FABRICATION AND MODELLING OF A ROBOT ARM

A Master's Thesis
Submitted to the Faculty of the

Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Alberto Zubiri Carrizosa

In partial fulfilment
of the requirements for the degree of

Master's degree in Telecommunications Engineering

Advisor: Vicente Jiménez Serres

Barcelona, November 23, 2020

Abstract

The goal of this project is the fabrication and modelling of a robotic arm . The build robot is obtained by printing its mechanical elements that are obtained by modifying a readily available template. A CAD program has been used to modify the template. Then, a G-Code generator was used to obtain the instructions for the 3D printer operation. Although only one robot has been fabricated, two have been analysed in this document. This Project is the first TFM for the current strategic plan of biorobotics in the ETSETB.

Two electronic designs have been made for both robots. Both circuits are designed to move the joints of the robots. In addition, a closed feedback has been incorporated into the joints to add more information on their position. The circuit is designed to make have a cabling complexity independent on the number of motors so that the design can be used for more complex future robots. To guarantee the operation of the circuit operation, a prototype has been implemented and tested.

Acknowledgements

Quiero agradecer a Vicente Jiménez por haberme dado la oportunidad de hacer este trabajo, y toda la ayuda prestada para poder llevar a cabo este proyecto hacia adelante.

También a Manel, Melchor, Pedro y Ramón pues una parte de este proyecto, se ha podido realizar gracias a su ayuda.

No me olvido de mis amigos que también han colaborado como Alejandro, Adri, Asier y Dani.

Y finalmente, quiero mencionar a mis padres que han sido el mayor apoyo moral, y especialmente en los momentos más difíciles.

Revision history and approval record

Revision	Date	Purpose
0	24/03/2020	Document creation
1	19/11/2020	Document revision
2	20/11/2020	Document revision
3	22/11/2020	Document revision
4	23/11/2020	Document revision

Written by:		Reviewed and approved by:	
Date	23/11/2020	Date	23/11/2020
Name	Alberto Zubiri	Name	Vicente Jiménez
Position	Project Author	Position	Project Supervisor

Table 1: Revision history and approval record

List of Figures

2.1 Example of Robots Classifications 4
2.2 Typical joints 5

3.1 6 Revolution (R) joint Robot 12
3.2 Application of the PoE formula for an n-link spatial open chain 14
3.3 Example of a Robot with a Closed Chain mechanism 15

4.1 Mantis Robot: Appearance 17
4.2 Mantis Robot: Base 18
4.3 Mantis Robot: Shoulder 19
4.4 Mantis Robot: Elbow 20
4.5 Mantis Robot: Forearm 20
4.6 Mantis Robot: Wrist part1 21
4.7 Mantis Robot: Wrist part2 and Gripper 21
4.8 MK2 Versions 22
4.9 MK2 Modification 22
4.10 MK2 Modification: The inside of the Base representation 23
4.11 MK2: Schematics and its parts representation 24

5.1 Mantis Robot: Screw axis representation 26
5.2 MK2: Motion 28
5.3 MK2: Schematics and frames of each loop 29
5.4 MK2: Schematics and frames of each loop 30

6.1 System Overview 32
6.2 Blocks: Mantis Robot 33
6.3 Blocks: MK2 Robot 34
6.4 Electronic Design: CorePCB 37

6.5 Electronic Design: DriverPCB 38
6.6 Electronic Design: EncoderPCB 40
6.7 Electronic Design: EncoderPCB2 40
6.8 Rotatory Magnetic Encoder 41
6.9 4-Wire Mode 43
6.10 3-Wire Mode 44
6.11 Multislave with 4-Mode configuration 44

7.1 Prototyping: SH-PO + Encoder + Driver Schematic 45
7.2 Prototyping: MISO, CSn and Clock from Encoder 46
7.3 Prototyping: Winding Energy 46
7.4 Prototyping: Two Encoders Schematic 47

List of Tables

1 Revision history and approval record iii

2.1 The number of degrees of freedom f and constraints c provided by common joints. 5

4.1 Mantis Robot: Link lengths 18

4.2 Mk2 Robot: Link lengths 24

6.1 Number of cables for each electronic device 35

Contents

Abstract **i**

Acknowledgements **ii**

Revision history and approval record **iii**

List of Figures **iv**

List of Tables **vi**

Contents **vii**

I Introduction and State of the art **1**

1 Introduction **2**

2 State of the art **4**

2.1 Context 4

2.2 Mechanical calculations 5

2.3 Robot actuators 7

2.4 3D Printing 8

2.4.1 Software for 3D Printing 8

II Robot Mechanics and Modeling **10**

3 Mechanical Concepts **11**

3.1 Rigid-body Motions 11

3.1.1 Frames and Homogeneous Transformation Matrices 11

3.2 Forward Kinematics: Open Chains 13
3.2.1 Product of Exponentials 13
3.2.2 Forward Kinematics: Steps 15
3.3 Forward Kinematics: Closed Chains 15

4 Robots Presentation **17**
4.1 Mantis Robot 17
4.2 MK2 Plus Robot 22

5 Robot models **26**
5.1 Forward Kinematics: Mantis Robot 26
5.2 Forward Kinematics: MK2 Plus Robot 28

III Electronics concepts and design **31**

6 Electronic Design **32**
6.1 System Overview 32
6.2 Electronics Design Blocks 33
6.3 Electronic design: Third block 34
6.3.1 Signal management 34
6.3.2 Number of cables solution 35
6.3.3 CorePCB 36
6.3.4 DriverPCB 38
6.3.5 EncoderPCB 39
6.4 Magnetic Encoder 39
6.4.1 Magnetic Encoders: Introduction 41
6.4.2 Magnetic Encoders: Static Errors 42
6.4.3 Magnetic Encoders: Dyanamic Errors 42
6.4.4 Magnetic Encoders: Communication Interface 43

7 Prototyping and Results **45**
7.1 Prototype: SH-PO + Encoder + Driver 45
7.2 Prototype: Two Encoders 47

IV Conclusions and Future Work **48**

8 Conclusions and Future Work **49**

Bibliography **51**

Acronyms **52**

Part I

Introduction and State of the art

CHAPTER 1

Introduction

The main objective of this project is the assembly of a robotic arm using 3D printing. The assembly includes modification of robot elements from a known design, the design of the electronics, the cable management and the 3D printing of its elements.

In addition, it is desired that the robot allows future evolutions, in both hardware and software within the current strategic plan of biorobotics of the ETSETB. To comply with this objective, redundant information about the robot state, both from the incremental movement of its stepper motors and by the use of absolute encoders is provided. This is the first master's final project within the school's biorobotics strategic so most of the developments start from scratch.

This work started on November 2019. Due to the student lack of experience in the mechanical field, two and a half robotics courses have been added to the student curricula. These courses were followed on an online platform, and they were taught by Northwestern University.

In this document we will work on two different robot arms: Mantis and MK2. Both are open hardware projects and any modification we made on them will be made available worldwide using Github and Thingiverse. Initially we were going to work with the first one. Because of the COVID pandemic we are having in this year 2020, we decided to change from Mantis to MK2 as the last one required less lab work. Due to the unexpected confinement period, lab testing and prototyping was also compromised.

This work includes the electrical design for the robot. It has been tested to verify its correct operation. But due to the confinement, testing was too late to be able to develop the final PCBs.

The final MK2 robot electrical design comes from the first one. Although this second robot has not the same requirements, we maintained the original concepts as they are more easy to adapt to any future robot. The finally fabricated robot is derived from a readily available design, but some elements have been modified to better suit this particular project.
Although only one robot has been fabricated, the kinematic model for both robots has been developed. As both robots has feature different mechanical structures this document

will ease future kinematics analysis on other robots.

To carry out this project, the author had to learn the use of CAD programs to create and modify parts for 3D printing. Also it was needed to learn the use of the software tools that convert 3D solid description to G-Code for the printing machine operation.

Thus, this document is structured in the following chapters:

Chapter 2: We will see concepts of mechanics, electronics and 3D printing that we need to understand the rest of this document.

Chapter 3: The main objective of this chapter is to understand how to compute the forward and inverse kinematics. That is, how to model the positions of a robot, and what are their dependencies on the actuators operation.

Chapter 4: In this chapter we will see the structure of the two robots and their most important parameters.

Chapter 5: In this chapter we compute the kinematics of the two robots. That is, the relationship between the motor actuator positions and the final arm effector position and orientation

Chapter 6: In this chapter the design of the electronics is presented. That includes the schematics, the components and the rationale for the the decisions that had been made.

Chapter 7: Before developing the final PCBs, an electrical prototype of the electrical circuits is needed to guarantee its proper operation. In this chapter, the results obtained in this prototyping stage are presented.

Chapter 8: This chapter is devoted to the final work conclusions and some proposals for future work related to this project.

CHAPTER 2

State of the art

2.1 Context

In the wide documentation that we can find about robotics, there are a lot of definitions for what is a robot. But in this document we define a robot as a machine which can apply different physical motions in a smart way. That is, robotics field belongs to the physical motions, and these will be depended by the intended application. This point is important because depending on the application, robots can belong to different classes. The figure below shows some examples of robot classes.

(a) Humanoid (b) Delta (c) Arm

Figure 2.1: Example of Robots Classifications

Humanoids 2.1a are useful for human interactions because their human appearance provide more comfortable interactions for humans. Delta robots 2.1b are adequate for medicine applications since they provide high mechanical precision in remote control of surgical devices. Finally Arms 2.1c for manufacturer industry, as they can move with high precision and they can have high robustness in front of high loads. Those are just some examples of applications.

From the mechanical point of view, there is another type of classification which it depends on the body deformations. If there is any deformation of the robot bodies, we are talking about **soft robotics**, if not, it is a **rigid robotics**. In this work, we only consider

rigid bodies.

2.2 Mechanical calculations

Mechanically, a robot is formed by a set of bodies, called **links**, that are connected by **joints**. A joint is usually moved by one or several **actuators**, such as electrical or pneumatic. The number of actuators that can be connected to the joint depends on the type of joint (Figure 2.2), and the mechanical structure of the robot.

Joint movement is related to the number of **Degrees of Freedom (DOF)**. The number of DOF is the number independent motions a link can make, The full robot number of DOF, as we will see, relates to each joint number of DOF. The maximum number of DOF for a joint is 3 in two dimensions a 6 in three dimensions. In 2 Dimensions (2D), or planar mechanism, we have 2 linear and 1 rotational movements. In 3 Dimensions (3D), or spatial mechanism, we have 3 linear and 3 rotational movements. The number and types of motions that a body can move will be depended by the kind of joints it includes.

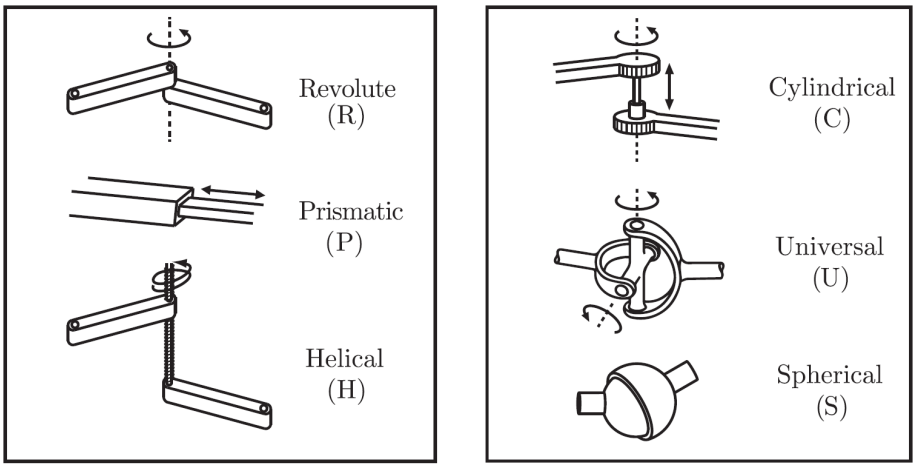


Figure 2.2: Typical joints

Joint Type	DOF number	Constraints c between two planar rigid bodies	Constraints c between two spatial rigid bodies
Revolute (R)	1	2	5
Prismatic (P)	1	2	5
Helical (H)	1	N/A	5
Cylindrical (C)	2	N/A	4
Universal (U)	2	N/A	4
Spherical (S)	3	N/A	3

Table 2.1: The number of degrees of freedom f and constraints c provided by common joints.

In the Figure 2.2 and the Table 2.1 (both from [1], Chapter 2) we can see the common

joints that are used, and the DOF and constrains associated to them. A **constrain** is a limitation where the body can't move, that is, it's the opposite concept of a DOF. Both concepts give the same information but from opposites points of view. The joints that can work in two dimensions are the Revolute (R) and Prismatic (P), the others can only be implemented in 3D.

Until now, we have described the joints between robot links. Now we need to translate the information about all the joints and links in the robot to the mechanical behaviour of the full robot structure.

First of all, robots can have two kinds of mechanical structures, **open chain** and **closed chain**. In the second case there is a closed loop among links, for example, a Humanoid robot (Figure 2.1a) has a closed loop between its legs and the ground (the ground is considered as a link as well). On the contrary, in an open chain mechanism there is not any loop, and all links are in serial, such as an arm of the humanoid robot. A robot can have only one type of mechanism like Delta (Figure 2.1b) or Arm robot (Figure 2.1c) where there are only closed and open chain respectively, or both types of mechanisms like Humanoids.

That classification is important in order to know if we need a motor for each DOF on a particular joint. It will depend on the type of joint: Active and Passive. An **active joint** is the joint where their DOFs are associated to connected directly to actuators, in contrast to a **passive joint** is free to move. Open chain mechanisms can only have active joints, but closed chain can contain both active and passive joints.

Until now, we were focusing the number of DOF of a joint. This parameter will be labelled as f_i, that is, the number of DOF of the joint i. Now we want to know how many DOF has the full robot. To do it, we can use **Grübler's Formula** ([1], Chapter 2):

$$NRobotDOF = m(N - 1 - J) - \sum_{i=1}^{J} f_i \tag{2.1}$$

Where m represents the maximum of number of DOF that a link can have depending if it is a planar 2D (m=3) or spatial 3D (m=6) mechanism. N tells the number of links and J the number of joints of the robot. And finally, the number of DOF at every i joint is given by f_i parameter.

Once we know how many DOF the robot has, we can know the number of actuators to use for the robot. A DOF means an independent motion for one coordinate, such as a rotation around an axis or a translation through an axis. But the robot needs some device which provides this motion. There are motors which provides a linear motion, and others a rotational motion. Therefore, if each motor can control one joint DOF, a robot will need the same number of motors as the number of DOF has.

In a robot, it is key to know its current mechanical state, known as its **configuration**. The configuration is a specification of the positions of all points of the robot. The number of DOF of a robot is the smallest number of real-valued coordinates needed to represent its configuration. For an open chain robot that contains 6 Revolution joints, for example, its representation could be the six angle rotations of the joints.
The n-dimensional space containing all possible configurations of the robot is called the **configuration space (C-space)**.

We define the **work space** as a multidimensional space that contains all the positions and orientations the end-effector can have. In a 3D robot it will be typically a 6 dimension space (3 rotations and 3 translations).

For a particular application we can define a **task space** that is a subset of the work space that includes all the positions and orientations the end-effector must be able to reach to perform the intended application.

For instance, for a robot drawing on a paper, the work space can be all the positions of the table the effector can reach whereas teh task space could be limited to just one sheet of paper.

The end-effector is at the tip of the robot, thus, in order to reach to some point of the space the links between the ground and the end-effector have to move them. That is, each link of the robot collaborates for the end-effector can reach at the desired point.

Briefly, **kinematics** describes robot motions without considering the origin causes, that is, without considering any forces and torques. It uses velocities that they describe how the position is changed during the time [2]. **Forward kinematics** of a robot is the calculation of the position and orientation of its end-effector (end-effector space) given the joint parameters of the robot (actuator space). But, as our interest is mainly to control the end-effector space, we can do the inverse computation that obtains the required active joint actuator, or actuator space, from the desired end actuator state, called **Inverse kinematics**.

In order to guarantee the correct operation of the robot we also need to understand the forces and torques that causes the motion of the robot. This field is known as **robot dynamics**. **Forward Dynamics** computes the accelerations given the positions, velocities, motor torques or forces and the wrench applied by the end-effector. Meanwhile, **inverse dynamics** refers to the motor torques or forces computation given the positions, velocities and accelerations.

2.3 Robot actuators

Active joints are moved by **motors**, that is, devices whose propose is converts electrical energy into mechanical energy. Although there are linear motors, in this project we only consider rotational motors. There are several types of motors, such as DC motors, servomotors and stepper motors, which they produce rotational motions. In first one the mechanical energy is produced by a continuous DC electrical energy, the second one has a internal DC motor with speed or position feedback control. And finally, the stepper motor is a little bit different than the others because it needs a pulse sequence for the shaft motion. When a stepper motor is receiving a DC electrical signal the shaft won't move. A stepper motor can be supplied directly by a voltage power supply, but its operation would be very inefficient. In general we will need high voltage to overcome the motor inductance at the start of the motion and lower voltage to maintain the final position with the desired torque. Thus, we will typically use a driver that manages the electrical energy for a stepper motor in a smarter way.

The robot will need also other mechanical elements like gears to increase/reduce torque and speed and bearings to reduce joint friction.

In order to have a feedback of a joint DOF state, we can use an **encoder**. There are several types of encoders based on mechanical, optical or magnetic principles. In our project we will use absolute magnetic rotational encoders as they are cheap and provide the desired precision. It should be noted that stepper motors provide a good tracking of the incremental position of the joints they control. Adding absolute encoders provide some redundancy that can be used both to determine the joint state at start-up and to check that the load is not above the motor capabilities.

One important part of the robot design is cable management. When cables pass through a rotation, they can have torsion efforts. Here we have three options. Firstly, we can put mechanical limits. The second option is to include detector switches that provides a signal to stop the motor motion. And the third option is to use a device which allows continuous rotational movement. A **Slip ring** is one of these devices that allow rotational movements preventing torsion stress on the cabling.

2.4 3D Printing

Once a robot is designed, it needs to be fabricated. Some elements will be standard, like motors and bearings, and some others will need to be custom made, like most link elements.
In a mass produced robot, there are several fabrication methods available for the custom parts. For low volume robots, the options are more restricted. One fabrication method is additive 3D printing commonly in plastic, but there are other type of materials as well.
As 3D printed pieces are custom made, one at a time, it is a versatile fabrication process to develop new robot prototypes. Plastic 3D printed elements are usually weaker that other alternatives, but they are also easier and cheaper to produce for single pieces. But depending on the application they are sufficiently good. In our case 3D printing provides a good way to develop robotic prototypes at low cost, or other field where 3D printing is cost effective is prosthetics where each element needs to be custom made.

In order to print a piece, previously the designer has to generate a G-Code file which is required by the 3D printer.

2.4.1 Software for 3D Printing

To obtain the required printer file, we need to perform two steps:

Firstly, we have to design the piece. **Computer-aided design (CAD)** software is the properly tool to make it. With this program, we can create and modify 3D pieces. Besides, once several pieces are created and/or modified, the software provides tools to perform a virtual assembly. It is very useful to make sure that all created pieces will fit together. Also, it has another tools, like making drawings with measurements, or animations to see the assembly process.
There are several commercial software like Inventor, Fusion 360 or Solidworks, and free software like FreeCAD or OpenSCAD. After designing, we have to export piece by piece as a Standard Triangle Language (STL) file format, or another format able to describe a 3D object.

The previously described CAD tool gives, as its final result, a 3D descripton of a solid object. Now, we need to generate the steps the 3D printer need to perform to create this piece. Numeric machine lenguaje is G-Code. G-code is composed of instructions that control the machine movements and other actions like plastic filament extruding. So, we need to use a software tool to generate the G-Code from the 3D solid description. Examples of G-Code generators are Ultimaker Cura or Slic3r.

The G-Code generator also configures the pieces according to their performance requirements. For example what is the pieces infill, how many layers for the walls of the pieces or if there is a necessity of external supports.

Part II

Robot Mechanics and Modeling

CHAPTER 3

Mechanical Concepts

3.1 Rigid-body Motions

In this section we want to explain the mathematical tools that we need to compute the Forward kinematics with Product of Exponentials (PoE) methodology (explained at the section 2.2).

3.1.1 Frames and Homogeneous Transformation Matrices

First we must define frames. A frame is a way to represent a rigid-body, that is, a link. This representation consists of an origin and orthogonal x, y, and z coordinate axes. Several examples are in the Figure 3.1. The figure shows a six revolution joint robot, where there are four links including the ground, thus there are four frames too. A frame represents only a link, and as we can see in the image, frames are located in the rotational axis of each link. For example, joint $\{1\}$ rotate directly L_1. The frame $\{0\}$ represents the ground which is considered as a link too. This frame is special because it will be the called space $\{s\}$ frame. Space frame is the reference frame, and in order to represent the other frames, they will be defined respect to the space frame. For example, frame $\{1\}$ is just rotated θ_1 with respect to the frame $\{0\}$, and the frame $\{2\}$ is rotated and translated with respect to the $\{0\}$. The space frame is chosen by convenience. In that case, frame $\{0\}$ is the most intuitive to know the relations with respect to the others. Frames are used for every open and closed chain mechanism.

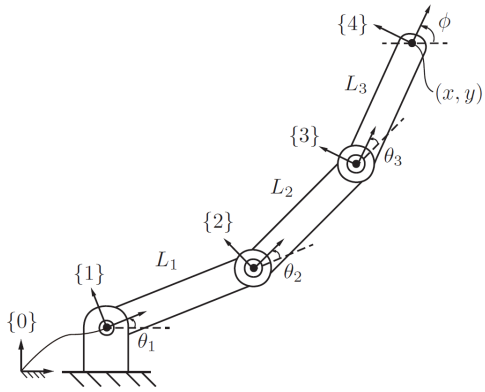


Figure 3.1: 6 Revolution (R) joint Robot

In order to represent the relations between frames, we can use matrices called homogeneous transformations. These matrices have two main components, the rotational part (R) which is a matrix, and the translation part (p) which is a vector.

$$T_{sb} = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{x1} & r_{y1} & r_{z1} & p_x \\ r_{x2} & r_{y2} & r_{z2} & p_y \\ r_{x3} & r_{y3} & r_{z3} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

The "sb" sub-index means the relation between the space and the body frame. The first column of rotational matrix represents the rotation of the "x" coordinate, and the same in the second and third column, with "y" and "z" coordinate respectively.

If we continue the example of the Figure 3.1. The relation between the frame {1} with respect to the frame {0} will be the following. In this case we have 3x3 matrix because it is in a planar space:

$$T_{01} = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{x1} & r_{y1} & p_x \\ r_{x2} & r_{y2} & p_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} cos\theta_1 & -sin\theta_1 & 0 \\ sin\theta_1 & cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

And the relation between the frame {2} with respect to the frame {0} will be:

$$T_{02} = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{x1} & r_{y1} & p_x \\ r_{x2} & r_{y2} & p_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} cos(\theta_1+\theta_2) & -sin(\theta_1+\theta_2) & L1 \cdot cos(\theta_1) \\ sin(\theta_1+\theta_2) & cos(\theta_1+\theta_2) & L1 \cdot sin(\theta_1) \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

But, instead of a direct T_{02} computation, we can do that in three steps. The first one T_{01} computation, the second one T_{12}, and the third: $T_{02} = T_{01} \cdot T_{12}$. Thus, if we want to compute the relation between {0} and {4}, T_{04}:

$$T_{04} = T_{01} \cdot T_{12} \cdot T_{23} \cdot T_{34} \tag{3.4}$$

Then, in order to make the inverse transformation, we can do the following:

$$T_{sb}^{-1} = T_{bs} = \begin{bmatrix} R^T & -R^T p \\ 0 & 1 \end{bmatrix} \tag{3.5}$$

Homogeneous transformation matrices are useful for open and closed chain. And they have three uses:

- Represent an orientation, as we saw in the example T_{01} and T_{02}
- Change reference frame, by multiplying two frames as we have seen before with $T_{02} = T_{01} \cdot T_{12}$.
- Displace a vector frame, if we multiply a vector instead of a homogeneous transformation matrix.

3.2 Forward Kinematics: Open Chains

Forward kinematics refers of a robot refers to the calculation of the position and orientation of its end-effector frame from its joint coordinates θ.

The most important consideration when we start the kinematics computation is to know what kind of mechanism of the robot is. Open and Closed chains have different methods. We will start for open chain mechanism as they are easier to deal with.

There are two ways in order to make it: Based on Denavit-Hartenberg Parameters (D-H Parameters), or based on Product of Exponentials (PoE). The first one is based on the relative distances between frames including the among between space and body frames, whereas for PoE only screw axis of each joint. Both offer a closed form computation, and there is not a clear advantage in order to choose a methodology. But here we will follow the recommendation of the book [1] Chapter 3. More information about D-H Parameters [1] Appendix C of [1].

3.2.1 Product of Exponentials

The objective with PoE method is to compute the position and orientation of the $\{b\}$ frame, or end-effector frame, with respect to $\{s\}$. Thus, the homogeneous transformations and the angle for each rotation are required. But in this case, the calculation of the homogeneous transformations matrices is based on screw axis theory ([1], Chapter 3.2, Chapter 3.3, and Chapter 4.1), which uses the angular and linear velocities as a tool to realise instantaneous rotations.
The mathematical expression that relates the $\{b\}$ frame with respect to $\{s\}$ is the following:

$$T(\theta) = e^{[S_n]\theta_n} \cdot e^{[S_{n-1}]\theta_{n-1}} \cdots e^{[S_0]\theta_0} \cdot M \tag{3.6}$$

And its application is described in the following figure:

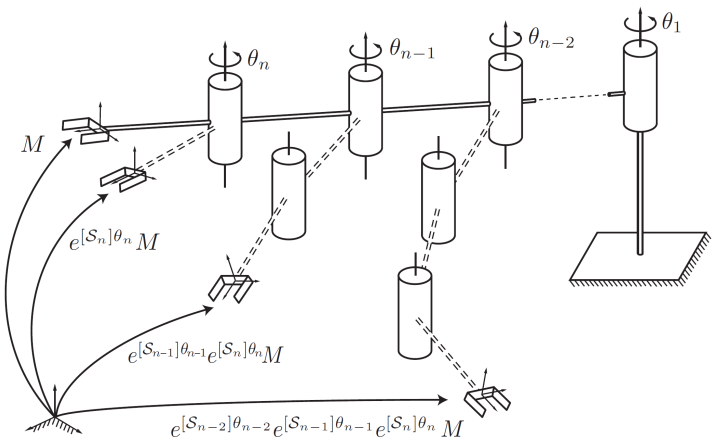


Figure 3.2: Application of the PoE formula for an n-link spatial open chain

Where S_n is the screw axis at n joint, θ_n is the angle rotation at n joint, and M is the home position matrix of the end-effector frame with respect to, and this parameter M is the homogeneous transformation when all $\theta's$ are zero.

S_n is a 6x1 vector which contains the unit angular velocity (ω_n) and linear velocity (v_n) at joint n. The form of $[S_n]$ is described as the following expression:

$$[S_n] = \begin{bmatrix} [\omega_n] & v_n \\ 0 & 0 \end{bmatrix}, \qquad [\omega_n] = \begin{bmatrix} 0 & -\omega_{n3} & \omega_{n2} \\ \omega_{n3} & 0 & -\omega_{n1} \\ -\omega_{n2} & \omega_{n1} & 0 \end{bmatrix} \tag{3.7}$$

The $[\omega_n]$ is the angular velocity vector ω_n 3x1, which is converted as a matrix 3x3. In order to calculate the angular ω_n and linear v_n velocities, we make the following expressions:

$$\begin{bmatrix} \omega_n \\ v_n \end{bmatrix} = \begin{bmatrix} \widehat{s_n} \\ -\widehat{s_n} \times q_n + h\widehat{s_n} \end{bmatrix} \tag{3.8}$$

Where $\widehat{s_n}$ is the unit screw axis, q_n is a point on the screw axis, and the pitch h of the screw which is the ratio of the linear speed along the axis to the angular speed about the axis. The screw axis defines the direction the body is moving.

The h value is zero for revolution joints.

And now we have all elements to compute the exponentials:

$$e^{[S_n]\theta_n} = \begin{bmatrix} e^{[\omega_n]\theta_n} & G(\theta_n)v \\ 0 & 1 \end{bmatrix} \tag{3.9}$$

$$e^{[\omega_n]\theta_n} = I + sin\theta_n\,[\hat{\omega_n}] - (1 - cos\theta)\,[\hat{\omega_n}] \tag{3.10}$$

$$G(\theta_n) = I\theta_n + (1 - cos\theta_n)\,[\omega_n] + (\theta_n - sin\theta_n) \tag{3.11}$$

Being $e^{[\omega_n]\theta_n}$ the definition that is called Rodrigues' formula.

The Figure 3.2 shows what is doing the homogeneous transformation via PoE methodology. If we apply $e^{[S_n]\theta_n}$ to the matrix M home position, we know the end-effector position when just the joint n is moved θ_n.
If we apply $e^{[S_n]\theta_n}e^{[S_{n-1}]\theta_{n-1}}M$, we know the end-effector position when the joints n and $n-1$ are moved θ_n and θ_{n-1}, respectively.

3.2.2 Forward Kinematics: Steps

Therefore, the following steps have to be done when we want to compute the Forward Kinematics on open chain robots.

1. Let M be the transformation matrix of the end-effector frame {s} when $\vec{\theta} = 0$.
2. Find the {s}-frame screw axes $S_1, ..., S_n$ for each of the n joint axes when $\vec{\theta} = 0$.
3. Given θ, evaluate the product of the exponentials PoE formula in the space frame.

3.3 Forward Kinematics: Closed Chains

The forward kinematics computation in closed chains can not be so easy. Unlike the open chain which the forward kinematics is generally straightforward, the presence of unactuated joints doesn't allow the same methodology.

The presence of passive joints implies that there are not actuators that directly control their movement, therefore, we can not know in advance how much they have moved, since they move in relation to the mechanical structure of the robot, and the movement made by the actuators which are in the robot. Therefore, in forward kinematics computation for robots which have closed chain, we want to find the movements of the passive joints for given rotations in the active joints.

In the Figure 3.3 we can see an example of a robot with a closed chain.

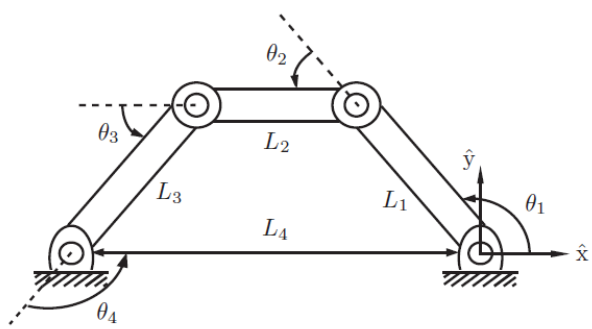


Figure 3.3: Example of a Robot with a Closed Chain mechanism

If the joint 1 (θ_1) was the actuator, the other would be a passive joints because this system has 1 DOF. Thus, we want to resolve the rotations θ_2, θ_3 and θ_4 for a given rotation in joint θ_1.

A way to compute the forward kinematics is to exploit certain features of the particular case of the robot in order to find out the equations, and then we can try to simplify in order to solve it. It is not general way to solve it.

For example, if we want to find the joint angles of the robot of the Figure 3.3, we can express the following equations.

$$
\begin{aligned}
L_1 cos(\theta_1) + L_2 cos(\theta_1 + \theta_2) + L_3 cos(\theta_1 + \theta_2 + \theta_3) + L_4 cos(\theta_1 + \theta_2 + \theta_3 + \theta_4) = 0 \\
L_1 sin(\theta_1) + L_2 sin(\theta_1 + \theta_2) + L_3 sin(\theta_1 + \theta_2 + \theta_3) + L_4 sin(\theta_1 + \theta_2 + \theta_3 + \theta_4) = 0 \\
\theta_1 + \theta_2 + \theta_3 + \theta_4 - 360 = 0
\end{aligned}
$$

Further, there is a relation between joint angles that we can simplify the calculation, as we can see: $\theta_1 = \theta_4$ and $\theta_2 = \theta_3$. Thus, we can solve it, since we have two unknowns and 3 equations.

CHAPTER 4

Robots Presentation

4.1 Mantis Robot

In this section, we will see the first robot robot considered in this work. In the Figure 4.1, we can see how is the external appearance:

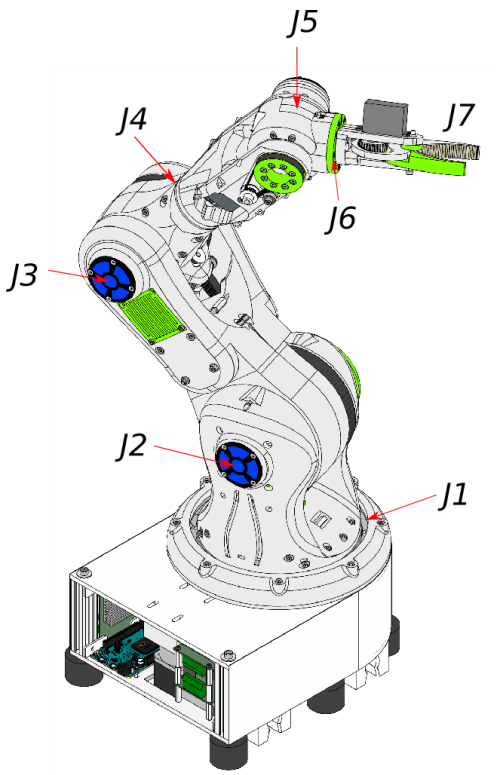


Figure 4.1: Mantis Robot: Appearance

This robot is composed by six main parts: Base, Shoulder, Elbow, Forearm, Wrist divided by two parts, and Gripper. It consists on an open chain robot with 6 (R) revolution joints robot and a gripper, and 7 links including ground. The lengths of the links are as

the described in the following table.

Link	Length (mm)
L1	173.5
L2	225
L3	29.5
L4	200.855
L5	31.4

Table 4.1: Mantis Robot: Link lengths

The first analysis of this robot will be its number of freedoms. If we apply Grubbler's formula in space representation ($m = 6$):

$$DOF = m(N - 1 - J) - \sum_{i=1}^{J} f_i = 6(7 - 1 - 6) - \sum_{i=1}^{6} 1 = 6 \tag{4.1}$$

The number of DOF that this robot has is six. It makes sense because this robot has six revolution joint, and we know that a revolution joint has 1 DOF. Thus, each joint has to be driven by an actuator. If we added the gripper joint, the robot would reach up to 7 DOF. However, gripper joint is usually not included because it is usually a modular piece. That is, this part of the robot can be exchanged to another gripper with more DOF.

At this point, we will see how works the robot internally to understand its mechanism.

In the **base** (Figure 4.2), we can see that it is where is placed all drivers (later we will present our proposal of the electronic design). From a mechanical perspective, the most important components are the geared stepper motor and the pulleys. Geared stepper motor means that the motor has a set of gears inside providing more torque to the shaft rotation.

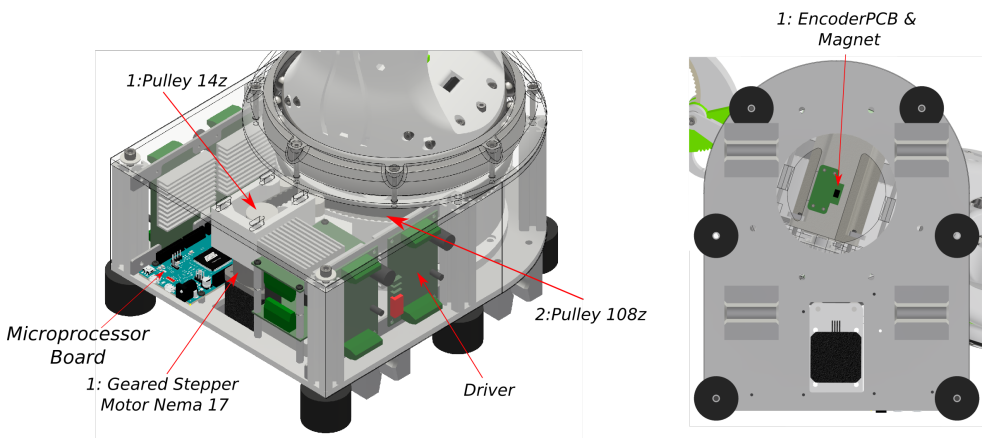


Figure 4.2: Mantis Robot: Base

Then, the function of the pulleys is very similar to gears since they are geared, thus, torques and angular velocities are reduced or increased in a similar way. In this case, there

is a reduced torque thanks to this pair of pulleys. Here the first joint of the robot "$J1$" is rotated, and the motor needs to produce 7.71 times less torque.

$$M_{base} = \frac{Z_2}{Z_1} = \frac{108}{14} = 7.71 \tag{4.2}$$

The first encoder and magnet, which are explained at Chapter 5, are placed at the bottom (see the right view of the Figure 4.2, the plastic where the magnet is placed is transparency to better see the PCB where is the encoder). The encoder is placed at the bottom of the shoulder, that is the mobile part of the joint "$J1$", and the magnet is placed at the internal surface of the base, that is, the fixed part. Both have the same axis.

In **shoulder**, there is the stepper motor which moves the second joint ($J2$).

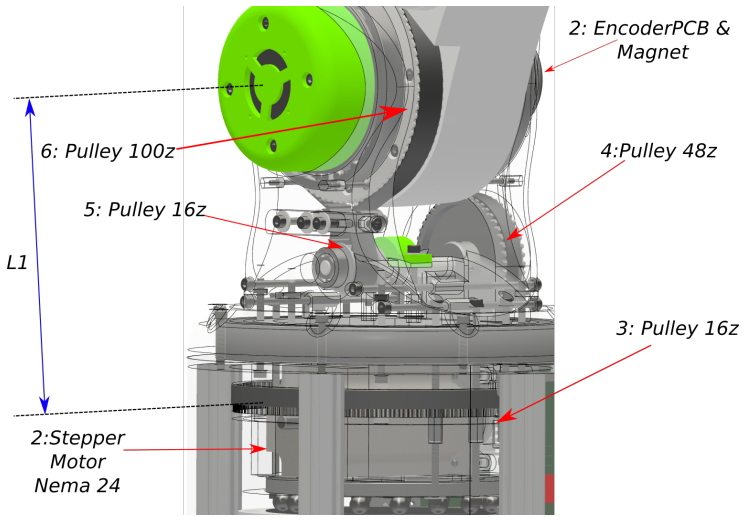


Figure 4.3: Mantis Robot: Shoulder

Here, there is a increased torque in two phases. The first phase is composed by the driver pulley "$3 : Pulley16z$" and the driven pulley "$4 : Pulley48z$". Meanwhile, the second one is composed by the pulleys "$5 : Pulley16z$" and "$6 : Pulley100z$". All pulleys are metallic except the last which is plastic, making by a 3D printer.

$$M_{shoulder} = \frac{Z_4}{Z_3} \cdot \frac{Z_6}{Z_5} = \frac{48}{16} \cdot \frac{100}{16} = 18.75 \tag{4.3}$$

At this case, the motor (*2:Stepper Motor Nema 24*) needs to make 18.75 times less of torque than the joint "J2" needs. The encoder is placed at the fixed part of the joint $J2$, the shoulder, and the magnet is at the mobile part, the elbow.

The **rest** of joints "*J3, J4, J5 and J6*" works with the same way. At the bottom of the part is placed is where the motor is placed, then there are a set of geared pulleys which put a reduction factor for the torque. In addition, the encoder is placed at the fixed part of the joint "Ji" and the magnet is placed at the mobile part.

The particularities are at the first part of the wrist (Figure 4.6), and the second part (Figure 4.7) because of the slip ring component. It is the component which allow

continuous rotations avoiding stressing the cables that pass through these joints of the robot.

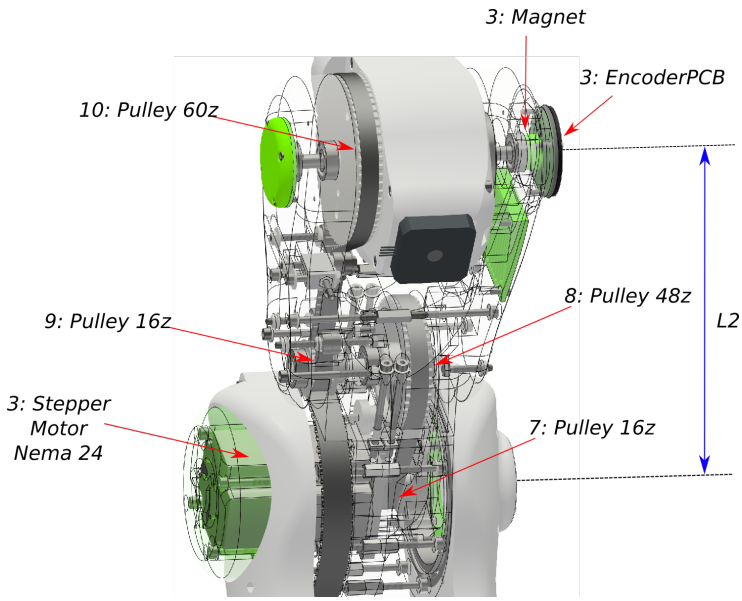


Figure 4.4: Mantis Robot: Elbow

$$M_{elbow} = \frac{Z_8}{Z_7} \cdot \frac{Z_{10}}{Z_9} = \frac{48}{16} \cdot \frac{60}{16} = 11.25 \tag{4.4}$$

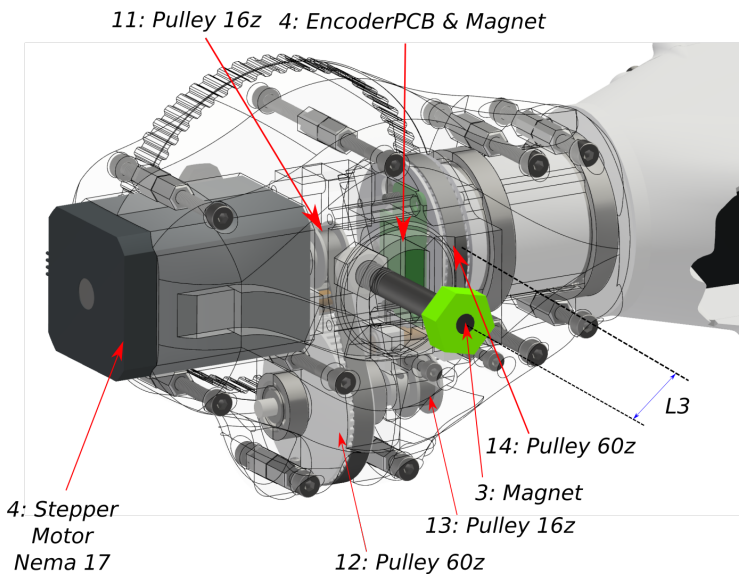


Figure 4.5: Mantis Robot: Forearm

$$M_{forearm} = \frac{Z_{12}}{Z_{11}} \cdot \frac{Z_{14}}{Z_{13}} = \frac{60}{16} \cdot \frac{60}{16} = 14.06 \tag{4.5}$$

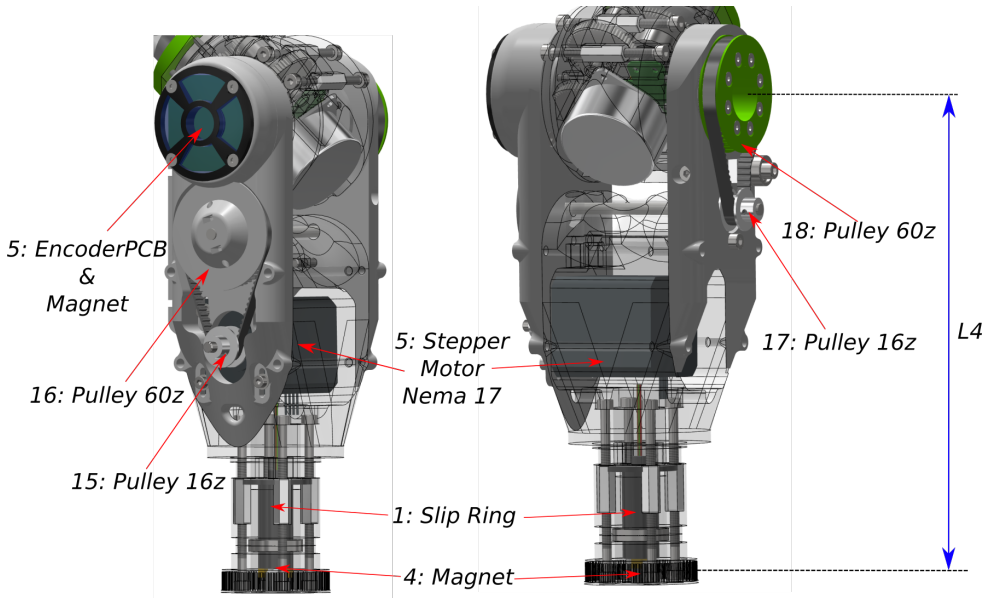


Figure 4.6: Mantis Robot: Wrist part1

$$M_{Wrist_{p1}} = \frac{Z_{16}}{Z_{15}} \cdot \frac{Z_{18}}{Z_{17}} = \frac{60}{16} \cdot \frac{60}{16} = 14.06 \tag{4.6}$$

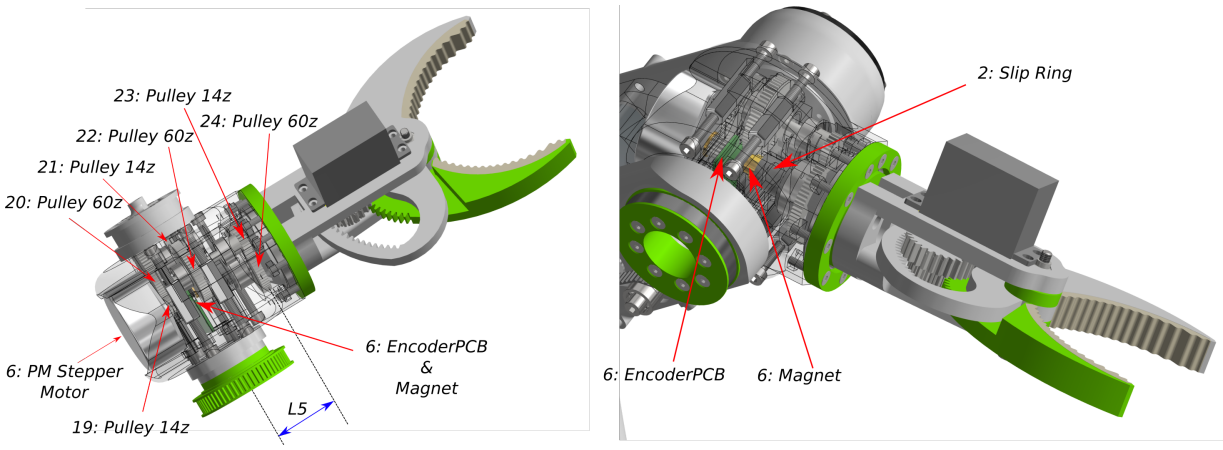


Figure 4.7: Mantis Robot: Wrist part2 and Gripper

$$M_{Wrist_{p2}} = \frac{Z_{20}}{Z_{19}} \cdot \frac{Z_{22}}{Z_{21}} \cdot \frac{Z_{24}}{Z_{23}} = \frac{60}{14} \cdot \frac{60}{14} \cdot \frac{60}{14} = 78.71 \tag{4.7}$$

The budget of this robot is the appendices. In the original project there was not list of components, and the only information was in the CAD project. Therefore it has been necessary to make a list of components and to look for where to buy them. In the search we tried to find them in stores that had an agreement with the ETSETB, such as RS Amidata or Farnell, but there are several components that have had to be in other stores. The components that have presented difficulties in their location have been screws and bearings due to their great diversity of standards. And also the slip rings, and

the magnets. Diametrical magnetised magnets are more unusual than axial ones on the market. In addition, regarding the stepper motors, it has been necessary to choose some ones with similar characteristics to be able to buy them in the preferred stores.

4.2 MK2 Plus Robot

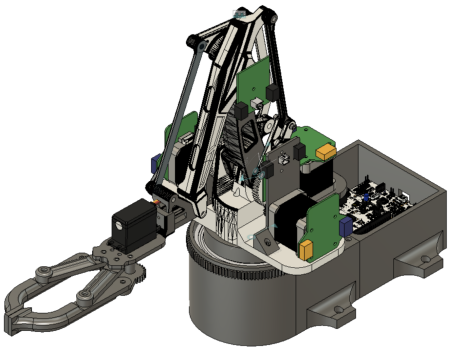
Because of the COVID pandemic reasons that we are having in this year 2020, we decided to change the robot. This work starts before COVID, but due to the confinement period associated to this unexpected situation, it was impossible to go to the laboratory for testing and prototyping some components that the robot required, and we hadn't enough time to prototype the electronic part, print the pieces and put together all pieces.

The main feature of this kind of robot is the motors position where all of them are on a rotating base and we can avoid to put in the arm. This is thanks to the second main feature, that is, the use of closed chains. Thanks to these closed chains there are recursive mechanisms which are useful for the control of these parts of the robot.

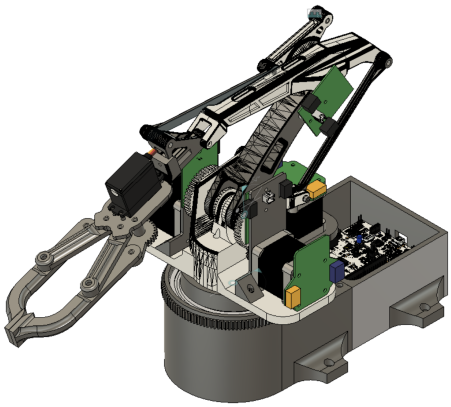
(a) MK2: Original

(b) MK2 Plus

Figure 4.8: MK2 Versions



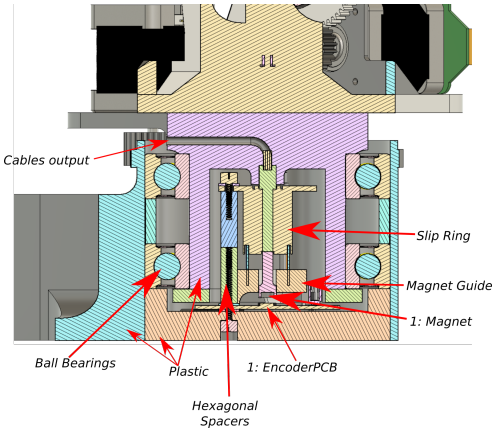
(a) MK2 Modification: Limits



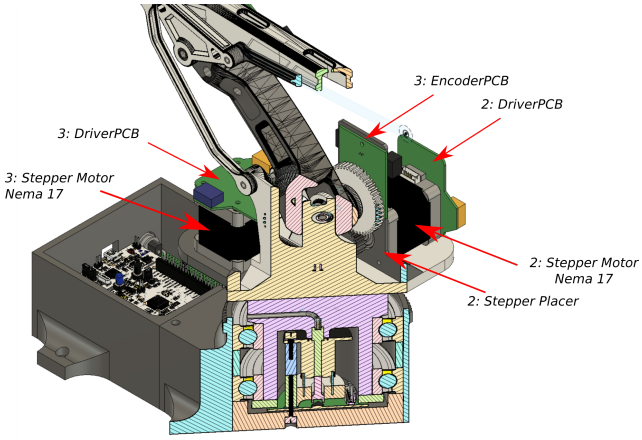
(b) MK2 Modification: Home Position

Figure 4.9: MK2 Modification

This robot has two previous versions as we can see in the Figure 4.8. Figure 4.8a comes from the original project. It's a robot based on servo motors. Then, in the Figure 4.8b the main change the motor type, which in this case the stepper motor is used instead. As we want take an advance of the designed electronic circuit, the second option is fit for our purpose (Figure 4.9).



(a) MK2 Modification: Base in 2D



(b) MK2 Modification: Base in 3D

Figure 4.10: MK2 Modification: The inside of the Base representation

Some modifications have been made for this robot:

1. The **base** is modified to add a slip ring inside. In this way, the robot can rotate more than 360 degrees infinitely. Figure 4.10 shows a plane that it is cut through the centre of the rotational axis. We can better how is the base inside. There is the first encoder at *1:EncoderPCB*, and the first magnet on the top of the encoder surface. The magnet is placed inside a guide made by a 3D printer to ensure magnet-encoder alignment. As we can see the slip ring, the top centre part is the same than the bottom, thus, the rotation done by the pink part of the figure is the same than the magnet.

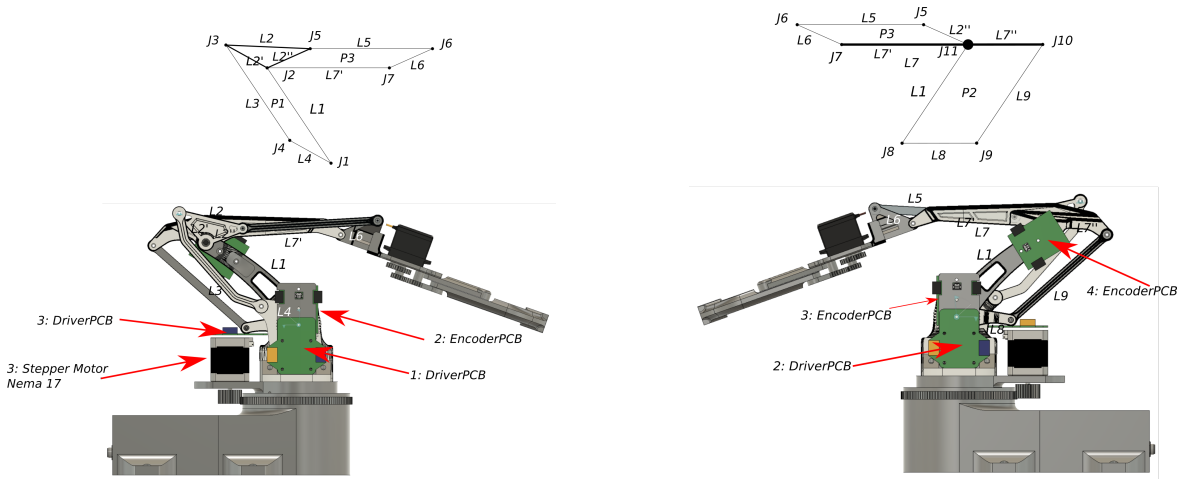
2. The second one is the **box** addiction for the microprocessor board, and a PCB where the power supply connector is placed. The box has the holes for the external connections.

3. Over the base, plastic pieces where before they just support the stepper motors Nema 17 but also for **for the encoder PCB boards**. Indeed they are aligned to the axis where there are each corresponding magnet. The idea is the same than Mantis robot.

In order to make easier to calculate the n DOF, we can divide the robot into two parts. The first one is the base part where clearly there is just one rotation DOF. However, for the other part, the arm part, we can use the Grubblers' Formula as before:

$$DOF = m(N - 1 - J) - \sum_{i=1}^{J} f_i = 3(9 - 1 - 11) + \sum_{i=1}^{11} 1 = 2 \tag{4.8}$$

Despite the fact that the robot works in a 3D space, the closed chains parts work in a planar space, that's why in this case $m = 3$. The result makes sense (excluding the gripper), but it could not be so intuitive than Mantis robot because of the closed chain mechanisms put together. But if we divide the arm in three parts like this:



(a) MK2: Part 1 and 3

(b) MK2: Part 2 and 3

Figure 4.11: MK2: Schematics and its parts representation

These parallelograms form require 1 DOF each one. Two of them there are independent, and the last is depended on the two firsts, since it is serial with respect to of them. Therefore, two need an actuator, and the other doesn't it. The 2 DOF make sense again.

Let's to explain somethings to understand better the arm part of the robot. $P1$, $P2$ and $P3$ are the closed chain mechanisms. $J1$ and $J8$ are connected to a set of gears where the torque reduction for the actuators is $47/19 = 2.47$ for both. And the *3: Stepper motor Nema 17* has a reduction torque of $108/24 = 4.5$. Besides, $J1$ and $J8$ are at the same axis, but in principle they haven't a dependency, later we will see that it's not true for all positions. Also, $J2$ and $J11$ are at a different axis. Then, bodies or links like $L2$ and $L7$ are links with more than 2 joints, therefore they belong to more than one closed loop.

The following table shows the measurements of each link:

Link	Length (mm)	Link	Length (mm)	Link	Length (mm)
L1	135	L8	57	L5	147
L2'	43	L9	135	L6	43
L3	135	L7"	57	L7'	147
L4	43			L2"	43

Table 4.2: Mk2 Robot: Link lengths

In order to save time for the PCB design, we try to place each encoder for

In order to give the space for the PCBs, the parts that hold them are designed to allow for a unique PCB design. This will save time on their design. In total there will be 4 different designs.

1. For the encoders where are on the top of the base.
2. For the encoder where is on the bottom of the base.
3. For the drivers.
4. For the PCB where there is the power supply management and the microprocessor board connections.

CHAPTER 5

Robot models

5.1 Forward Kinematics: Mantis Robot

In this section we will see the forward kinematics computations for the Mantis robot. That is, the relative position and rotation between the $\{b\}$ frame and the $\{s\}$ frame at an instant according to the joint rotations of the Mantis robot. The following image shows the schematic of this robot.

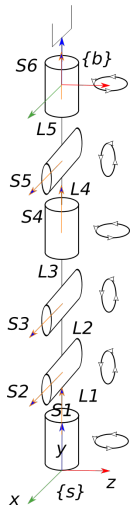


Figure 5.1: Mantis Robot: Screw axis representation

In the home position matrix, we can see the relative position and rotation between the $\{b\}$ frame and the $\{s\}$ frame. Both are shown in the Figure 5.1. As we can see there is not any rotation between both frames, and there is a translation through the $\hat{y}$ axis.

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 660.25 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In order to compute the screw axis, firstly we need to find the angular and linear velocities of each joint applying the following expressions:

$$S_n = \begin{bmatrix} \widehat{s_n} \\ -\widehat{s_n} \times q_n + h\widehat{s_n} \end{bmatrix} = \begin{bmatrix} \widehat{s_n} \\ -\widehat{s_n} \times q_n \end{bmatrix} \tag{5.1}$$

Since all joints are Revolution (R), the pitch $h = 0$. The q_n is a point where it is at the screw axis (S_n) of the joint n. All q_n points have the relative position to the $\{s\}$ frame origin. For example, q_2 is placed at the screw axis (S_2) of joint 2, and the relative position with respect to the $\{s\}$ frame origin is $(0, 173.5, 0)$.

$$\begin{bmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 173.5 & 398.5 & 428 & 628.85 & 660.25 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The unit vector of the direction of the screw axis. Here we can see the axis rotation of each joint.

$$\begin{bmatrix} \hat{s}_1 & \hat{s}_2 & \hat{s}_3 & \hat{s}_4 & \hat{s}_5 & \hat{s}_6 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Therefore, once we have all needed parameters, we can compute the linear and the angular velocities:

$$\begin{bmatrix} \hat{S}_1 & \hat{S}_2 & \hat{S}_3 & \hat{S}_4 & \hat{S}_5 & \hat{S}_6 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -173.5 & -398.5 & 0 & -628.85 & 0 \end{bmatrix}$$

Now the PoE computation:

$$\begin{aligned} e^{[S_n]\theta_n} &= \begin{bmatrix} e^{[\omega_n]\theta_n} & G(\theta_n)v \\ 0 & 1 \end{bmatrix} \\ e^{[\omega_n]\theta_n} &= I + sin\theta_n \left[\hat{\omega_n}\right] - (1 - cos\theta) \left[\hat{\omega_n}\right] \\ G(\theta_n) &= I\theta_n + (1 - cos\theta_n) \left[\omega_n\right] + (\theta_n - sin\theta_n) \end{aligned}$$

And finally, the homogeneous transformation and we can find the relative position and rotation of the body frame with respect to space frame according to the angle rotation of each joint:

$$T(\theta) = e^{[S_1]\theta_1} \cdot e^{[S_2]\theta_2} \cdot e^{[S_3]\theta_3} \cdot e^{[S_4]\theta_4} \cdot e^{[S_5]\theta_5} \cdot e^{[S_6]\theta_6} \cdot M \tag{5.2}$$

To test this computations, a program coded in python is developed. This program will be in the annexes. If we apply the equation above, we can see some examples with different angles at each joint:

If we just rotate the joint 1 by $\pi/2$.

$$\begin{bmatrix}\theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 & \theta_6\end{bmatrix} = \begin{bmatrix}\pi/2 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$$

$$T_{sb} = T_{16} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 660.35 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

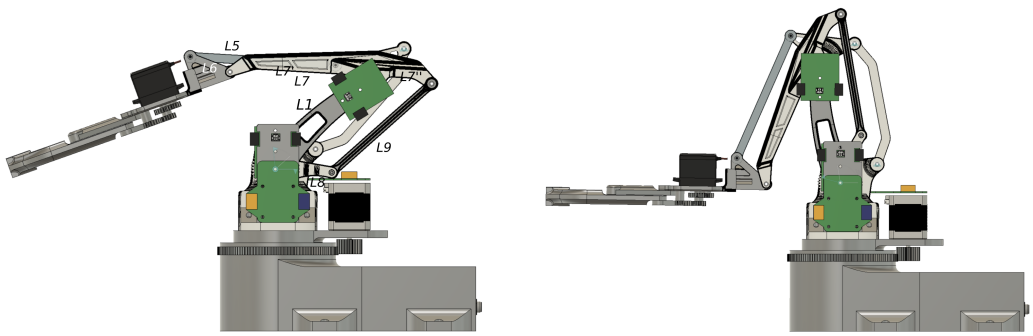
It makes sense because body frame just changes its rotational part, but its linear part is the same.
But we can see an example where there is only changes in the linear part by the following example:

$$\begin{bmatrix}\theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 & \theta_6\end{bmatrix} = \begin{bmatrix}0 & 0 & \pi/2 & 0 & -\pi/2 & 0\end{bmatrix}$$

$$T_{sb} = T_{16} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 430 \\ 0 & 0 & 1 & 230.35 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

5.2 Forward Kinematics: MK2 Plus Robot

The following images show how can be the behaviour of the robot at the loops when we move the both actuators:

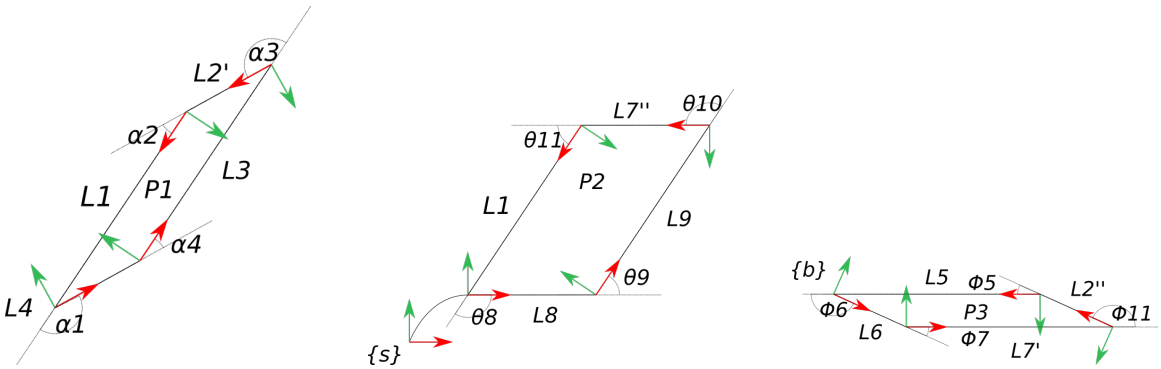


(a) MK2: At home position

(b) MK2: After 69 degrees at θ_8 and α_1

Figure 5.2: MK2: Motion

We could employ the generic methodology to compute the forward kinematics at this part of the robot. But it can be tedious. Instead, as all loops has a parallelogram shape as the Figure 5.3. But the first, we have to decide which home position will have the robot. In the Figure 5.2a, we can see how is the position. We have to remember that the home position is when active joints are chosen as a zero $\theta_8 = \alpha_1 = 0$. There is a main reason to choose as this, in sleep mode (when stepper motors are not energised) the robot can stay at this position. Later we will see it can be posible.



(a) MK2: Frames of the First loop

(b) MK2: Frames of the Second loop

(c) MK2: Frames of the Third loop

Figure 5.3: MK2: Schematics and frames of each loop

We can take an advance the following relations:

$$\begin{cases} \alpha_1 & = \alpha_3 = 180 - \alpha_2 = 180 - \alpha_4 \\ \theta_8 & = \theta_{10} = 180 - \theta_9 = 180 - \theta_{11} \\ \phi_6 & = \phi_{11} = 180 - \phi_5 = 180 - \phi_7 \end{cases}$$

So, how are the relations to parameterise the body frame $\{b\}$. First of all, when the actuator, which moves α_1, is moving L_1 link in the first loop, the L_4 position is not changed and that means it is acts as a ground. L_3 and L_2' follow the movements with the angle rotations as equation above indicates. Then, at the second loop if L_1 is moved by α_1, it is moved at the second loop indeed too. All links are moved with the exception of L_8 who acts as a ground. The third closed loop, that is $\alpha_1 = \phi_6$ and $\alpha_2 = \phi_5$.
Finally, if we move the joint θ_8 that it controls directly the link L_8. It happens a very similar procedure. In this case, the first loop is not moving, however, second and third loop do it. The angle motions at the third will be directly the motion from loop 2. For example, if the actuator moves $\Delta\theta$ the joint 11 will rotate $\phi_{11} + \Delta\theta$

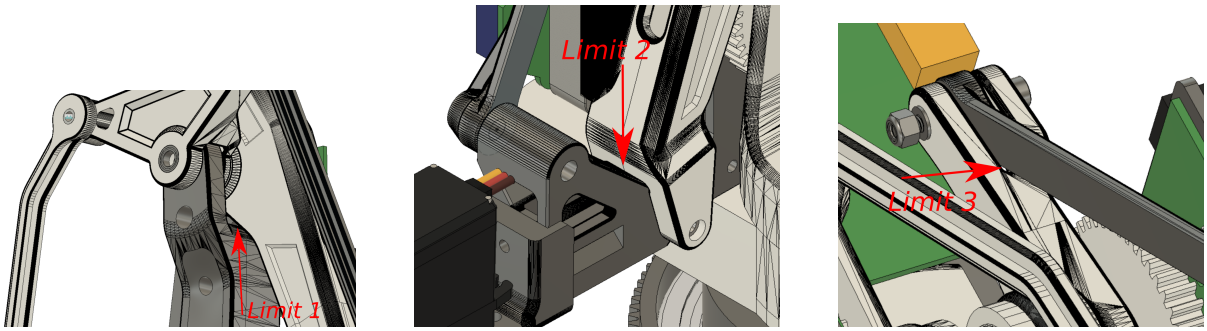
So, if a closed loop follows these equations (example the second loop):

$$\begin{aligned} L_9 cos(\theta_9) + L_7" cos(\theta_9 + \theta_{10}) + L_1 cos(\theta_9 + \theta_{10} + \theta_{11}) + L_8 cos(\theta_9 + \theta_{10} + \theta_{11} + \theta_8) = 0 \\ L_9 sin(\theta_9) + L_7" sin(\theta_9 + \theta_{10}) + L_1 sin(\theta_9 + \theta_{10} + \theta_{11}) + L_8 sin(\theta_9 + \theta_{10} + \theta_{11} + \theta_8) = 0 \\ \theta_9 + \theta_{10} + \theta_{11} + \theta_8 - 360 = 0 \end{aligned}$$

In order to get the $\{b\}$-frame position (x_b, y_b):

$$x_2 = -L_1 cos\alpha_1 \tag{5.3}$$
$$y_2 = -L_1 cos\alpha_1 \tag{5.4}$$
$$\alpha_1 = \alpha_3 = 180 - \alpha_2 = 180 - \alpha_4 \tag{5.5}$$
$$\phi_{11} = 180 - \phi_5 = \alpha_1 \tag{5.6}$$
$$x_b = -L_2 cos(\phi_{11} + \Delta\theta) - L_5 cos(\phi_{11} + \phi_5 + \Delta\theta_8) + x_2 \tag{5.7}$$
$$y_b = -L_2 sin(\phi_{11} + \Delta\theta) - L_5 sin(\phi_{11} + \phi_5 + \Delta\theta_8) + y_2 \tag{5.8}$$

In addition, there are some mechanical constrains that they are from the arm robot structure.



(a) MK2: First constraint (b) MK2: Second constraint (c) MK2: Third constraint

Figure 5.4: MK2: Schematics and frames of each loop

These constrains causes that in some conditions the robot can't move other parts of the robot. In particular, depending on the actuator α_1, the actuator θ_8 will have different motion ranges due to the limits shown at Figure 5.4. They were extracted by observation in the CAD program since we can animate the robot model and we can see its behaviour. These conditions are the following:

1. If $\alpha_1 = 41.5 \Rightarrow \theta_8 = 0$: If the α_1 is equal to 41.5 degrees, θ_8 will always be at zero degrees. That's why these angles are the home position. L_1 tends decrease the angle in sleep mode, but it can do it because of the limit 1. And the actuator who moves L_8 can't move because it's imprison between the $3 : DriverPCB$ and the limit 1.

2. If $41.5 \leq \alpha_1 \leq 110.5 \Rightarrow 0 \leq \theta_8 \leq \alpha_1 + 41.5$: If α_1 is between 41.5 and 110.5 degrees, θ_8 will be able to move between 0 and $\alpha_1 + 45.5$ degrees. The upper range of θ_8 is due to the first limit.

3. If $110.5 \leq \alpha_1 \leq 152.5 \Rightarrow 0 \leq \theta_8 \leq 110.5$: If α_1 is between 110.5 and 152.5 degrees, θ_8 will be able to move between 0 and 110.5 degrees. Here the θ_8 actuator is where it has the widest range of rotation. The upper range of θ_8 is due to the second limit.

4. If $152.5 \leq \alpha_1 \leq 193.5 \Rightarrow \alpha_1 - 152.5 \leq \theta_8 \leq 110.5 - \Delta\alpha_1$: If α_1 is between 152.5 and 193.5 degrees, θ_8 will be able to move between $\alpha_1 - 152.5$ and 110.5 degrees. Now the lower range is due $theta_8$ is due to the third limit.

Part III

Electronics concepts and design

CHAPTER 6

Electronic Design

6.1 System Overview

In the robots described in this work, the main purpose of the electronics is to drive the motors that move its active joints. Its main blocks are, a power supply to provide energy to all the system, a microcontroller that hosts the robot firmware and controls all its elements and a motor control loop composed of motor drivers, stepper motors and encoders. The motor control loop takes step commands from the microcontroller that are sent to the drivers that energise the motors in a fast and efficient way. The motors give movement to the active joints. Although stepper motors provide relative position information, we add absolute encoders to provide feedback to the microcontroller and detect any deviation of the joints from their intended state. High level commands are provided to the robot from a PC using a serial link implemented inside the USB protocol. In the following figure we can see an overview of the system.

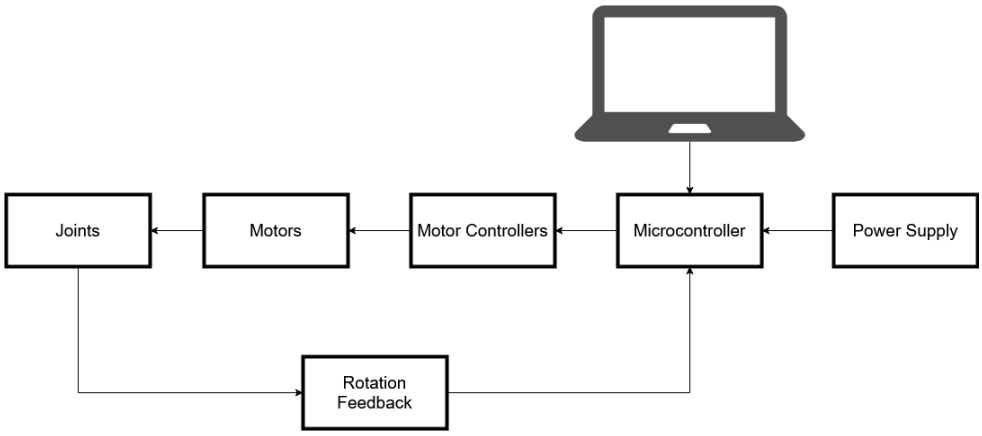


Figure 6.1: System Overview

The electronics have been designed so that they could be used in any robot driven by stepper motors. Due to that, the design of the two considered robots are similar. There will be few differences that will be indicated during the next sections.

One of the goals when designing the circuit has been to reduce the number of wires so that we can use readily available slip rings. By using slip ring devices we can provide continuous rotation of the robot joints improving the robot capabilities. The wiring reduction has been possible by the the use of serial signalling implemented by shift registers connected to SPI peripherals.

6.2 Electronics Design Blocks

In this section, the global design of the robot electronics is described. Both the Mantis robot (Figure 6.2) and the MK2 (Figure 6.3), have similar blocks, but with some differences. In both robots due to the use of slip rings, there was a limitation of the number of cables through the robot path. This maximum was was 12 cables, since it's the maximum number of circuits readily available slip rings can manage.

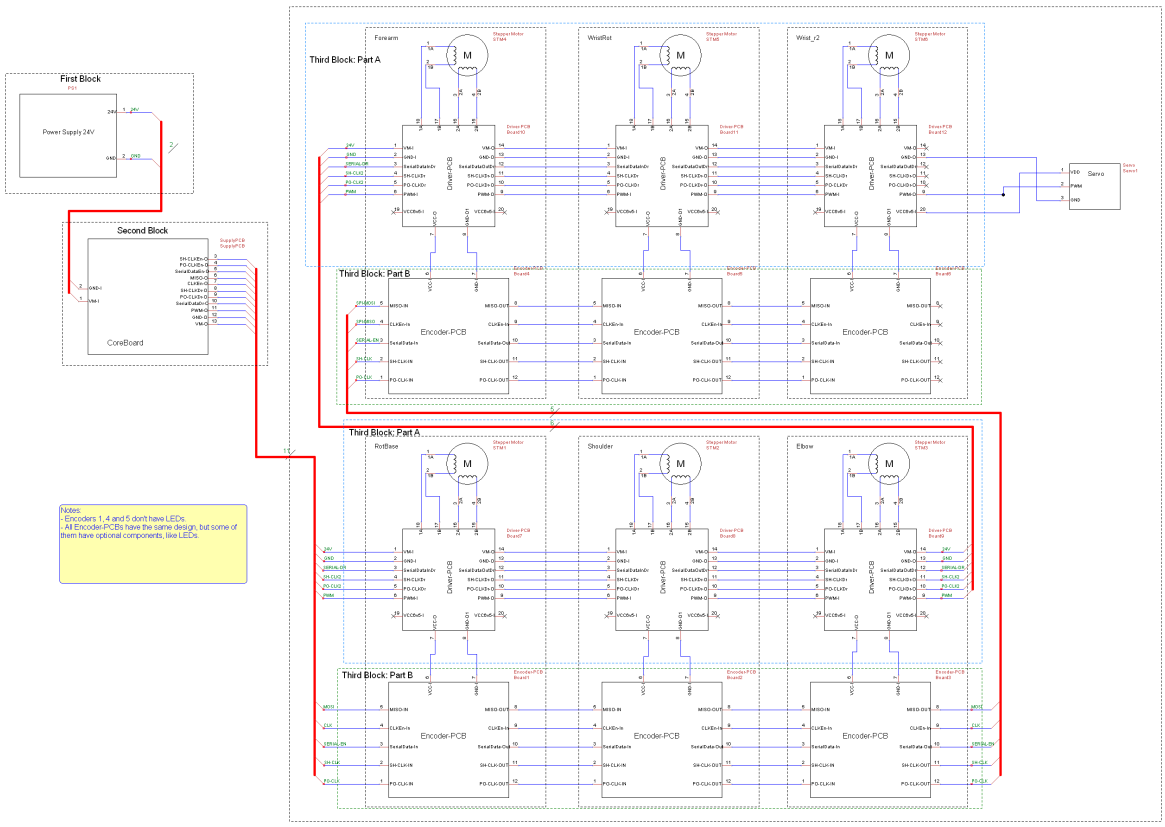


Figure 6.2: Blocks: Mantis Robot

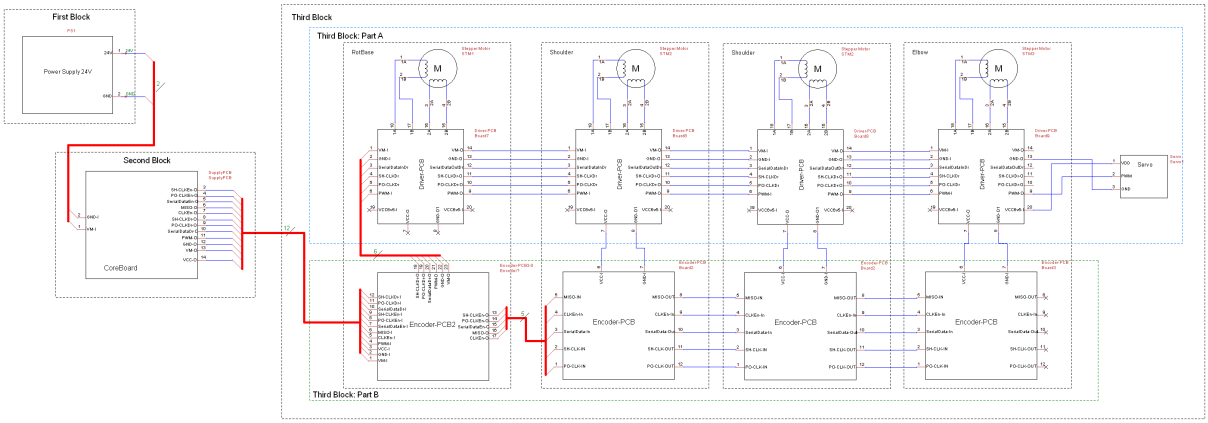


Figure 6.3: Blocks: MK2 Robot

The first block only consists of a 24V power supply that can deliver a maximum of 14.6 amps.

The second block is constituted by the CorePCB board that includes a Nucleo board from STMicroelectronics that hosts the system microcontroller. The CorePCB provide on one hand power the included Nucleo board, and on the other hand for the entire system using two wires for the Mantis robot (VM, GND), and three wires for the MK2 robot (VM, GND and VCC). VM is a 24V line that drives the motors, while VCC is a 3V3 line for the digital elements. In both robots, the number of digital signal cables will be nine in total. The first, eight cables carry three SPI links that will be described later. Finally, the ninth digital line carries a PWM signal to control the end actuator gripper servo.

6.3 Electronic design: Third block

6.3.1 Signal management

The main purpose of the electronics is, for each active joint motor, control its operation and give feerback of its operation.

If all lines were connected the microcontroller in parallel, the wiring will exceed any slip ring capabilities.

Each driver needs 8 signal wires and 3 power wires, while the encoder needs 3 cables for SPI communication and two more cables for power. The servo needs 2 cables for power, and one for the PWM signal. In the following table we see a summary of the number of wires per device:

Device	Type of signal	Number of wires
Driver	Power	3
	Digital	8
Encoder	Power	2
	Digital	3
Servo	Power	2
	Digital	1

Table 6.1: Number of cables for each electronic device

The wirings are limited by the parts of the robot that have a slip ring. In the case of the Mantis robot, it has two slip rings, the first between the forearm and the first part of the wrist, and the second between the second part of the wrist and the servo. In the case of the MK2 robot, it has an only slip ring that is placed between the rotational base and the shoulder.

Next we will explore the limits that are specific for each considered robot.

For the Mantis robot in the second slip ring only 3 cables are needed for the servo, therefore there is not problem here as there are less than 12 cables. While in the first slip ring you would need cables for 2 drivers, 2 encoders and the servo. Therefore, the number of cables would be 19 for the drivers, 6 for the encoders, and 3 for the servo. In other words, a total of 28 cables would be needed between the forearm and the first part of the wrist. It clearly exceeds the allowed limit.

For the MK2 robot in the compromised area we would need cables for 3 drivers, 3 encoders and a servo. A total of 27 for the drivers, 7 for the encoders and 3 for the servo. That is, 37 cables in total.

6.3.2 Number of cables solution

The solution to this problem is given by three actions implemented in the electronic design:

1. The first action is using local regulators to obtain the supplies that are different that the main 24V supply. That is, instead of the CorePCB board providing the power that each device needs. The CorePCB only sends 24V to the rest of the boards, and then each board implements regulators to obtain other needed power rails.

2. The use of shift registers with parallel output. This is the key component that allows us to reduce most of the wiring requirements. Serial shift registers convert a serial stream in multiple parallel outputs just using one serial input and two clock signals (SH-CLK and PO-CLK). As shift registers are daisy chained, the number of signals don't depends on the number of outputs required. These devices have a similar operation to SPI communication, therefore it allows us to connect them directly to the SPI peripheral of the microcontroller.

3. Finally, we serialise all the boards that have the same functionality. That is, all driver boards are serialised and all encoder boards are also serialised. That way the system has a wiring requirement that is independent on the number of boards.

Therefore, the cable management is designed as follows:

- The first part (PartA in the figures), is constituted by the DriverPCB boards , one for each motor, that are placed in series. As both robots feature different number of motors, they have a different number of driver boards. At the input connections there are signals to power supply, a data signal (SerialDataInDr), two clock signals (SH-CLKDr and PO-CLKDr) and gripper PWM signal. In the output connections there are three different paths, the first is to supply the motor windings (1A, 1B, 2A and 2B), the second path is to supply the EncoderPCB with 3.3V. The third path is to redirect all input signals to the next DriverPCB, with the exception of the last one because the only three wires are needed for the servo.

- And finally the second part of the third block is formed by the EncoderPCB boards that are also placed in series. The main goal for this block is provide a joints positions feedback. As we can see in the figures, the Mantis robot has 2 more boards than the MK2 robot. However, there is one more difference. A board has to be placed at base rotation of the MK2 robot. Due to the structure of the robot, a second EncoderPCB design has to be performed. Therefore there are two designs of the feedback.
  In the first design, the input connections comes the power phase from the DriverPCB. Whereas the digital signals are provided by the CorePCB board. On the output cables it redirects all digital signals to the next EncoderPCB.
  The second design, Encoder-PCB2, which is specific to the MK2 robot, is made to receive all the signals coming from CorePCB. Their output connections are distributed, on the one hand to the DriverPCB board, and on the other hand to the first design of the EncoderPCB board. The power supply of this board is also provided by CorePCB since it has more space for circuitry of its power phase.

Finally, with this design we have reduced the wiring requirements in the Mantis robot from 28 wires to 11 wires. And , in the MK2 robot, it ahs been possible to reduce the requirements from 37 wires to 11 wires.

Therefore 3 SPI communications have been implemented:

1. The first one for serializing the Encoder PPB signals with SerialDataEn, SH-CLKEn and PO-CLKEn signals.

2. The second one for the encoder chip communications with MISO and CLKEn signals.

3. Finally, the third one for the DriverPCB signal serialization with SerialDataDr, SH-CLKDr and PO-CLKDr signals.

The signals that are called as SerialData act as MOSI signals, and the PO-CLK signals act as CSn. In the following schematics we will see how CSn is implemented for the second SPI communication.

6.3.3 CorePCB

This PCB has two functions. The first one is to connect to the board that hosts the microcontroller that will manage all the intelligent part of the robot's behaviour. And the

second function is to provide the energy for the rest of the robot components. From the input voltage is 24V line obtained from an external power supply.

The board that hosts the microcontroller is the Nucleo 411RE model. It has been selected because it has a maximum CPU clock frequency of 100 MHz and 5 SPI ports. Other suitable models like the 303RE, with maximum frequency of 72 MHz and 4 SPI ports, but, although they have the same price, they don't provide better performance.

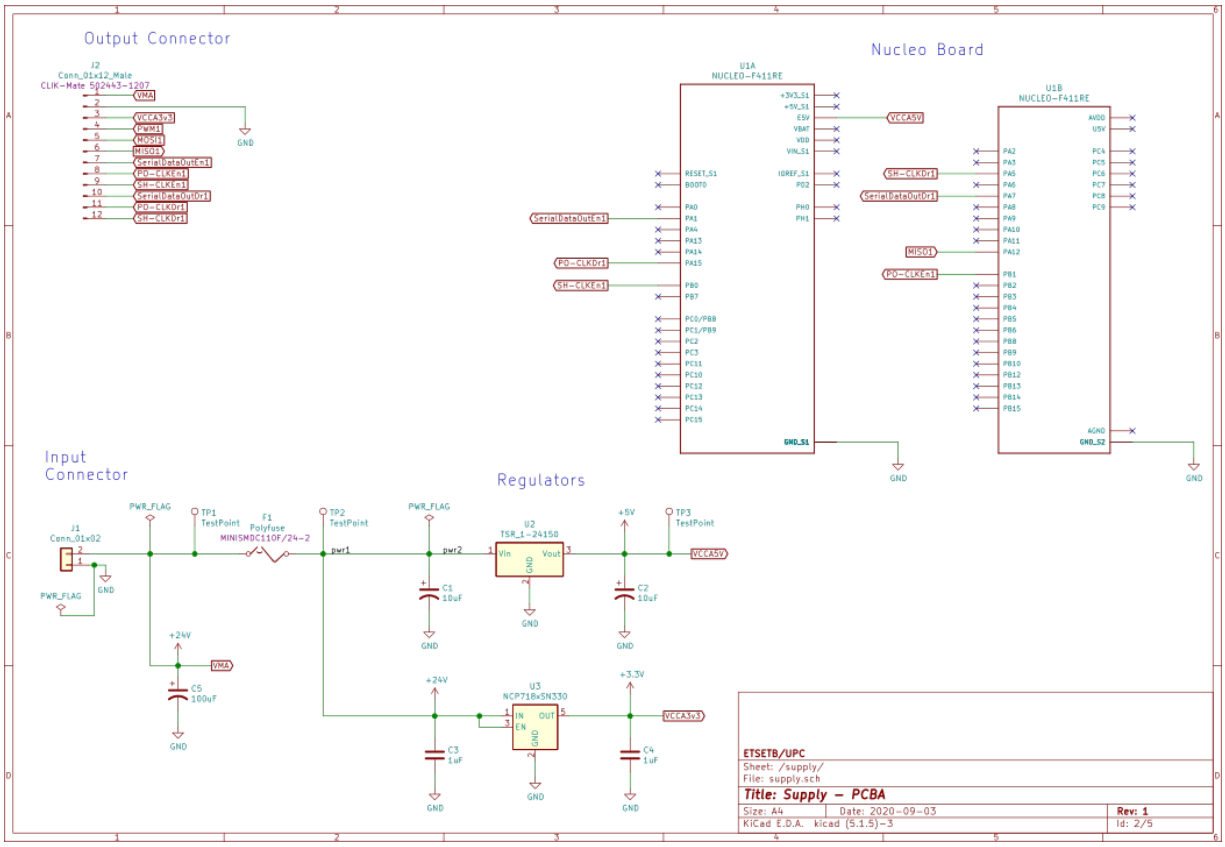


Figure 6.4: Electronic Design: CorePCB

The power section generates three power rails, the first 5V rail powers the Nucleo board. The second 3V3 rail powers EncoderPCB that is in the base, and the third rail just redirects the 24V to the rest of the robot. The second 3V3 rail is only needed for the MK2 robot.

The Nucleo board[3] is powered trough its E5V input. This option enables powering the board regardless of its connection to a PC. To improve the efficiency, a switch mode regulator has been used.

The 3V3 rail that is exclusive for the MK2 robot also uses a switch mode regulator to improve efficiency.

To protect against short failures in the digital elements, a polyfuse with 1A holding current and 2A triggering current has been included previous to the generation of the 5V and 3V3 rails.

6.3.4 DriverPCB

This board is designed to perform 3 functions. The first one is to drive an stepper motor, the second one is to pass the signals to the next DriverPCB board, and , finally, the third function is to supply power to an EncoderPCB board. The last board in the chain has an additional function for driving the gripper.

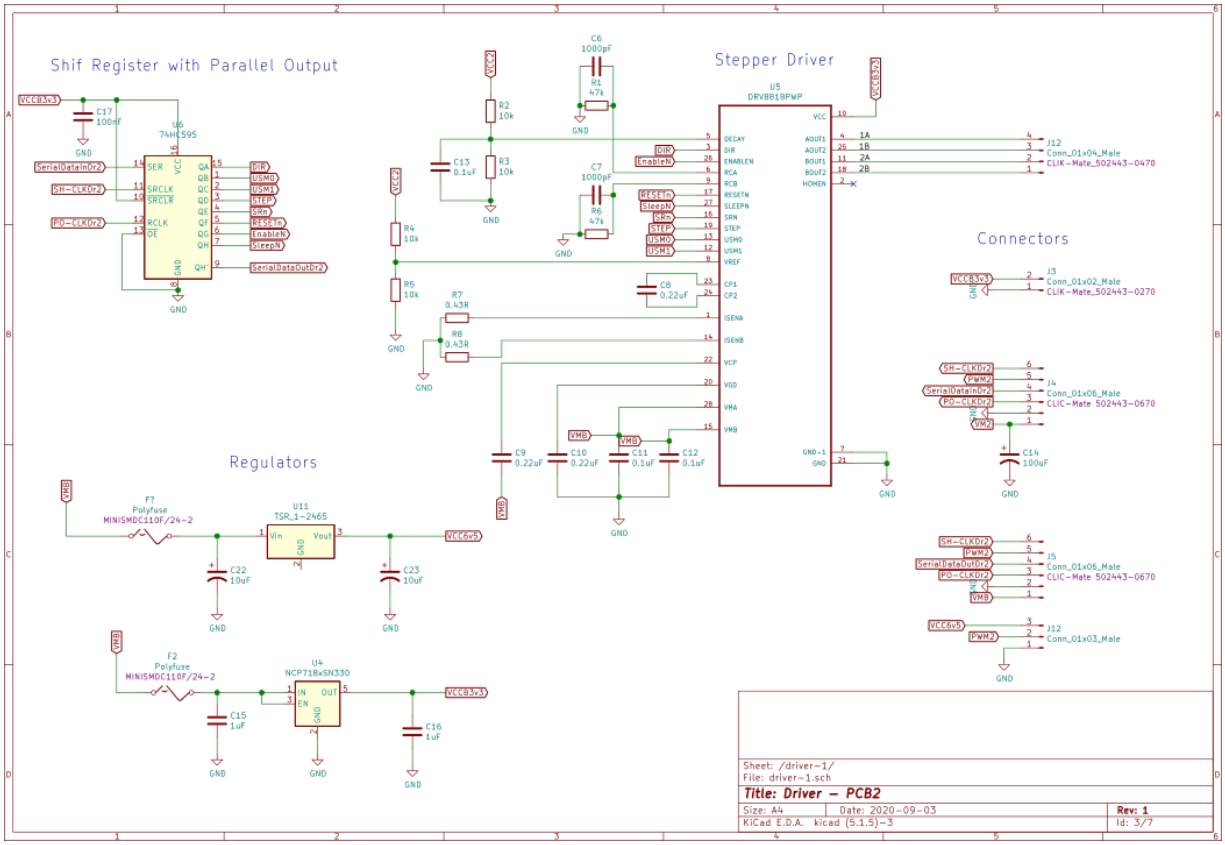


Figure 6.5: Electronic Design: DriverPCB

The DRV8818 stepper driver is used to drive the stepper motor coils. The value of the resistors connected to the ISENA and ISENB set the current to be delivered to the motors windings [4]. The values shown in Figure 6.5 are configured for steppers with a current of 1A per coil, that value is adequate for the MK2 robot, but the driver is capable to operate at the 2A value required for the Mantis robot. The rest of the components have followed the manufacturer's recommendations, except for the digital input signals that are connected to a shift register with parallel output. The shift registers are compatible with the microcontroller SPI ports. So we will use one of the available SPI peripherals to drive the motor board shift registers. The digital signals obtained from the shift registers control all the motor operation including microstepping, energy saving mode, and the direction of rotation of the motor.

The 6 pin J4 connector receives the digital signals from the microcontroller or the previous driver board. Those signals are daisy chained to the next board using the J5 connector.

The 3V3 rail needed for the digital elements is obtained from the 24V rail by using a linear regulator. AS the required current is small, there is no need to use a switch mode

regulator. The 3V3 rail is provided to the encoder board of the same joint board to reduce the number of regulators needed.

Finally, the 3-pin connector J12 is used only for the last board in the chain, since it is the only one that has a direct connection to the servo. The three pins provide power, ground and the positioning signal for the servo. The 6V5 rail that powers the servo[5] is generated on this last board by the use of a regulator connected to the 24V rail.

Both power rails feature a polyswitch to protect against current hazards

6.3.5 EncoderPCB

The main purpose for this boards is to provide a joint position feedback to the microcontroller via magnetic encoders. The chosen encoder is the AS5047D [6] that features a SPI port communication and has an uncertainly of 0.1 degrees, this error is small enough compared to the 1.8 degrees of a full step of the motor.

For both designs, CSn signal of the encoder is connected to a shift register with parallel output. But, the MISO and clock signals are provided by the input connector of the board. As the MOSI pin is connected to the rail 3V3, the register pointed of the encoder will be the reading angle.

Two different EncoderPCB designs are shown on figures 6.6 and 6.7. The first design is used for both robots, whereas the second design is specified to be placed in the base rotation of the MK2 robot.

The first design perform 3 functions. The first one is to provide a joint position, the second one is to pass the digital signals to the next EncoderPCB board, and the third one to provide a visual signals using RGB LEDs, which is connected to the digital outputs of the shift register with parallel output.

The 5 pin J6 connector receives the digital signals from the microcontroller or the previous driver board. Those signals are daisy chained to the next board using the J7 connector. The 3V3 rail needed for the digital components is obtained from the DriverPCB.

The second design has 2 functions. The first one is to provide the base rotational joint position of the MK2 robot, and the second one is to pass all signals from CorePCB to the EncoderPCBto the next EncoderPCB board. The 3V3 rail needed for the digital devices is obtained from the CorePCB.

In the next section we will see more details about how this kind of encoder works.

6.4 Magnetic Encoder

The main goal for an encoder that is used in this design is having a feedback from rotation positions. These electronic devices transform angle rotations into a codification, commonly in bytes.

There are two types of encoders: the **incremental** and the **absolute** encoders. Absolute encoders have the advantage that don't need the system to go to a home position on start-up.

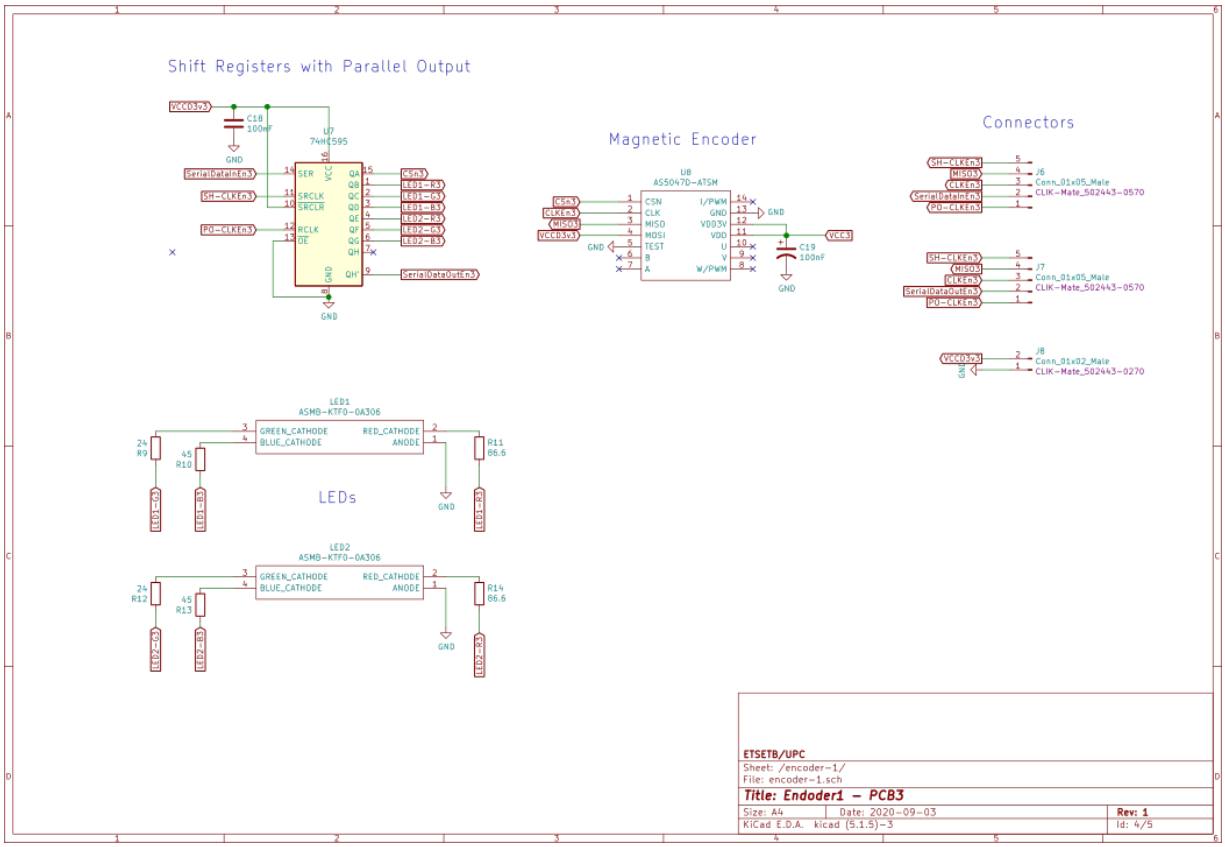


Figure 6.6: Electronic Design: EncoderPCB

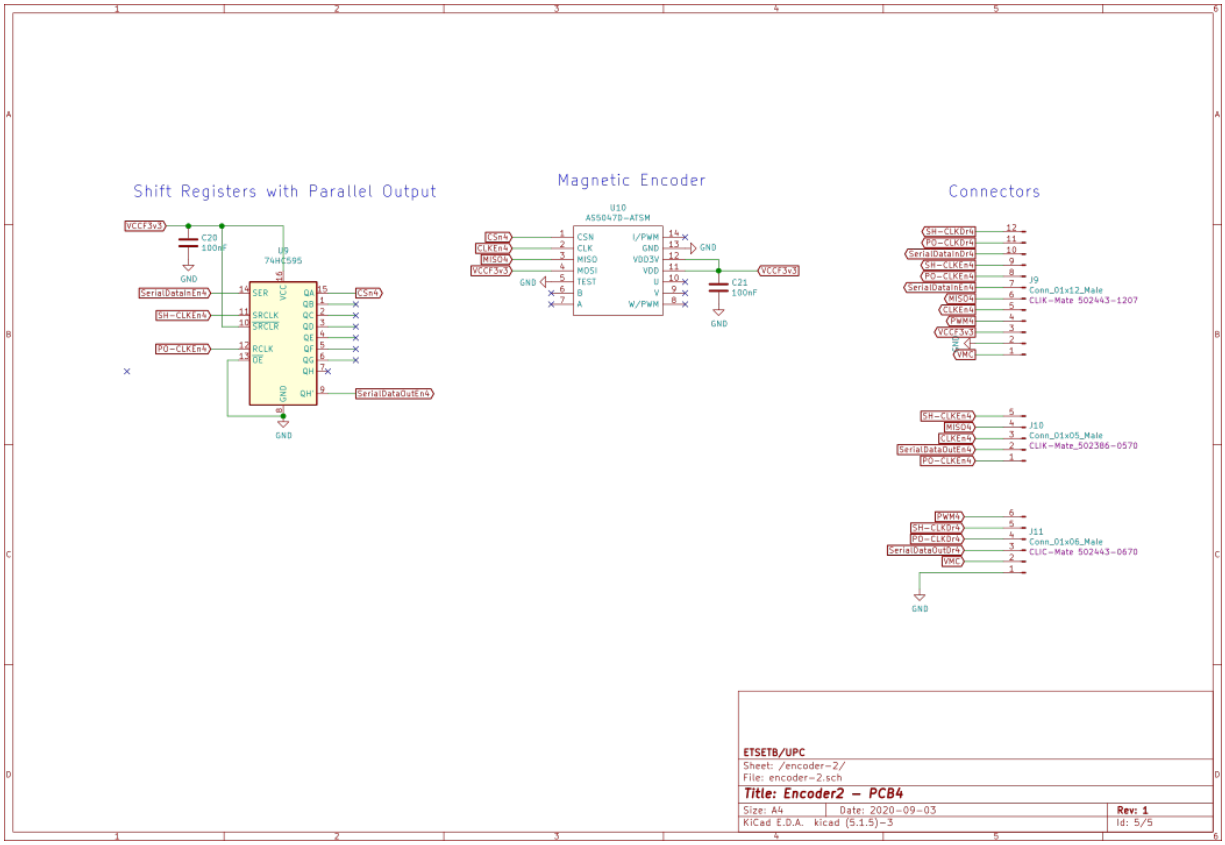
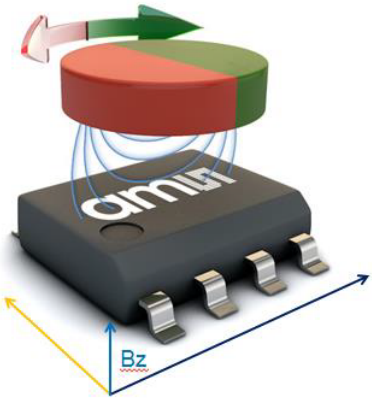


Figure 6.7: Electronic Design: EncoderPCB2

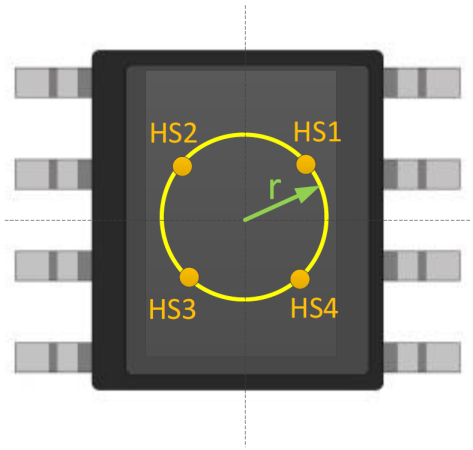
We can avoid using limit switches if we implement absolute encoders in the system. The major disadvantage from absolute encoders is its cost, since they are typically expensive. However, cureently we can find absolute magnetic encoders based on hall sensors whose cost is relatively cheap.

6.4.1 Magnetic Encoders: Introduction

Rotatory magnetic sensors from **ams** offers several **Integrated Circuit (IC)** absolute encoders where the magnetic flux density is measured. This magnetic field comes from a **magnet**, which is diametrically magnetised, and it is placed on the IC surface commonly placed at the center of XY plane. Apart from that, there is an air gap between the magnet and IC, as Figure 6.8a shows. According to [7], the only measured component of magnetic field is B_z, that is, B_x and B_y are ignored.



(a) On-axis magnetic position sensor IC + magnet



(b) Example Hall sensor locations and measurement radius

Figure 6.8: Rotatory Magnetic Encoder

In the Figure 6.8b, there is an example where these hall sensors can be located. When the magnets rotates over the chip, a sinusoidal signal is generated by each sensor [7], and because of this rotatory position, an angle α can be calculated as the following expressions are described:

$$Signal_1 = V_{HS1} + V_{HS2} - V_{HS3} - V_{HS4} \tag{6.1}$$
$$Signal_2 = V_{HS1} - V_{HS2} - V_{HS3} + V_{HS4} \tag{6.2}$$
$$\alpha = ATAN2(Signal_1, Signal_2) \tag{6.3}$$

The combination of summing and subtracting on the four signals from hall sensors generates two sinusoidal signals with a difference at their phase. Making the ATAN2 operation, the angle between 0 and 360 degrees. *More details in [7].*

In order to compute an angle measurement, there are **static** and **dynamic errors** that we have to keep in mind: . In magnetic encoder context, statics error refers to:

- Mechanical misalignment between the magnet and IC surface produces Integral Non Linearity (INL). INL means the difference between the ideal and the real measured angle.
- The magnet material, its shape, thickness, grades and magnetisation errors.

Whereas dynamic error are produced by two components:

- Angular velocities and accelerations.
- Delay propagation.

There are some recommendations to avoid these static errors. However, the most important to avoid the dynamic errors is not exceed the maximum rotational speed that it is indicated by the datasheet.

6.4.2 Magnetic Encoders: Static Errors

The most important is the shape of magnet, and cylinder magnets are recommended because of the sinusoidal magnetic field they generate.

In addition, there is a relationship between the diameter of the magnet and the horizontal/vertical misalignments. A larger diameter has wider horizontal misalignment tolerance, whereas a small diameter has better signal/noise ratio and bigger air gaps. The more misalignment, the more Integral Non Linearity (INL) we will get. Mechanical errors are more significant than electrical ones.

The materials which are recommended are Neodymium Iron Boron (NdFeB) and Samarium Cobalt (SmCo). The first one is easier to find in the market. The recommended magnetisation grade of NdFeB is labelled as N35. Then there is another letter behind the label grade, such as N35H and N35SH. The main difference is the operating temperature. N35H and N35SH works up to 120 ºC and 150 ºC respectively. However, both work properly.

There is another agreement between the diameter and the thickness of the magnet. According to [7], the best rate Height/Diameter is 0.42. For example, if its height is 2.5mm and its diameter is 6mm, the rate will be $0.4166 \simeq 0.42$.

Other error from the magnet could be about its magnetisation. Magnetisation errors means the magnet is not symmetrically polarised.

Finally, when the mounting magnet it is recommended to avoid over other magnetic materials. The main reason is because the magnetic field produced by magnet is disturbed. Thus, this sinusoidal shape is changed. It can be mounted closed to other magnetic material but with a non-magnetic piece among them.

6.4.3 Magnetic Encoders: Dyanamic Errors

Dynamic errors appear due to the propagation delay resulting from the signal processing needed to obtain the angle. This propagation delay is depended on the encoder model.

But instead of using an external hardware to compensate these errors, the encoders we selected perform the compensation inside the encoder IC, since theoretically the propagation delay is much shorter. This technique is called Dynamic Angle Error Compensation (DAEC). They don't specify how works this technique, but they provide us the following mathematical expressions:

$$DAE = PD(s) \cdot \frac{Speed(rpm) \cdot 360deg}{60s}$$
$$DAE_{ACC} = \frac{acceleration(rad/s^2)}{2} \cdot (PD(s)^2 + PD(s) \cdot \Delta_{speed})$$

The fist equation describes the DAE when an constant angular speed is produced, whereas the second one indicates the DAE due to a constant angular acceleration.

More information in [8]

6.4.4 Magnetic Encoders: Communication Interface

The encoders are available Serial Peripheral Interface (SPI) and Inter-Integrated Circuit (I2C). As we want some encoders of the same type, we can not use I2C due to they are addressed with the same identifying. Therefore we choose SPI communication.

It is a **16 bit** SPI as slave device, and the module is active as soon as CS is pulled low. These encoders are configured as **Mode 1 (CPOL = 0, CPHA = 1)**, that is, the clock starts as low value and the lecture via MISO is done at the first falling edge.

Two modes can be used for an angle reading:

- 4-Wire Mode
- 3-Wire Mode

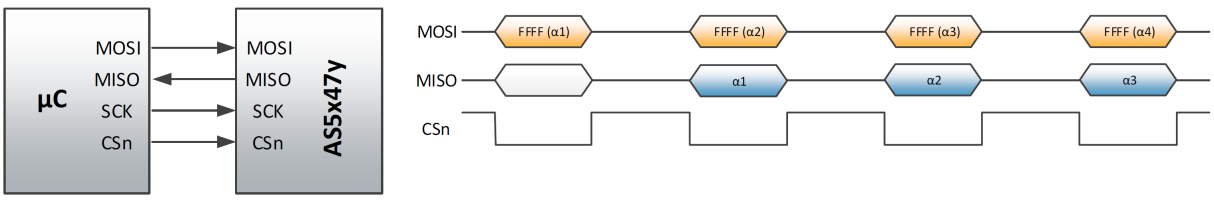


Figure 6.9: 4-Wire Mode

4-Wire Mode means that all SPI connections are used (MISO, MOSI, CLK and CS), see Figure 6.9. The module receives a read request on 0x3FFF register including the parity and read bit (Write bit = 0 and Read = 1) on the most significant bits. Therefore, the sensor receives a data packet with a 0xFFFF value.

As the diagrams show, there is a full-duplex communication but the angle reading comes from the previous 16 bits communication. For example, in the Figure 6.9 the angle

reading α_1 is sent after the request $FFFF(\alpha_1)$. So, the delay is at least the timing of 16 bits between the lecture request and the lecture sending from the sensor to the micro.

Nevertheless, for a non continuous requesting it is better to send a "No Operation" register (0x0000) instead of the $FFFF(\alpha_2)$ after $FFFF(\alpha_1)$. The reason is to avoid that the micro receives the previous requested angle instead of the current angle. Another way to avoid via software ignoring the first 16 bits.

In **3-Wire Mode** the MOSI pin can left open at the microcontroller, whereas at the sensors side the MOSI pin can be connected to VDD thorugh a pull-up resistor. (Figure 6.10.

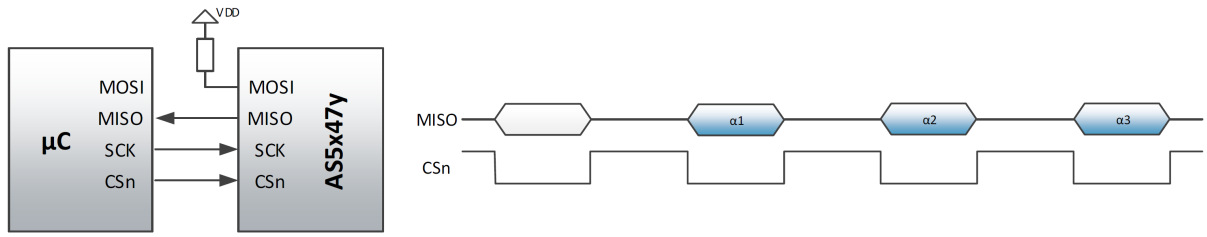


Figure 6.10: 3-Wire Mode

In **multislave configuration** (Figure 6.11), we can implement both the 4 and 3 wire modes but we use different CSn signals to communicate with each encoder.

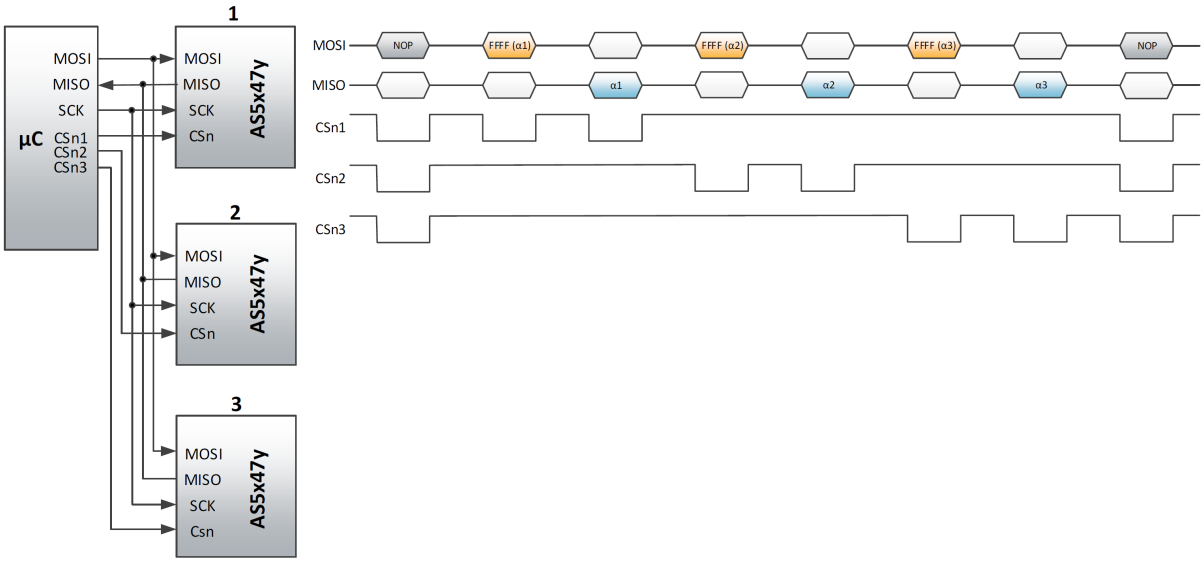


Figure 6.11: Multislave with 4-Mode configuration

More information in [9, 6]

CHAPTER 7

Prototyping and Results

7.1 Prototype: SH-PO + Encoder + Driver

The purpose this prototyping stage is to verify that the encoder and driver are properly controlled by the microcontroller SPI peritheral. The driver and encoder inputs are connected to the outputs of shift registers so that they can be set using a SPI serial interface. A magnet is placed on the top of the encoder surface that is aligned through a platform that holds it. Also, we have put a voltage divider on the encoder's MISO output to The driver and encoder inputs are connected to the outputs of shift registers so that they can be set using a SPI serial interface. The firmware for the prototype operation will be found in the annexes.

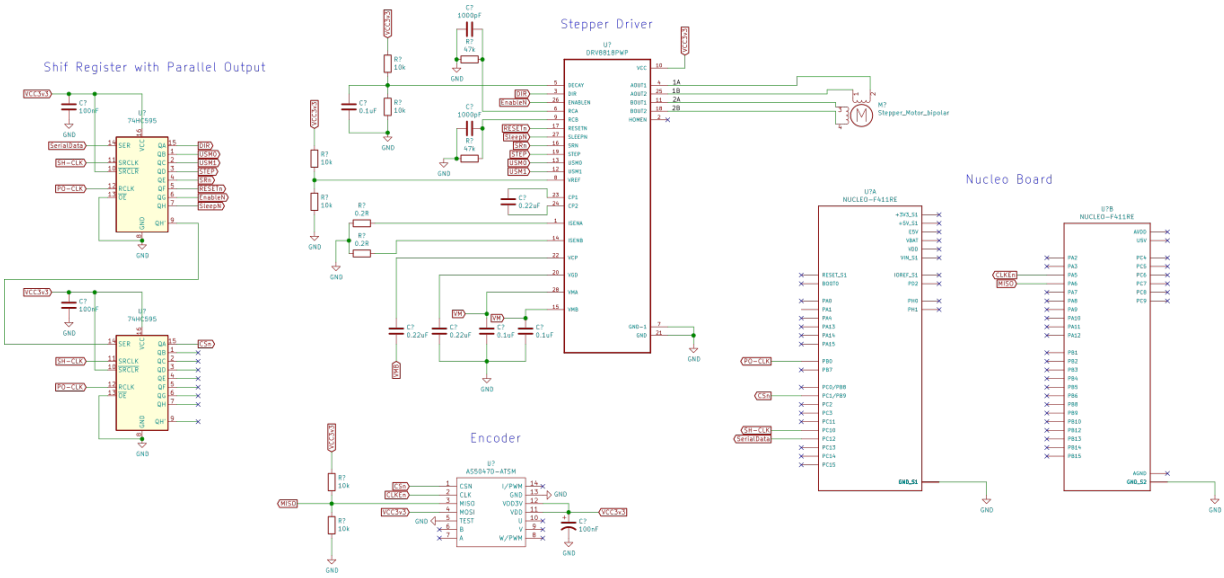


Figure 7.1: Prototyping: SH-PO + Encoder + Driver Schematic

By observation, we can see that the angle rotations correspond to the rotations of the magnet. Also, if we leave the magnet completely still, the read angle incertainty is limited

to 0.2 degrees.

In the following images we can verify that the encoder and driver work satisfactorily. In Figure 7.2, we have the CSn, CLK and MISO signals, which are green, yellow and blue respectively.

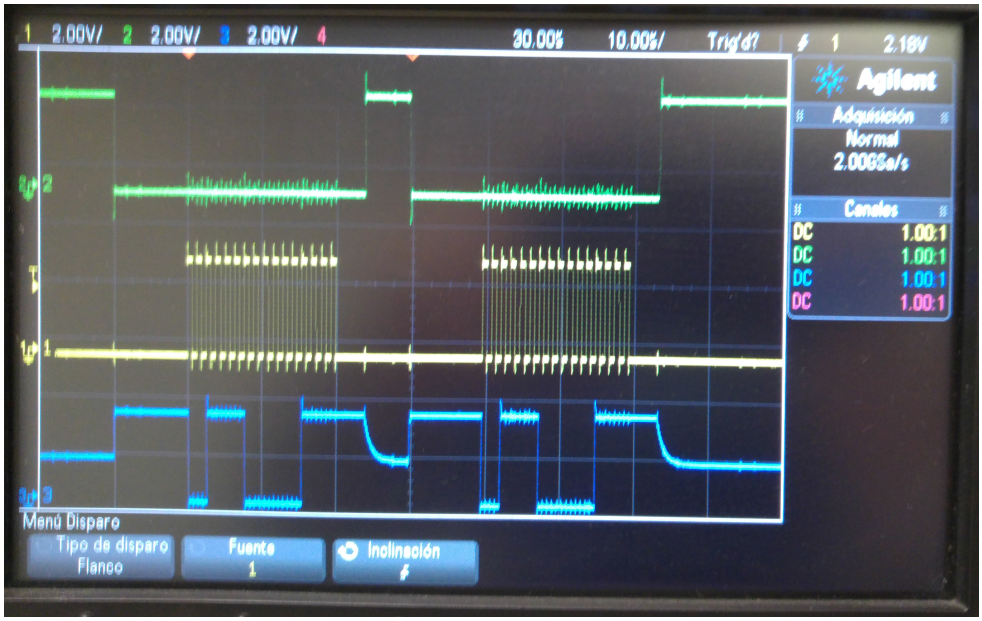


Figure 7.2: Prototyping: MISO, CSn and Clock from Encoder

In this figure we can see the voltages on one of the motor windings. The pink signal is the potential differential between the input and the output of the motor winding.

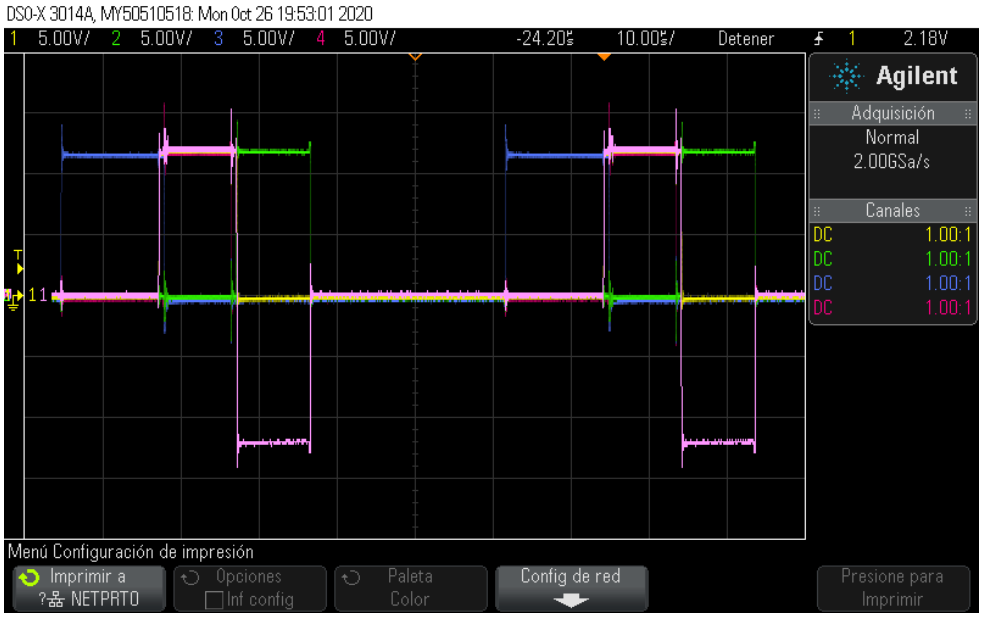


Figure 7.3: Prototyping: Winding Energy

To check that there is no loss of steps, we have put a gear with a signal on the motor shaft. If the gear signal is still in the same place after it has moved several full turns, there

is no loss of steps. In this case the result has been satisfactory.

In addition, the program has been prepared to control the digital input signals of the driver through the serial port (USB) between the Nucleo board and a computer with a keyboard.
The microstepping that the drivers provide has also been successfully tested.

Also, we have increased the stepping rate until we can see that there are step losses. The approximate limit is one stepping signal for every 8ms.

7.2 Prototype: Two Encoders

This test has only been realised to verify that two encoders can work together in the same SPI bus. The result shows that it works without any problem.

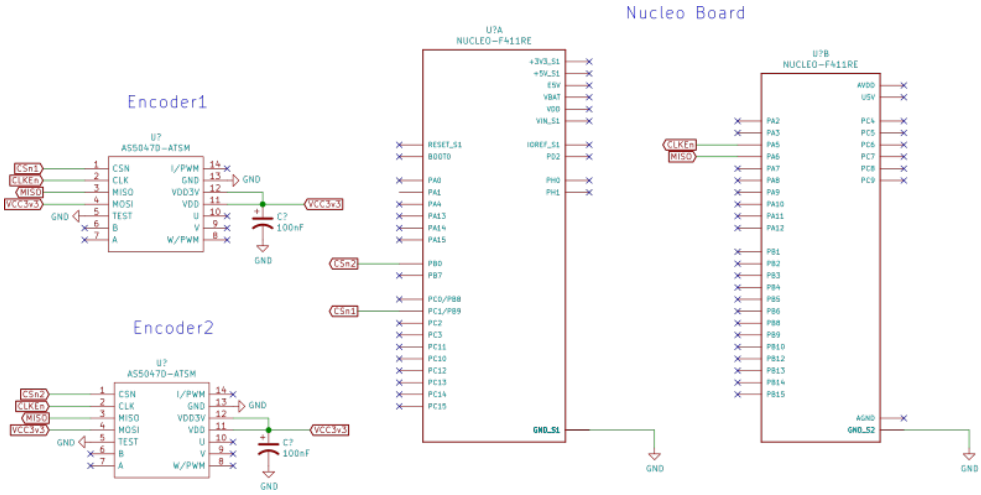


Figure 7.4: Prototyping: Two Encoders Schematic

Part IV

Conclusions and Future Work

CHAPTER 8

Conclusions and Future Work

The main objective of this project has been the modification and assembly of a robotic arm using 3d printing. The project has worked with two robots. Firstly, the Mantis robot was chosen, but due to the COVID pandemic in the end it was decided to implement the MK2 robot. In this second robot, several modifications have been made to adapt the electronics designs that have been previously made for the Mantis robot.

For the mechanical part of the robot, we have made an analysis of how each robot is built, and we have calculated its Forward Kinematics equations. As the first robot features only open chains and the second uses closed chains, the analysis covers mots arm topologies. We have seen that, for open chains there are methods such as PoE which offers a closed solution. In open chains, however, the methodology depends on the specific case of the robot. In our case for the MK2 it was needed to develop a specific calculation method.

For the electronic part of the robot, an electronic design has been made for both robots. They have similar designs but with small differences. These differences are forced by the mechanical structure of the robot. The objective of its electronic circuit is to run the stepper motors that move the joints of the robot, and to have a closed loop feedback to the microcontroller via magnetic encoders. The use of slip rings to achieve continuous rotations limit the number of wires in the cabling. In the design for this project, it has been possible to make the cabling independent on the number of controlled motors. A prototyping has been done to verify the circuit operation including the motor communication and the encoder feedback. The cable minimization has been possible by serialising the data using SPI communication. Programming the Nucleo board that hosts the robot microcontroller with the MBED framework has been very successful, due to the abstraction layer that it provides.

Although only one robot has been manufactured, the bill of materials has been obtained for both robots. Some mechanical components were not available from the UPC providers. That has complicated the sourcing of the needed materials. To minimise the sourcing problems, the component availability has been considered for the MK2 modifications.

Finally, for this project we have learned to use CAD programs for the design and

modifications of 3D objects and assemblies. We have learned to use G-Code generator programs, and finally we have learned to use the 3D printer of the FAB lab, in the electronics labs of the ETSETB.

The following list shows some future work that could be performed in other projects:

- Design and assembly the different PCBs for the circuits that are designed and tested during this work. This point will be done by the author.
- Fabrication of the Mantis robot.
- Make the Inverse Kinematics and the Forward and Inverse Dynamics for both Mantis and Mk2 robots.
- Use the encoder circuits feerback to test the operation of the stepper motors under load.
- Add an smart loop control, like PID, to optimize the robot movements.
- Make use of the multithreading capabilities offered by the MBED real time operating system.
- Interface the robot with Robot Operating System (ROS), since it's a framework that is designed for robotics applications.

Bibliography

[1] Kevin M. Lynch and Frank C. Park. *Modern Robotics Mechanics, Planning, and Control*. Cambridge University Press, 2017. http://hades.mech.northwestern.edu/index.php/Modern_Robotics.

[2] Wikipedia. Kinematics — Wikipedia, the free encyclopedia, 2020. [Online; accessed 20-November-2020].

[3] STMicroelectronics. *STM32 Nucleo-64 boards (MB1136)*, 2020. Rev14, https://www.st.com/resource/en/user_manual/dm00105823-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf.

[4] Texas Instruments. *DRV8818 StepperMotor Controller IC*, 2016. RevE, https://www.ti.com/lit/ds/symlink/drv8818.pdf?ts=1606068228516&ref_url=https%253A%252F%252Fwww.google.com.ua%252F.

[5] TowerPro. *MG995*. https://www.towerpro.com.tw/product/mg995/.

[6] ams. *AS5047D*, 2016. v1.07, https://ams.com/documents/20143/36005/AS5047D_DS000394_2-00.pdf.

[7] ams. *AN5000 Rotary Magnetic Position Sensors*, 2016. v1.04, https://ams.com/documents/20143/36005/AnglePositionOnAxis_AN000271_2-00.pdf.

[8] ams. *AS5x47y DAEC*, 2015. v1.02, https://ams.com/documents/20143/36005/AS5x47y_AN000687_1-00.pdf.

[9] ams. *AS5x47y Interfaces*, 2016. v1.01, https://ams.com/documents/20143/36005/AS5x47y_AN000298_1-00.pdf.

Acronyms

2D 2 Dimensions. 5

3D 3 Dimensions. 5

C-space Configuration space. 6

CAD Computer-aided design. 8

CS Chip Select. 43

D-H Parameters Denavit-Hartenberg Parameters. 13

DAE Dynamic Angle Error. 43

DAEC Dynamic Angle Error Compensation. 43

DOF Degrees of Freedom. 5, 6, 8, 18, 23, 24

I2C Inter-Integrated Circuit. 43

IC Integrated Circuit. 41, 43

INL Integral Non Linearity. 42

MISO Master Input Slave Output. 43

MOSI Master Output Slave Input. 43, 44

NdFeB Neodymium Iron Boron. 42

PCB Printed Circuit Board. 23, 24

PoE Product of Exponentials. 11, 13, 15

SmCo Samarium Cobalt. 42

SPI Serial Peripheral Interface. 43

STL Standard Triangle Language. 8