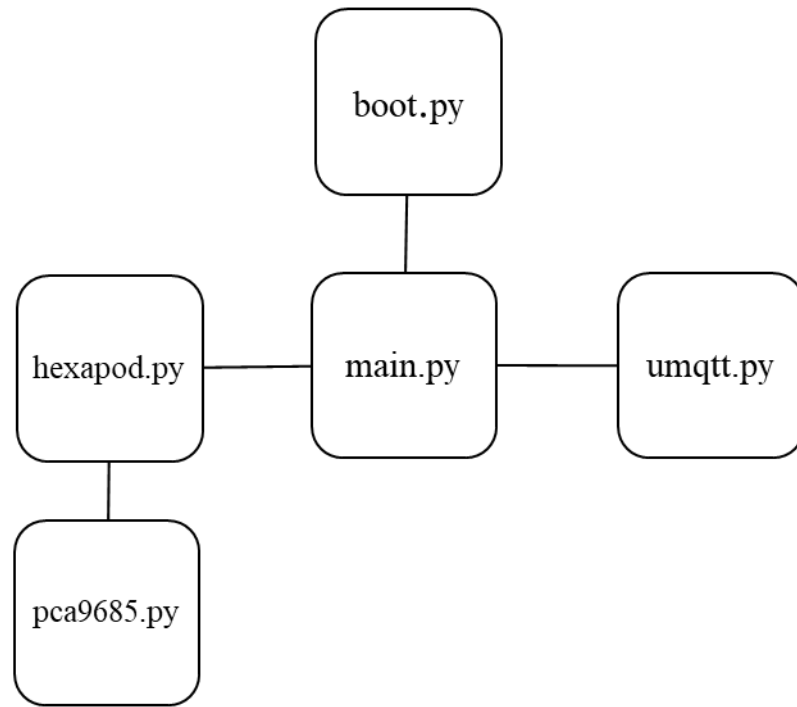


Figure 20: MCU software architecture

After a successful network establishment to the MQTT broker, the main program creates a Hexapod object from the class with the same name found inside the hexapod.py library. Inside this Python file is where the control algorithm of the robot is found which takes care of the locomotion, rotations around the principal axes and other auxiliary functions. The hexapod.py library also makes use of the pca9685.py library with which the MCU establishes I2C communication with the PWM driver board and with which the MCU can read/write to the registers found on the board.

3.4.3. MQTT network

For the wireless communication between the MCU and the Android application to be accomplished a MQTT network protocol is created. MQTT [13] is an OASIS standard messaging protocol for the Internet of Things (IoT). It is designed as an extremely lightweight publish/subscribe messaging transport that is ideal for connecting remote devices with a small code footprint and minimal network bandwidth.

In this configuration, the MCU and the Android application act as clients that subscribe/publish messages to the broker. The broker is a piece of software that runs on a remote server, its job is to receive the published messages, save these messages to their respective topics and then transmit

the messages to the clients that were subscribed to these topics. In the case of the Hexapod's architecture, the MQTT broker runs on a Raspberry PI single-board computer. The MQTT network architecture for this project is shown in *Figure 3.14*.

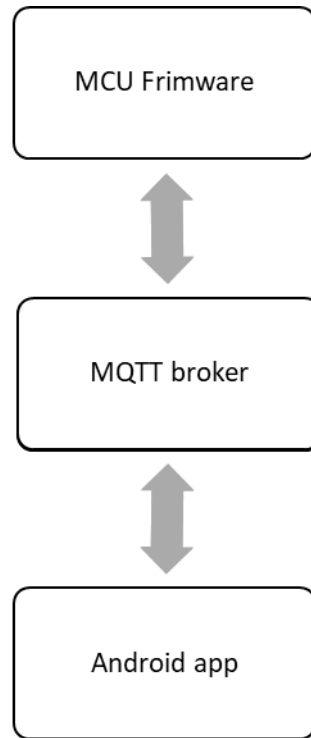


Figure 21: MQTT network

3.4.4. Android application

The Android application is created with Android Studio [14], an integrated development environment developed by Google.

Opening the Hexapod Controller app, the user is greeted by a 3 seconds long splash screen showcasing the Hexapod logo with some animation along with it. Following this, the user is taken to the menu of the application, shown in *Figure 3.15*.



Figure 22: Menu activity

Here the user is presented with two buttons, “Controller” and “Overview”. If the user touches the “Overview” button it will open an activity where a brief description of the project is found. Otherwise, touching the “Controller” button opens up the actually controller activity. Here the app awaits a READY MQTT message from the robot before the user can do anything, having the actions of the user blocked by an animated dialog showing the awaiting of the message. When the message is received, the pup-up dialog disappears and the user now has full authority over the controller of the robot. The “Controller” activity is shown in *Figure 3.16*.



Figure 23: Controller activity

Because the controller might be a bit confusing at first, touching the “?” found at the upper right corner of the controller, opens up a dialog (shown in *Figure 3.17*) where a detailed description of the controller’s functionality and the robot’s sensors such as the displayed gyro and speedometer are explained. The user can cycle through all of the controllers features by pressing the left or right arrows found besides the OK button.



Figure 24: Description dialog

4. Conclusions

During control of the Hexapod, two types of gait implementations can be realized. The first option is to generate all of the angles for horizontal, vertical and diagonal movement with a MATLAB script and then from a Python list let the firmware transmit the angle values to the PWM driver for the servomotors to move into those angles. The obvious drawback of this option is that the robot will be limited to just these three types of movements, this will result in a movement that will be a bit clunky as the desired position of the Hexapod will not necessarily be achieved. The second option is to implement the inverse kinematic algorithm and let the firmware calculate each angle accordingly to the input from the Android application's controller. This will result in a more smooth and agile response of the legs, as the change of the direction is dynamic. Compared to the first option this seems the better solution, but we need to consider the fact that the MCU needs to calculate the inverse kinematic model of each leg every time as long as the user displaced the controller's joystick from its null position. This disadvantage can be seen during operation, if the paths of the gaits' resolution is too high the movement will become slower, but decreasing the resolution will speed up the speed of the movements as fewer calculations will be needed.

While testing the motherboard of the Hexapod we have come across a startup sequence problem, which resulted from the MCU unable to boot when powering up the robot from an external power supply. As it turns out, this was because the EN line needs to be HIGH in order for the MCU to boot. The simplest solution to this problem was to add a 10-100 μ F capacitor between the ESP32's EN and GND pin, this way the capacitor will bring the line HIGH for a short amount of time before discharging and normal operation can continue. For the next version of the PCB this capacitor will be added into the design.

Because the MQTT network is orchestrated by a Raspberry Pi computer, the area of effect where the user can control the robot is limited by the reach of the wireless router's signal strength. The MCU also need internet connectivity, if this is not provided the whole system falls apart. Keeping in mind that this is just a prototype, and that we have chosen the MQTT network solution because is fast, lightweight and quick for these kinds of purposes, the communication network of the Hexapod can be changed in a future version. For now, this solution will suffice as the project is only a demo.

With this proof of concept, we have demonstrated that the current model is a viable solution for a working six-legged robot. No further changes are needed to be made to the CAD model, as

by changing the current plastic gear train servomotors with their metal gear train counterparts, the legs will uphold the weight of the robot.

As the current standing of the aforementioned and realized prototype, the Hexapod will be displayed on a platform and will be controlled from the Android application's controller. This way, the legs of the robot will not be touching the ground and a robot control demonstration can be made.

5. Future development possibilities

To make this concept a reality the first step would be to replace the MG90S servomotors for a more durable and stronger option that would support ground operation. If the design of the robot can be considered adequate one option is to install Savox SH0255MG servomotors into the Hexapod as their dimensions are the same as their MG90S counterparts. These servomotors come with a number of advantages over the MG90S, such as aluminum casing and metal gear train for durability and longevity, higher torque and rotational speed at the same supply voltage. These servomotors are digital servos, meaning the control algorithm is needed to be readjusted as the control signal frequency is 200-250Hz, in contrast to their analog counterparts where the frequency is between 50Hz and 60Hz.

As this robot is only a prototype it will serve well as the foundation for the next version of the Hexapod. For the next iteration of the Hexapod my goal is to redesign most of the mechanical structure of the robot to make it more compact and robust. This way space will be freed up to accommodate slightly larger and stronger servomotors to make the Hexapod faster and more agile. It will also feature a new motherboard with more computing power to facilitate a camera and LiDAR sensors to help the operator map the robot's surroundings. The idea is to design a robot that can explore hazardous environments, where it can be either teleoperated from a safe distance or be instructed to perform tasks autonomously.

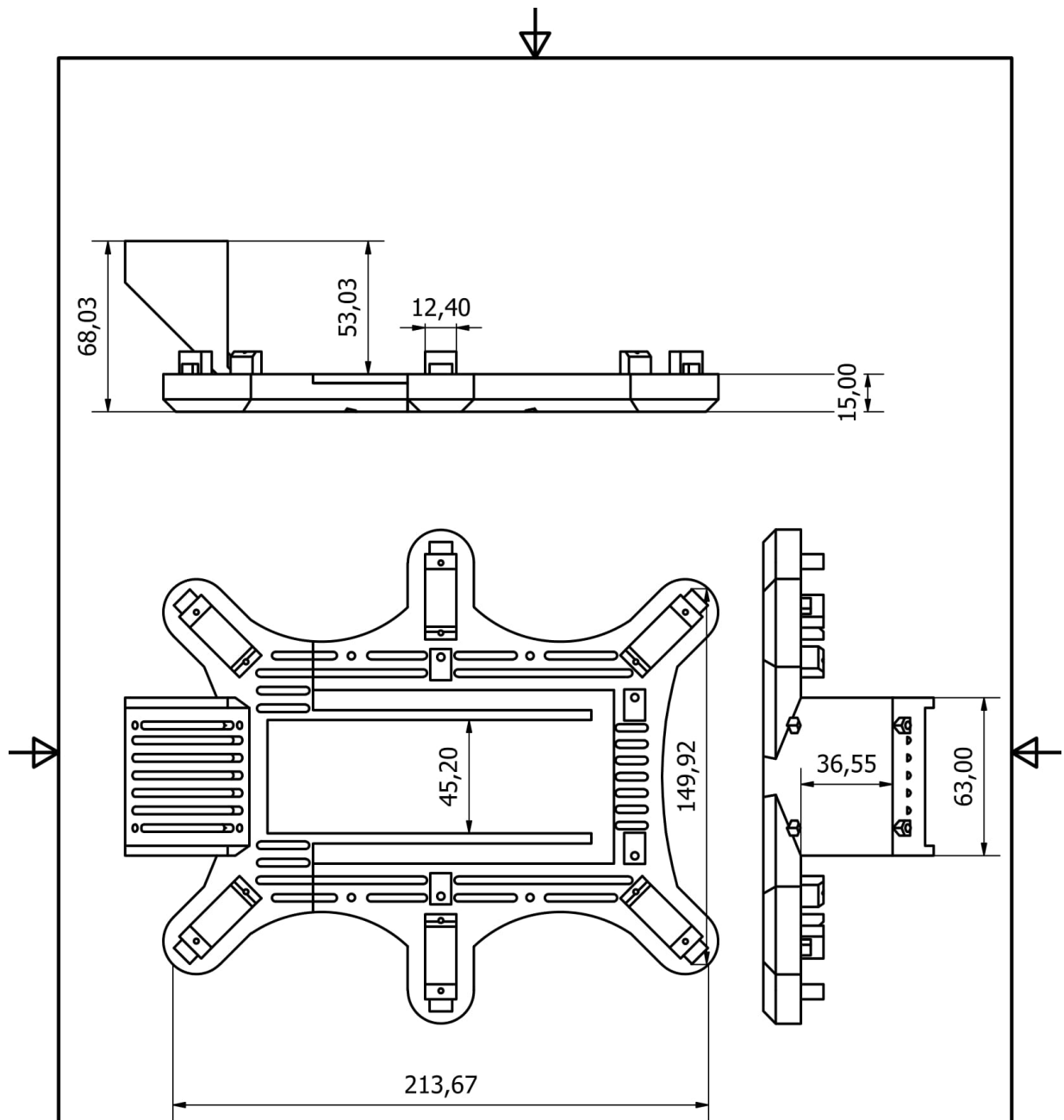
6. Bibliography



- [1] The ATHELETE Rover n.d.,
<https://www.nasa.gov/exploration/dio/athlete_promo.html>
- [2] Tedeschi, Franco, and Giuseppe Carbone. 2014. "Design Issues for Hexapod Walking Robots" Robotics 3, no. 2: 181-206. <https://doi.org/10.3390/robotics3020181>
- [3] 12A 180KHz 40V Buck DC to DC Converter n.d.,
<<http://www.xlsemi.com/datasheet/xl4016%20datasheet.pdf>>
- [4] Basics of the I2C Communication Protocol n.d., <<https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol>>
- [5] Adafruit PCA9685 Schematic n.d, <<https://learn.adafruit.com/assets/36269>>
- [6] Basics of the SPI Communication Protocol n.d., <https://www.circuitbasics.com/basics-of-the-spi-communication-protocol>
- [7] Williams, A., 2004. "Build your own printed circuit board. " New York: McGraw-Hill.
- [8] Denavit, J., & Hartenberg, R. (1955). "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices". Journal Of Applied Mechanics, 22(2), 215-221. doi: 10.1115/1.4011045
- [9] Fujiki, S., Aoi, S., Funato, T., Tomita, N., Senda, K., & Tsuchiya, K. (2013). "Hysteresis in the metachronal-tripod gait transition of insects: a modeling study". Physical review. E, Statistical, nonlinear, and soft matter physics, 88 1, 012717
- [10] Types of Robot Gait n.d., <<https://hexapodrobots.weebly.com/types-of-robot-gait.html>>
- [11] H. Tollervey, N. (2017). *Programming with MicroPython: Embedded Programming with Microcontrollers and Python* (1st ed.). O'Reilly Media.
- [12] MQTT: The Standard for IoT Messaging n.d., <<https://mqtt.org>>
- [13] Horton, J. (2021). *Android Programming for Beginners* (3rd ed.). Packt Publishing Ltd.
- [14] SH0255MG Micro Digital Servo n.d.,
<https://www.savoxusa.com/products/savsh0255mg-micro-digital-mg-servo-13-54>
- [15] Gentle, James E. (2007). "Matrix Transformations and Factorizations". Matrix Algebra: Theory, Computations, and Applications in Statistics. Springer.

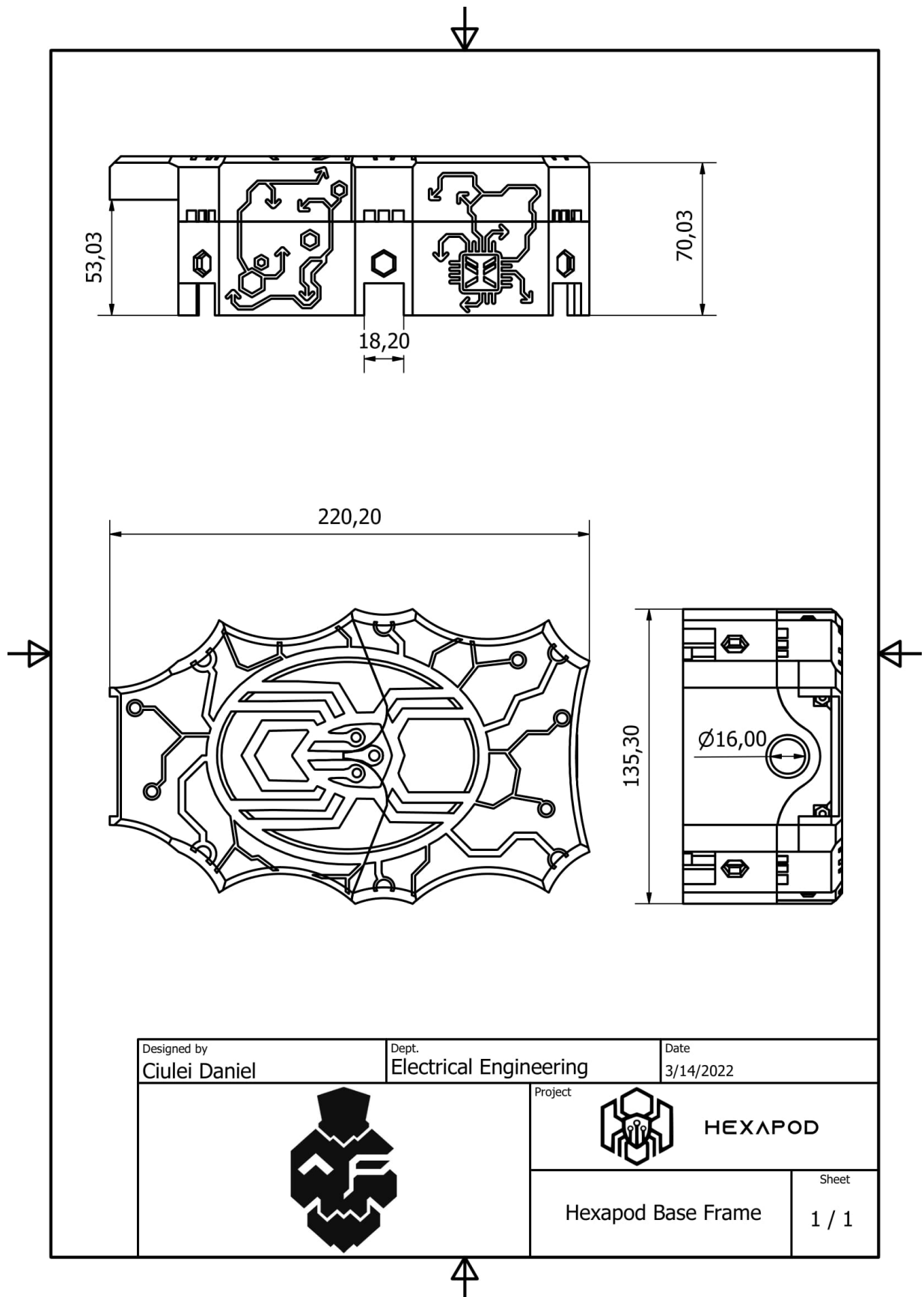
7. Appendices

7.1. Appendix A – Mechanical design

All measurements are given in mm, unless otherwise noted.

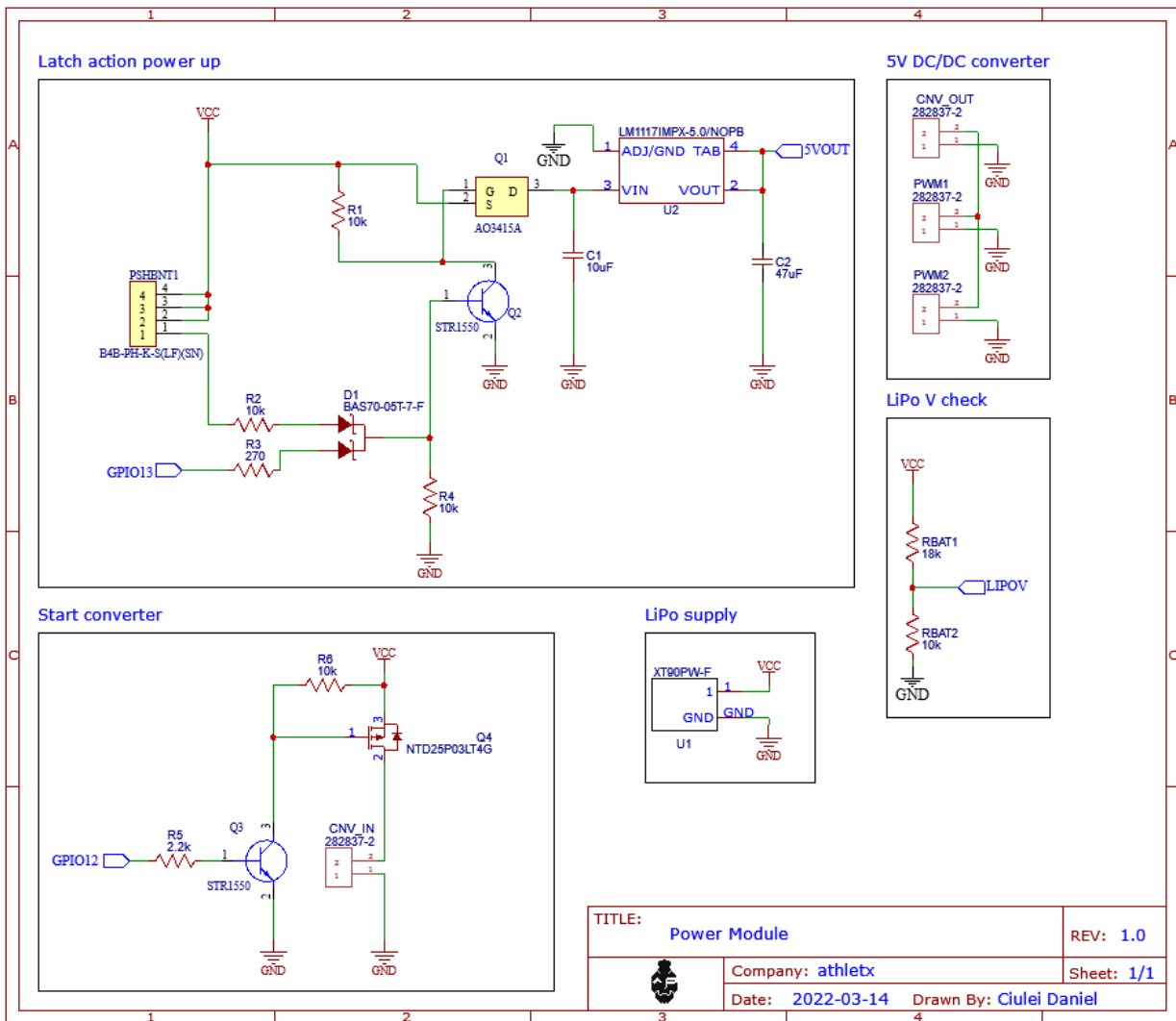


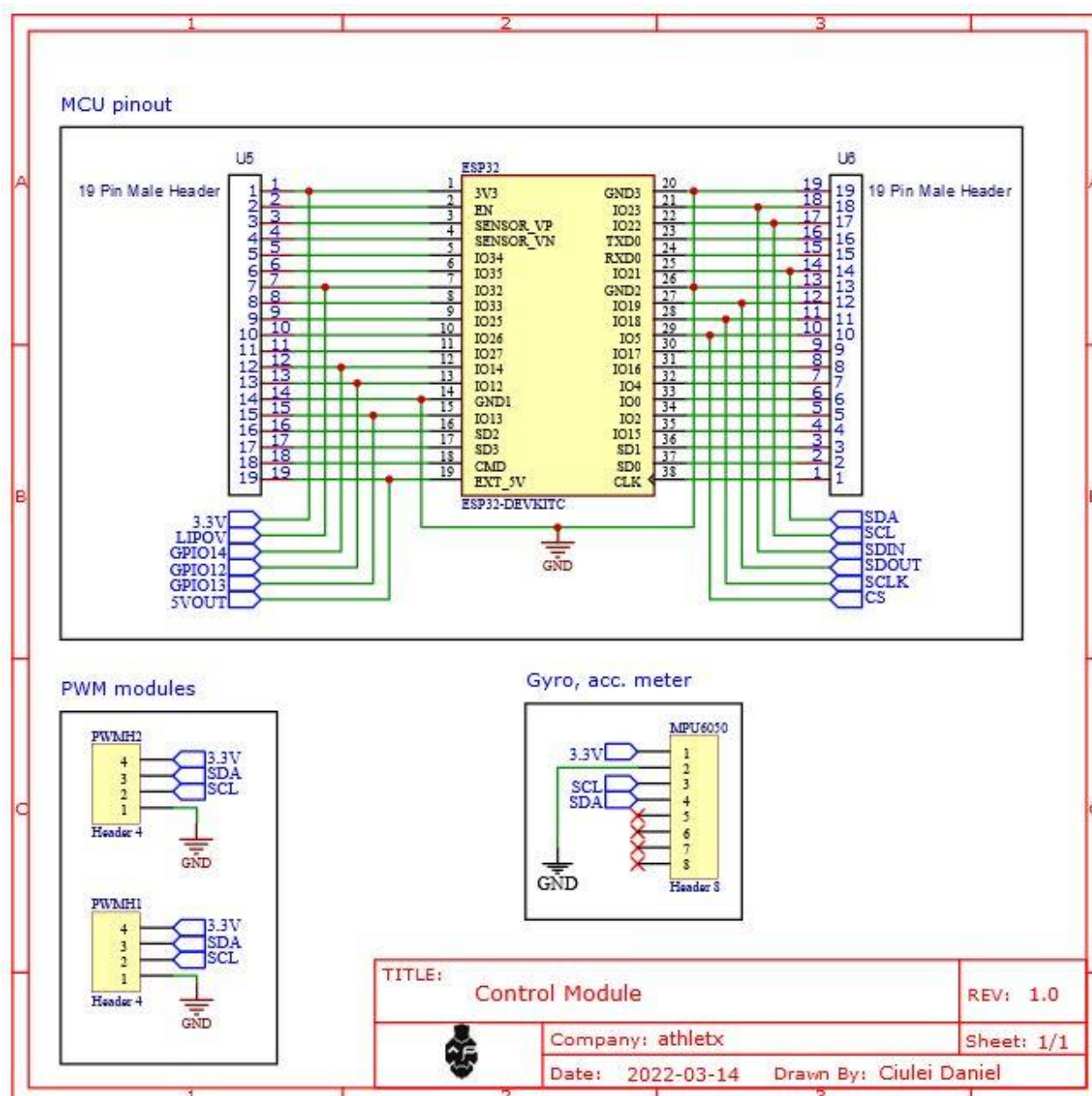
Designed by Ciulei Daniel	Dept. Electrical Engineering	Date 3/14/2022
	Project  HEXAPOD	
	Hexapod Base Frame	Sheet 1 / 1

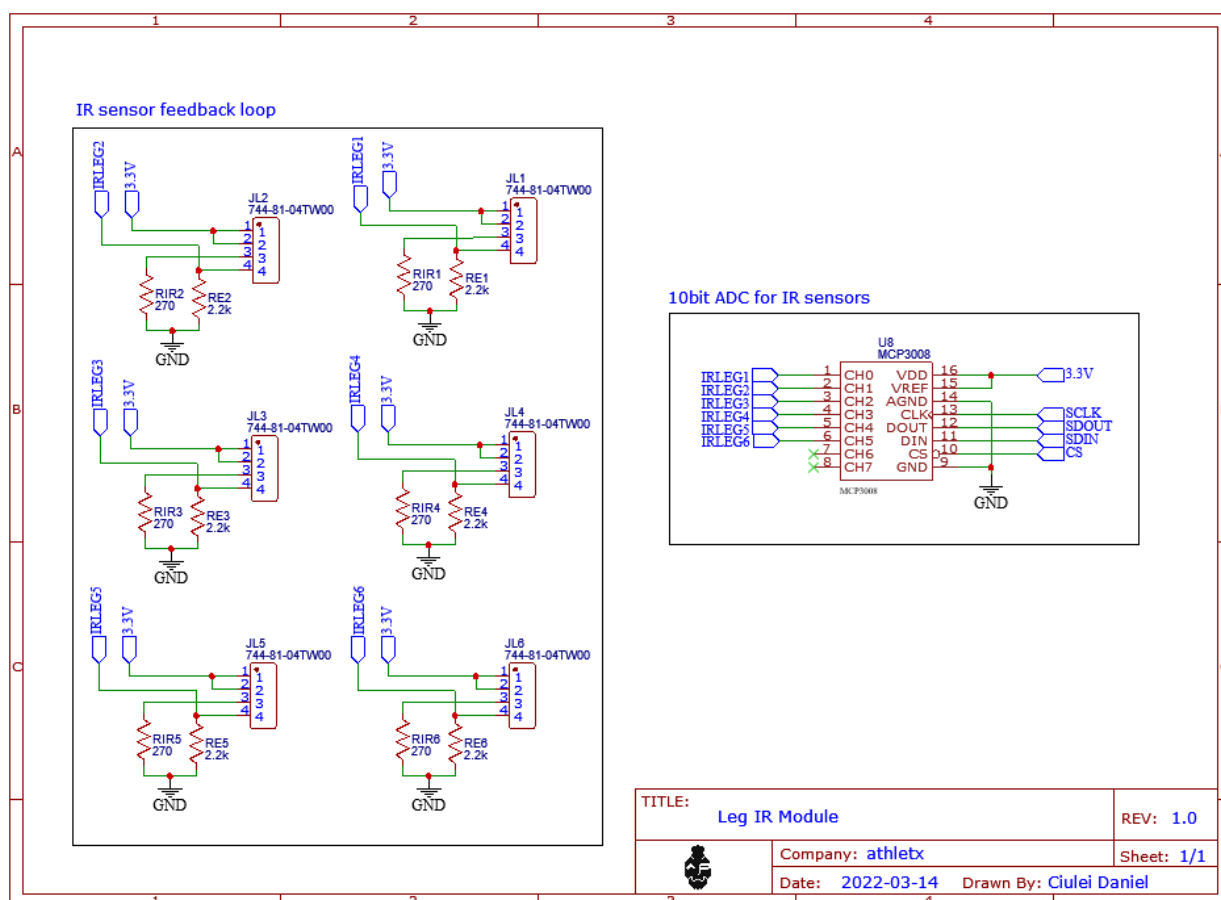




7.2. Appendix B – Hardware design







UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE, TÎRGU-MUREȘ
SPECIALIZAREA CALCULATOARE

Vizat decan
Ș.l. dr. ing Domokos József

Vizat director departament
Conf. dr. ing. Szabó László