POLITECNICO
MILANO 1863

SCHOOL OF INDUSTRIAL AND INFORMATION ENGINEERING

Master's Degree Course in Mechanical Engineering

Experimental Investigation of an Adaptive Sensorless Open-loop Strategy of Mechanical System with Stepper Motors

supervisor: prof. Andrea ZANCHETTIN

co-rapporteur: ing. Renzo VILLA

Giuseppe GODINO
864377

Anno Accademico 2017/2018

Politecnico di Milano:
www.polimi.it

School of Industrial and Information Engineering:
www.ingindinf.polimi.it

Acknowledgements

Al termine di questo impegnativo lavoro di ricerca, il quale ha decisamente messo alla prova le mie capacità intellettuali ma anche personali. Vorrei spendere due parole di ringraziamento nei confronti di tutte le persone che mi hanno sostenuto e aiutato durante questo percorso.

Prima di tutto, vorrei ringraziare il prof. Zanchettin, il relatore di questa tesi di laurea, per la sua disponibilità e il prezioso supporto fornitomi, non solo durante lo svolgimento del lavoro, ma anche durante tutto il periodo di stesura. Vorrei ringraziare anche il co-relatore ing. Villa e l'ing. Previtali per i preziosi consigli e gli strumenti fornitomi, i quali si sono rivelati utili ad intraprendere la strada giusta verso il compimento della mia tesi.

Un caloroso ringraziamento va a mia madre e mio padre che, con il loro prezioso ed instancabile sostegno, sia morale che economico, mi hanno permesso di arrivare fino alla fine del percorso, contribuendo alla mia formazione personale.

Per ultimi ma non meno importanti, ringrazio i miei amici. Sono stati fondamentali durante tutto il percorso di studi, sia durante le fatiche e lo sconforto, sia nei momenti di gioia e soddisfazione al raggiungimento del traguardo.

Un sentito grazie a tutti!

Milano, 2019 Giuseppe Godino

a mia nonna Giannina,
che dal cielo sono sicuro vegli su di me,
e che potrà finalmente vedere concluso questo percorso.

un bacione,
Giuseppe

Contents

Introduction **1**

1 State of art **7**

2 Characteristics of the Mechanical System and modelling **11**
- 2.1 Characteristics of the Mechanical System 11
- 2.2 Models Exhibition 16
  - 2.2.1 Model of Mechanical Systems 16
  - 2.2.2 Stepper Motor's Model 17
  - 2.2.3 Driver Model 25
- 2.3 Driving Logic 29

3 Trajectory Planning and Kinematics **33**
- 3.1 Implementation of Speed Profiles 33
  - 3.1.1 Real-Time Speed Profile 33
  - 3.1.2 Speed Profile with PWM Software 36
- 3.2 Mechanical System Kinematics 37

4 Numerical Analysis **43**

5 Experimental Tests **51**
- 5.1 Experimental Results 52
- 5.2 Discussion about the Present Tests 55

Conclusions **61**

Bibliography **63**
- Publications and Manuals 63
- Online Resources 64

List of Figures

1 Qualitative representation of the Adaptive Driving Logic working principle. 4

2.1 Photo of the Mechanical System. The blue box is connected to the linear guides through the bearings. It is connected to the motor by belt and pulley transmission. The stroke is limited by the limit switches. 11

2.2 Driver Scheme. The power supply and the stepper motor are connected to the DRV8825. The microcontroller sends the direction and the pulse signals to the step and dir. . . 14

2.3 Picture of a Bipolar Hybrid Stepper motor 14

2.4 Cross-Section of a Hybrid Bipolar Stepper motor. It is possibile to notice the stator windings and the toothed magnetic rotor. 17

2.5 Representation of a Generic Hybrid Bipolar Stepper Motor. On the left the stator with the eight poles is shown, while on the right the toothed rotor is represented. 19

2.6 Matlab plot showing the step response of the stepper motor second order transfer function. 22

2.7 Static Characteristic of Stepper Motor. 23

2.8 Qualitative Dynamic Characteristic Curves of Stepper Motor. The white area is called start/stop region, while the gray area is the working region. The latter is bounded by the pull-in and pull-out torque. 24

2.9 Block Diagram of the Driving System for Stepper Motor. The impulse signal enters in the Logic sequencer, that generates the eccitation signals for the Driver. The latter drive the bipolar stepper motor. 25

2.10 Bipolar Stepper motor with phase A and B. 26

2.11 Current signals in Full Step mode at left, Half step mode in the center e Microstep mode at right 27

2.12 An example of voltage signals in input to the Simulink model of stepper motor. 28
2.13 Block Diagram of the Adaptive Driving Logic. The adaptive code receives the error as input, and based on it, the code sets the acceleration and the maximum speed to generate the ramp. 31
2.14 Qualitative representation of the adaptive driving logic working principle. 32

3.1 Matlab Plot: Reference Speed vs Stepper Speed generated by Real-Time Algorithm: Speed profile of stepper motor in blue. Reference speed profile in red. 36
3.2 An Example of Speed Profile: in figure(a) the dotted line stress the linearity of the speed profile. 40
3.3 An Example of Speed Profile: In the two figures, the dotted line stress the linearity of the speed profile. 41

4.1 Open-loop Block Diagram. 43
4.2 Simulation of Trajectory planned with PWM Software. . . 45
4.3 Simulink Plot of Position Output X_{out} (m) of Trajectory planned with PWM Software. 46
4.4 Simulink Plot of Position Error (m) of Trajectory planned with PWM Software. 46
4.5 Simulink Plots of Trajectory planned with PWM Software. 47
4.6 Simulation of Trajectory planned with Iterative Algorithm. 48
4.7 Simulink Plot of Position Output X_{out} (m) of Trajectory planned with Iterative Code. 49
4.8 Simulink Plot of Position Error (m) of Trajectory planned with Iterative Code. 49
4.9 Simulink Plots of Trajectory planned with Iterative Code. 50

5.1 Experimental Setup. 52
5.2 Experimental Test with Load of 0.380 kg. ○ Step loss; □ No step loss; the dashed curve represents the qualitative load curve. 56
5.3 Test Adaptive Algorithm with a Load of 0.380 kg: The dashed curve represents the qualitative load curve. The green arrows show safety condition, while the red arrow show step loss. 57

5.4 Test Adaptive Algorithm with a Load of 0.215 kg: The dashed curve represents the qualitative load curve. The green arrows show safety condition, while the red arrow show step loss. 58
5.5 Test Adaptive Algorithm with a Load of 0.526 kg: The dashed curve represents the qualitative load curve. The green arrows show safety condition, while the red arrow show step loss. 59

List of Tables

2.1 Data-sheet for Linear Guides. 13
2.2 Belt and Pulley Transmission. 15
2.3 Data-Sheet Stepper Motor. 15
2.4 Model's parameters 17

3.1 Kinematics data 39

Sommario

Questo progetto di tesi tratta dello studio effettuato per l'implementazione di una logica di pilotaggio adattiva per un posizionatore lineare, movimentato tramite motore passo-passo, senza l'uso di sensori e senza conoscerne il carico. Si sceglie tale attuatore elettrico poiché, grazie alla sua proprietà intrinseca di eseguire spostamenti incrementali di una quantità fissa, è possibile sviluppare una logica di controllo ad anello aperto. Tuttavia, per carichi ingenti, si può presentare il problema della perdita del passo.
Soprattutto, se si pilota il motore attivando la cosiddetta modalità micropasso, la quale è indispensabile al fine di ottenere un movimento fluido del sistema meccanico. Dunque, l'obiettivo principale che si vuole soddisfare riguarda la realizzazione di una logica di pilotaggio in grado di intervenire, in caso di perdita di passo, sulla traiettoria del moto, di modo tale da riportare il sistema in condizioni di sicurezza.
La logica di controllo è stata sviluppata basandosi sullo studio dei diversi componenti del sistema meccanico, quali: il carrello, la trasmissione cinghia e puleggia, le guide lineari, il motore passo-passo, il driver e la scheda Raspberry Pi. Dopodiché si è realizzato un opportuno modello teorico atto alle simulazioni in ambiente Simulink. Scelta la traiettoria di riferimento, si è sviluppato uno studio sulla cinematica del sistema, con il fine di implementare una logica di pilotaggio adattiva. Quest'ultima è in grado di modulare il profilo di velocità scelto, in modo tale da minimizzare il tempo complessivo della traiettoria e nel contèmpo ridurre il rischio di perdita di passi. In fine, i risultati delle prove sperimentali fatte in laboratorio hanno validato la logica realizzata, permettendo il raggiungimento dell'obiettivo prefissato. Quest'applicazione va a dimostrare che, nonostante usualmente si adottino sistemi di controllo in retroazione per risolvere la perdita del passo, sia possibile comunque adottare una strategia di controllo ad anello aperto. Quest'ultima è ricercata al fine di ottenere vantaggi come: la semplificazione del controllo e il risparmio economico sull'uso di sensori.

Parole chiave: motori passo-passo, controllo in anello aperto, controllo in retroazione, logica di pilotaggio adattiva, modalità micropasso

Abstract

In this thesis project, an experimental investigation of an adaptive sensorless open-loop strategy for mechanical systems through stepper motors is developed. These electric actuators are usually selected because it is possible to realise an open-loop driving logic, thanks to their intrinsic property of performing incremental movements of a fixed quantity called step. Unfortunately, during the transport of heavy loads, a problem of step loss might occur. Especially if the motor is driven by activating the micro-step mode, required in order to obtain a fluid movement of the system. Therefore, the main objective of this thesis is the realisation of a driving logic able to modify the trajectory of motion, in the event of step loss, with the aim of bringing the system back to safe conditions. The control logic has been developed by studying the main components of the mechanical system, which are the cart, the belt-and-pulley transmission, the linear guides, the stepper motor, the driver, and the Raspberry Pi board. An appropriate theoretical model is realised, used for numerical analyses. Then, by choosing the reference trajectory for the motion, a study of the kinematics of the system was conducted, in order to implement an adaptive driving logic. The latter is able to modulate the speed profile chosen, so as to minimize the time trajectory and reduce the risk of step loss. Finally, the results of the experimental tests performed in the laboratory validated the coded logic and allowed the achievement of the objectives established at the beginning of the project. It demonstrates that, although feedback control systems are usually employed to solve step loss, it is possible to adopt an open-loop control strategy. This is sought in order to obtain advantages such as simplification of control and cost savings on the use of sensors.

Keywords: stepper motors, open-loop control strategy, adaptive driving logic, micro-step mode, feedback control systems

Introduction

This thesis presents an experimental study of a sensorless open-loop control strategy for a mechanical system drived by stepper motors.

The mechanical system taken into exam represents a linear positioner, which is suitable for transporting mechanical components. It is typically used for assembly, and production plants.
The project in exam is characterised by the following elements:

- Cart;
- Linear guides with limit swithces;
- Belt and pulley transmission;
- Stepper motor;
- Raspberry Pi board;
- Driver DRV8825;
- Driving Logic;

The cart is a plastic container, able to transport various components, which moves from one limit switch to the other along the linear guides. With regard to the chosen transmission, it is appropriate to make a comparison between two solutions typically adopted for the realisation of linear positioners, such as ball screw and belt-and-pulley.
Ball screw has several advantages, such as greater rigidity and stopping precision. It is, however, more expensive. On the other hand, belt-and-pulley allows faster movements, shorter time cycles, higher productivity, and it is less expensive than the former. However, belt-and-pulley has poor stiffness and stopping precision, but for these applications a higher stopping precision is not necessary, since the cart must go from one limit switch to the other.

Hybrid stepper motors are regularly used for simple point-to-point positioning tasks. In fact, these actuators present some important qualities such as high reliability, ease of use, high torque at medium-low speeds, high stopping precision and cost-effectiveness.
Thanks to their structure, stepper motors are able to move mechanical components through increasing rotor positions with fixed quantities called steps. Therefore, using a driver (a board able to drive bipolar stepper motors) and after outlining a proper control logic, it is possible to drive the mechanical system through an open-loop control strategy. However, if heavy loads must be moved, making fast starts might result in a loss of steps.
Therefore, to reduce the negative impact of this phenomenon, implementing linear speed profiles, such as ramps, is customarily used. This avoids abrupt starts. The control logic must be able to select the pulse frequency and the duty cycle according to an appropriate trajectory, to realise the pulse signal to be given as a reference to the driver. Then, the control logic will be loaded into the Raspberry PI board, where it will be executed.
The DRV8825 driver was chosen for its ability to activate the micro-step mode, through which to obtain a fluid and precise movement of the cart. Micro-steps are likely to increase the resolution, which, in turn, results in a more fluid motion. If higher loads are to be transported, the motor could lose steps. Should the sum of the load torque plus friction and stopping torque of the motor be greater than the incremental torque of a micro-step, several micro-steps must be performed until reaching a sufficient accumulated torque for the motion. Micro-steps are lost until this passage occurs. Usually, the load curves give information about step loss. These curves represent the limit load torque at varying frequencies, after which a loss of synchronism occurs. Since the transported load is unknown, these curves are not known previously. Therefore, the acceleration curve is considered with respect to the frequency. This should represent a good approximation to its behaviour. Consequently, the two key parameters to modify, in order to prevent step loss, are acceleration and maximum velocity.
Therefore, an adaptive driving logic is required to modify the speed profile (the slope and the peak of the ramp) without the use of sensors or knowledge of the load. This would avoid step loss. To implement the code, it is important to detect when step loss occurs. Typically, an encoder is exploited. It directly gives information on the angular position of the rotor. It thus allows to evaluate the error between the reference and the measure. However, in the following thesis project, since the use of sensor will be avoided, the step loss will be detected by estimating the actual number of steps and comparing it with the theoretical number of steps need to reach

the limit switch.

Purposes of the Thesis

The aim of this thesis is to create an open-loop adaptive driving logic for the mechanical system without the use of sensors and without knowing the load. This modifies the reference trajectory and has the aim of preventing step losses and minimize time cycle.

Taking into account the difficulties of this task, it is necessary to provide a description of the main components of the mechanical system under examination, in order to seek for theoretical models that describe their behaviour. Then, it is necessary to implement the linear speed profile through which to move the cart.
The realised trajectory will allow to delineate an appropriate driving algorithm, which will supply the voltage pulses necessary to the driver. Moreover, thanks to the study on the kinematics of the system, the code will be able to calculate the number of steps needed to complete the trajectory, and the error made with regard to the theoretical number of steps. In fact, during the motion, due to heavy loads, this error could become greater than zero, thus indicating the loss of steps.
A series of experimental tests will be carried out in the laboratory to represent the load curve – or rather – the approximate load curve. The experimental tests will be performed by inserting a certain load on the cart, and by varying the acceleration and the maximum speed. Once the load curve is obtained, it is possible to outline the adaptive driving logic. Once the limit switch is reached, the adaptive driving logic calculates the number of steps made and the time spent by the cart to move along the path until the limit switch is reached.
Then, the error made between the number of theoretical and measured steps is detected, and if it is bigger than the limit error, the code modifies the trajectory accordingly, so to return it to safe conditions in the next cycle. In particular, if the error made is higher than the limit value, the acceleration is reduced and, if this occurs again, the speed is decreased. Conversely, if the error made is less than the limit value, the acceleration is increased and, if this occurs again, the speed is increased.
A qualitative example of the results that the code should obtain is represented in Fig. 1.

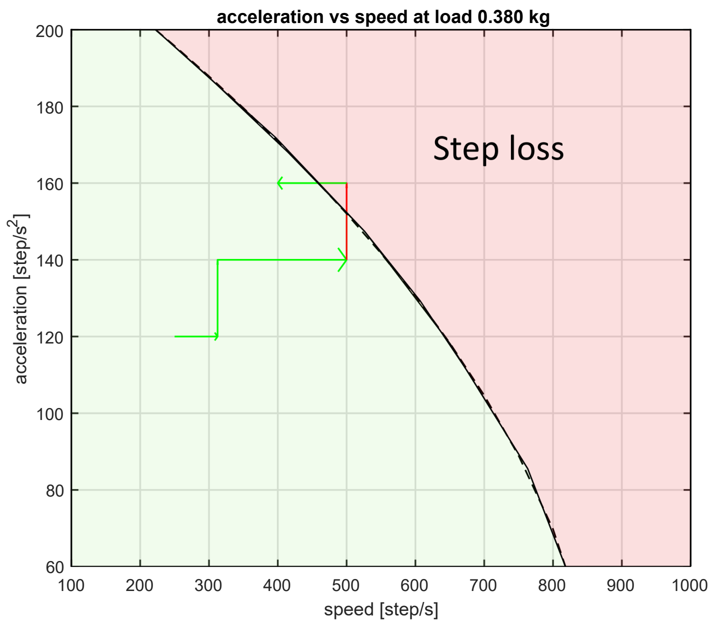


Figure 1: Qualitative representation of the Adaptive Driving Logic working principle.

The adaptive driving logic continues to increase the performance of the speed profile, so to minimise the time cycle, until the motor does not lose steps. when this happens, that is, when the arrow in the graph surpasses the qualitative load curve, the code modifies the speed profile to bring the system back to safe conditions. After a certain number of cycles, the code has successfully changed the acceleration and the maximum velocity of the speed profile, thus staying in the area of an optimum point. Under this condition, the speed profile has the minimum time cycle without losing steps. At this point, costs are minimised and productivity maximised.

Once the objectives of the following project thesis were set out, its structure is introduced.

Outline

The text of the thesis is structured as follows:

In the first chapter, the state of the art is outlined with regard to the driving of stepper motors without the use of sensors. A comparison between the closed-loop and open-loop control strategies is introduced, highlighting the advantages and disadvantages of each of them. The choice of adopting the open-loop control strategy is explained and motivated. The control strategies adopted by other researchers are introduced as well, together with some solutions to the various issues that may occur during the development of the aforementioned project. Guidelines for the implementation of the adaptive driving logic are also outlined.

The second chapter introduces the main components of the mechanical system. It provides a description of the theoretical models of its main components, through which it is possible to realise a suitable Simulink model. In the last section, the adaptive driving logic is discussed through the use of pseudocodes.

The third chapter expounds the trajectory planning through which to drive the mechanical system in open loop. Two speed profiles typically used in the driving of stepper motors are treated, a ramp profile realised with a pulse width modulation (PWM) Software and another with an iterative algorithm. The trajectory chosen is discussed, and the kinematics of the mechanical system is described. Finally, it illustrates the fundamental parameters that the control logic revise.

In the fourth chapter, the simulation results of the Simulink model describing the system under control are shown. These are carried out by supplying the two speed profiles studied. Moreover, observations on the behaviour of the mechanical system and the potential of the model are here presented.

The fifth chapter shows the results of the experimental tests carried out in the laboratory, which are reworked and shown through graphs. The results of these tests are interpreted to implement the control logic. In the section dedicated to the discussion of the tests, the validity of them will be verified. Finally, conclusions are drawn on the soundness of the control logic implemented.

Chapter 1

State of art

In this first chapter, the state of the art concerning the driving of stepper motors without the use of sensors is reviewed. The main issues and their resolutions by researchers will be discussed.

In general, mechanical systems for industrial production are controlled according to two possible strategies, open-loop and closed-loop. Typically, the closed-loop control strategy is preferred thanks to the feedback guaranteeing a better performance with regard to positioning accuracy and to fluid motion, as opposed to the open-loop strategy.
Oftentimes, in industrial applications requiring position and speed control, two conventional proportional (P) and proportional-integral (PI) closed-loop controllers are used because of their simple design and low cost. Unlike the simple proportional controller, the PI controller is also able to ensure an optimum reference tracking; its integral action in fact reduces the steady state error. However, it may present a slow response, large overshoots and oscillations. In [14], a self-tuning PI controller using field-oriented control (FOC) for hybrid stepper motor is presented. Unlike the conventional PI, the self-tuning PI controller is able to adapt itself to the motor operating conditions by varying the gains. The result is an improvement on the performance of the mechanical system. However, manufacturers are looking for new solutions to reduce production costs, for example, by adopting an open-loop control strategy, which saves on sensor costs and sophisticated controllers.
Hybrid stepper motors are widely open-loop controlled thanks to their property to move mechanical components through increasing rotor positions with fixed quantities. Moreover, this actuator presents some significant qualities such as high reliability, ease of use, high torque at medium-low speeds, high stopping precision and cost-effectiveness. For simple point-

to-point positioning tasks, stepper motors thus represent a great and low-cost solution. However, these presents several issues. The actuator is supplied with tension pulses that generate some torque ripple, overshoot, and resonance, which are responsible for mechanical vibrations and loss of performance. Moreover, in industrial applications, where heavy loads need to be moved quickly, the motor can lose steps. Therefore, if the mechanical system presents variability on the parameters and external disturbances caused by the load torque, it is preferred to adopt closed-loop control systems, which ensure a greater position accuracy, and a reduction of the sensitivity to load torque disturbances and inertia variations.
In fact, typical applications of closed-loop control systems are related to high-precision machining such as machine tools and robotic manipulators. However, the closed-loop control strategy also presents some disadvantages. In fact, the feedback occurs by means of mechanical encoders, which increase the size and cost of the system. Moreover, the mechanical measures, such as position and speed, suffer in high-temperature and high-vibration environments. Knowing these limitations, solutions have been sought for the control of stepper motors without the use of sensors.
The following research [3] presents the results of experimental tests about the closed-loop position control of stepper motors without the use of the encoder. The mechanical variables have been estimated by the use of steady state-extended Kalman filter. On the contrary, the uncertainty of the load torque is reduced by the use of current feedback. However, it is necessary to estimate the initial rotor position. In the aforementioned reference, an *impulse voltage* technique is used.
As stated by [4], in order to estimate the initial rotor position of a synchronous permanent magnet motor, two strategies are considered. These are a vector of impulse tension and a vector of rotating tension. However, the solutions mentioned above still require the use of electrical sensors, sophisticated controllers, and a high computational cost.
In this thesis project, the objective is to demonstrate that it is possible to drive linear positioners in open-loop, thus achieving a good dynamic behaviour, productivity, and low costs. The goal is obtained by driving the stepper motor through a proper driver, i.e. a board able to drive bipolar stepper motors.
Driving the motor through a driver with an open-loop strategy has already been studied in literature, for example in [1, 11]. The following [19] presents guidelines on how to implement a simple driving system for stepper motors through driver and Raspberry Pi board. The reference also includes a basic guide on how to realise a code, which represents the driving logic, written in Python with PigPio library [17]. The driver is an important tool as it

directly excites the phase through a correct pulse sequence. It also allows to activate the micro-stepping mode, through which the dynamic motion affected by the torque ripple and resonances, is improved by increasing the resolution – that is, the step is divided in a certain number of micro-steps. The result is a more precise and fluid motion.
In [2], an optimum pulse width modulation (PWM) system is realised, through which it is possible to drive stepper motors controlling the phase currents. In the article quoted above, a model of the phase currents suppling the stepper motor during the micro-stepping mode is developed. However, studies and experimental results obtained by electric actuator manufacturers confirm that the micro-stepping mode is not fail proof, because, with higher loads to transport, the motor could lose steps.
In fact, by activating the micro-step, a load torque is generated, resulting in a magnetic backlash that delays the rotor in reaching the desired position, as reported in [16]. Consequently, if the load torque plus the friction and stopping torque of the motor is greater than the incremental torque of a micro-step, a series of micro-steps must be made until the accumulated torque is enough for the motion. Until this scenario is reached, all micro-steps are lost. Hence, to reduce the negative impact of this phenomenon, implementing linear speed profiles, such as ramps, is customarily used. The correct choice of kinematic quantities for trajectory planning is essential to obtain a fluid and precise movement for the positioner.
The adaptive driving logic that will be developed in this thesis project will modulate the speed profile, so to minimise the time cycle and the risk of steps loss. Finally, to develop the code, it is important for the mechanical system to detect step loss. Different studies about this topic have been carried out, for example [6, 7]. Where the load angle is estimated through the extended Kalman filter or by means of measurements of an electromotive force. In fact, if the load angle is higher than the limit one, a stall or lose of synchronism can be expected.

In conclusion, the approaches taken by other researchers have been examined alongside the main topics discussed in this thesis. This allowed to raise awareness of the limits of this project, such as the lack of estimation of the mechanical parameters, the uncertainty of the load that cannot be reduced, and the impossibility of checking step loss through the sensors. In the following thesis, new possible solutions are presented in order to reach the target.

Chapter 2

Characteristics of the Mechanical System and modelling

In this chapter, the characteristics of the mechanical system, the design and implementation of its dynamic model are presented. In section 2.1, the behaviour of the mechanical system is briefly outlined. In section 2.2, the theoretical models of the components of the main system are described. In the last section 2.3, the logic implemented of driving the stepper motor through the drafting of pseudocodes is presented.

2.1 Characteristics of the Mechanical System

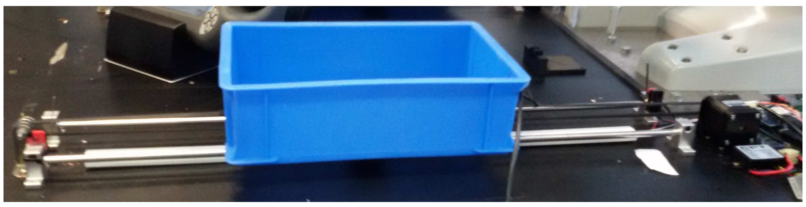


Figure 2.1: Photo of the Mechanical System. The blue box is connected to the linear guides through the bearings. It is connected to the motor by belt and pulley transmission. The stroke is limited by the limit switches.

The objective of this section is to present the main components of this project, which are listed and discussed individually.

Cart

The cart is a plastic box measuring: 280x175x80 [mm]. It allows small precision-assembly components to be transported, as well as other component with a more relevant mass.

Linear Guides

The cart moves along linear guides, typically used for 3D printers. They are made of aluminium bearings and steel axles. For each cart, there are two pairs of bearings and axles. For more details on technical characteristics, see Tab. 2.1. Their use creates an opposite force generated by friction, which becomes the starting torque. The mechanical torque must therefore overcome the starting torque to be set in motion. A further step involves the calculation of the coefficient of the static and rolling friction. However, due to the difficulty in estimating these parameters, and as such computations are beyond the scope of this dissertation, an approximate value taken from on [18] was preferred.

Mechanical Transmission

The transmission is characterised by the belt and the pulleys. The GT2 belts are made of rubber. They are synchronous and created specifically for a round-tooth profile, which ensures that the tooth of the belt fits properly into the groove of the aluminium roller. Thus, if the pulley is in the opposite direction, there is no chance of movement in the thong area. There are at least six teeth in touch with the roller. This minimises the possibility of slippage and helps to further reduce play. For more details on these technical characteristics, see Tab. 2.2.

Stepper Motor

The drive chosen for the movement is a NEMA-17-stepper motor. This choice was made on the basis of the characteristics of its main performance. These are excellent precision and repeatability of the step angle; fair maintenance torque; and excellent promptness. In the industrial environment, these features are highly sought-after, especially for position and speed-control applications. It also represents a low-cost solution and it is relatively simple to drive using the appropriate driver. In section 2.2.2, its features and functionalities will be expounded further.

Table 2.1: Data-sheet for Linear Guides.

Quantity	**Unit**	**Module**
Axle		
Length	mm	672
Diameter	mm	8
Material		Stainless steel
Linear bearing		
Diameter	mm	16
Hole diameter	mm	8
Material		Alluminum

A technical file is available, see Tab. 2.3.

DRV8825 Driver

The driving of stepper motors is performed through the driver, a hardware device which receives the logic signals, processed by a software via logic sequencer, and transforms them into current pulses. They excite the phases sequentially to realise the steps at the desired frequency. The DRV8825 board was chosen. It is a low-cost solution for controlling a single bipolar motor reaching a current up to 2.5 [A] and a supply voltage of 45 [V]. Further technical references are given in [10]. The device has two H-bridge converters and a micro-stepping indexer, which allows the micro-step to be exploited up to one thirty-second resolution.

Raspberry Pi Board

Raspberry Pi is a single-board computer developed in the United Kingdom by The Raspberry Pi Foundation. It represents a power solution for mechatronics applications. Through an appropriate port, it is in fact possible to insert an SD card, for the operating system. Linux was chosen for the present thesis. Some guidelines for stepper motor driving can be found in [19]. Physical connection on the stepper motor with the microcontroller is shown in Fig. 2.2.

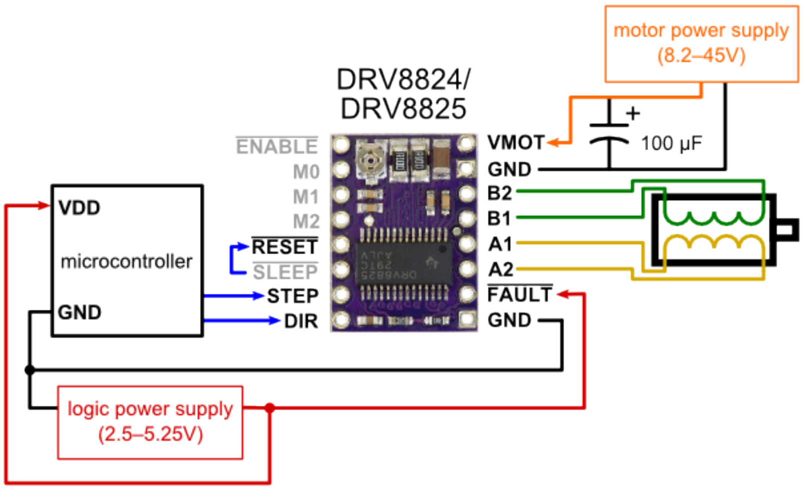


Figure 2.2: Driver Scheme. The power supply and the stepper motor are connected to the DRV8825. The microcontroller sends the direction and the pulse signals to the step and dir.

Figure 2.3: Picture of a Bipolar Hybrid Stepper motor

Table 2.2: Belt and Pulley Transmission.

Quantity	**Unit**	**Module**
GT2 belt		
Length	mm	6
Step	mm	2
Material		Rubber
GT2 pulley		
Diameter	mm	16
Hole diameter	mm	5
Teeth		20
Material		Alluminum

Table 2.3: Data-Sheet Stepper Motor.

Quantity	**Unit**	**Module**
Step Angle	*deg*	1.8
Holding Torque	Nm	0.59
Rated Current/phase	A/phase	1.7
Step Accuracy	%	5
Phase Resistance	ohms	1.8
Inductance	mH	3.8
Rotor Inertia	gcm^2	82

2.2 Models Exhibition

2.2.1 Model of Mechanical Systems

The purpose of this subsection is to present the study of the behaviour of the mechanical system by considering a simple model with one degree of freedom. For this system, it is possible to derive the appropriate differential equation. The approach typically used to analytically describe the motion of a generic mechanical system is to exploit the Lagrange equation. If we consider a generic vector of variables $\underline{q}$, the relation is represented by the following:

$$\left\{\frac{d}{dt}\left(\frac{\partial E}{\partial \dot{\underline{q}}}\right)\right\}^T - \left\{\frac{\partial E}{\partial \underline{q}}\right\}^T + \left\{\frac{\partial D}{\partial \dot{\underline{q}}}\right\}^T + \left\{\frac{\partial V}{\partial \underline{q}}\right\}^T = \underline{Q} \tag{2.1}$$

Since the system has one degree of freedom, the state vector consists of a single variable ϑ, which corresponds to the mechanical angle. Therefore, its derivatives are: angular velocity ω and angular acceleration $\dot{\omega}$. For the mechanical forces, we have: the mechanical and the resistant torques; the latter is caused by the friction force contrasting the motion of the cart. For clarity, only the friction force caused by the static component was considered, neglecting the viscous one. Thus, the presence of conservative force fields is not evaluated. The energy contributions involve only the kinetic energy of the pulleys and the moving mass, while the potential energy and the dissipative one are practically negligible, since there are neither springs nor dampers. Given these considerations, the system's equation of motion is represented by eq. (2.2). Instead, in the tab. 2.4 the main system's parameters are listed.

$$\left(2J_d + MR_d^2\right)\dot{\omega} = C_m - \mu M g R_d \tag{2.2}$$

Where the following parameters are:

J_d = Pulley inertia [kgm^2];

M = Mass to move [kg];

R_d = Primitive pulley radius [m];

μ = Friction coefficient;

C_m = Motor Torque [Nm];

Table 2.4: Model's parameters

Quantity	Symbol	Unit	Module
Pulley's radius	R_d	m	0.0064
Cart's mass	M	kg	0.5
Pulley's inertia	J_d	kgm^2	1.6211×10^{-7}
Mechanical torque	C_m	Nm	
Load Torque	C_r	Nm	

Notice that, it is possible to consider an equivalent inertia equal to:

$$J_{eq} = J_m + \left(2J_d + MR_d^2\right) \tag{2.3}$$

Where J_m is the motor inertia. Having obtained the equation of motion, it is appropriate to switch from the time domain to the frequency domain through the Laplace transform, in order to perform numerical simulations. The following equation is thus obtained:

$$J_{eq}s^2\Theta(s) = C_m(s) - \mu MgR_d \tag{2.4}$$

2.2.2 Stepper Motor's Model

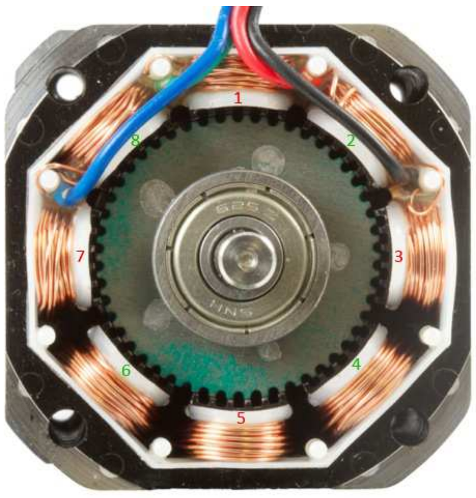


Figure 2.4: Cross-Section of a Hybrid Bipolar Stepper motor. It is possibile to notice the stator windings and the toothed magnetic rotor.

This subsection briefly introduces the features and the working principle of stepper motor, with the purpose of obtaining the differential equations that describe its behavior.

As examined in detail in [1, 11], the stepper motor is an electric actuator powered by direct current, mainly used for the following applications: point-to-point positioning (where high acceleration, speed and positioning accuracy are required); speed control; and controlled path. There are three main categories of them: the permanent magnet, the variable reluctance and the hybrid.
The permanent magnet stepper motor presents rotor and stator poles that are not toothed. Instead the rotor have alternative north and south poles parallel to the axis of the rotor shaft. The hybrid stepper motor is a combination of both permanent magnet and the variable reluctance. It has a permanent-magnet toothed rotor, which guides well the magnetic flux through the air gap. The magnetic rotor presents two sets of fifty teeth each, which are offset by a certain angle. One ring is all south poles, and the other ring is all north poles. In the stator, instead, there is the excitation circuit consisting of windings around the poles, which are used to encourage or discourage the flow of magnet flux through certain poles according to the rotor position required. The hybrid stepper motor has two phases, A and B, which are situated on four of the eight stator poles. The windings of each phase can be excited by positive and negative current. By exciting the phases in sequence through electrical impulses, an electromagnetic force is produced, causing the realignment of the rotor teeth, which rotate by a fixed quantity. Then, the continuous rotation of the motor is produced by sequential excitation of the phase windings.
A generic representation of the stator and rotor of a hybrid stepper motor is provided in Fig. 2.5.
Instead, in the variable reluctance motor the teeth tend to realign, so that the reluctance of the stack's magnetic circuits is reduced. When the rotor teeth are aligned with the stator, the reluctance is at minimum, while the magnetic flux in the stack is at maximum. The number of rotor teeth is different from the stator's, so that they are not already aligned at the start.

After introducing the actuator and its physical working principle, the dynamic model's differential equations are defined.

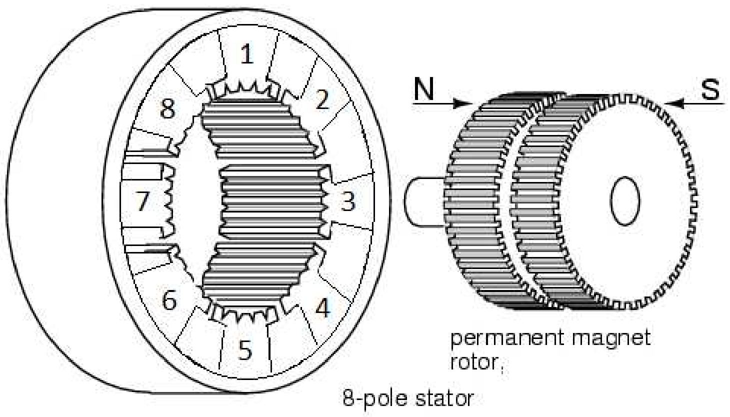


Figure 2.5: Representation of a Generic Hybrid Bipolar Stepper Motor. On the left the stator with the eight poles is shown, while on the right the toothed rotor is represented.

First, the equivalent excitation circuit represented by the following eq. (2.5) is considered.

$$V_{eq} = RI_{eq} + L\frac{dI_{eq}}{dt} - e_m \tag{2.5}$$

$$e_m = K_m\omega \sin\left(N\vartheta + \varphi\right) \tag{2.6}$$

Where the following parameters are listed:

R = Excitation circuit resistance $[ohm]$;

L = Inductance of the excitation circuit $[H]$;

e_m = Electromotive force [V];

K_m = Motor constant $\left[\frac{Nm}{Arad}\right]$;

ω = Angular speed of the motor; $\left[\frac{rad}{s}\right]$;

ϑ = Motor angular position $[rad]$;

φ = Phase;

N = Number of rotor pole pairs;

As the excitation circuit is composed by two phases A and B, it is represented by the following differential equations:

$$V_a = RI_a + L\frac{dI_a}{dt} - K_m\omega \sin\left(N\vartheta + \varphi\right) \tag{2.7}$$

$$V_b = RI_b + L\frac{dI_b}{dt} + K_m\omega \sin\left(N\vartheta + \varphi\right) \tag{2.8}$$

Once the electrical component's equations have been determined, it is possible to define that of the mechanical torque. Assuming that the magnetic circuit is linear, and that the mutual inductance between the two phases is negligible, it can be considered as the sum of two components.

$$C_m = C_a + C_b \tag{2.9}$$

Where, the mechanical torque generated by the two phases are:

$$C_a = -\frac{e_a I_a}{\omega} = -K_m I_a \sin(N\vartheta) \tag{2.10}$$

$$C_b = -\frac{e_b I_b}{\omega} = K_m I_b \cos(N\vartheta) \tag{2.11}$$

Consequently, the resulting final expression is the following:

$$C_m = -K_m I_a \sin(N\vartheta) + K_m I_b \cos(N\vartheta) \tag{2.12}$$

Knowing the mechanical torque, the stepper motor's differential equation is thus found. It is obtained from a simple dynamic equilibrium between the motor side and the load side, assuming that the latter is connected by a rigid joint.

$$J_m\left(\frac{d\omega}{dt}\right) + D\left(\frac{d\vartheta}{dt}\right) = C_m - C_r \tag{2.13}$$

Where the term $D\left(\frac{d\vartheta}{dt}\right)$ represents an energy dissipation caused by the intrinsic viscous friction component within the motor side. Furthermore, the estimate of the parameter D is complex, and has not been considered in this thesis project. For this reason, a value of good approximation is chosen. The mathematical model that describes the behavior of a bipolar hybrid stepper motor is represented by the following system of nonlinear differential equations:

$$\begin{cases} \dfrac{dI_a}{dt} = \dfrac{[V_a - RI_a + K_m\omega \sin(N\vartheta)]}{L} \\ \dfrac{dI_b}{dt} = \dfrac{[V_b - RI_b + K_m\omega \cos(N\vartheta)]}{L} \\ \dfrac{d\omega}{dt} = \dfrac{[-K_m I_a \sin(N\vartheta) + K_m I_b \cos(N\vartheta) - D\omega - C_r]}{J} \\ \dfrac{d\vartheta}{dt} = \omega \end{cases} \tag{2.14}$$

Where the so-called dent torque T_d is disregarded, as it is generally a negligible contribution. Knowing the set of equations, it is then possible to perform some dynamic analyses to understand the behaviour of the motor, by carrying out, for example, a step response. For this reason, it is necessary to move from the time domain to the so-called Laplace domain, as to obtain a transfer function.

Stepper-Motor Transfer Function

The stepper-motor pulse response can be well approximated by a second-order transfer function for a single phase actuator; as reported by [11, p. 100].

The relation obtained is the following:

$$G(s) = \frac{\Theta_0}{\Theta_i} = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{2.15}$$

$$\xi = \frac{D}{2J_m\omega_n} \tag{2.16}$$

Generally, the step response is oscillating, due to the fact that the damping ratio, in ordinary stepper motors, is often less than 0.5, with a settling time on the 5% of time. The step response is represented in Fig. 2.6.

The information given are useful to establish reliable values for parameters that cannot be estimated without suitable experimental tests carried out by means of sensors, as conducted in [3].

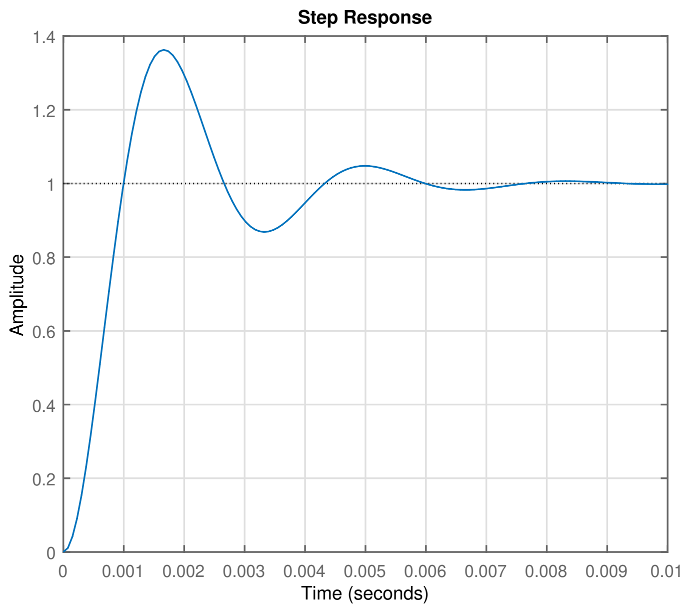


Figure 2.6: Matlab plot showing the step response of the stepper motor second order transfer function.

They are listed as follow:

- $K_m = 0.38\left[\frac{Nm}{Arad}\right]$;
- $\xi = 0.31$;
- $D = 2J_m\omega_n\xi = 0.01; \left[\frac{Nms}{rad}\right]$;

Once the parameters have been estimated, the actuator's performance values are reported following the excitation.

- Rise time $= 0.0007\,[\mathrm{s}]$;
- Settling time $= 0.057\,[\mathrm{s}]$;
- Overshoot $= 36.3\,[\%]$;

After determining the model of the motor, we are going to deal with further characteristics of the stepping motor, such as static and dynamic characteristic, and the resonance phenomenon.

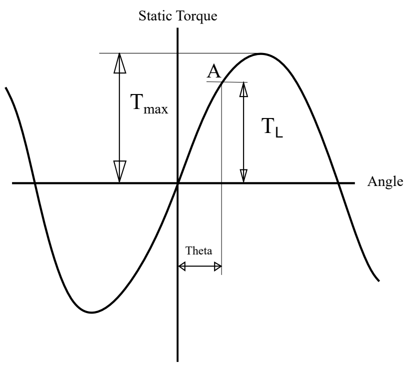


Figure 2.7: Static Characteristic of Stepper Motor.

Static Characteristic

The static characteristic of the stepping motor represents the relationship between the mechanical torque applied to the motor shaft and the angular displacement from the equilibrium position, when the actuator is powered in static conditions as shown in Fig. 2.7. Applying the maximum torque, called Holding Torque, the maximum angular displacement is reached, beyond which another equilibrium position is reached. The Holding Torque, therefore, represents the maximum torque that can be applied to the motor shaft without causing a rotation.

Dynamic Characteristic

The dynamic characteristic of the stepper motors presents two characteristic curves: pull-in and pull-out. The pull-in characteristic curve represents the range of the load torque caused by friction, in which the motor can start and stop without losing its synchronism (therefore steps) at various frequencies. The pull-in curve decreases as the inertia of the load increases. The pull-out characteristic curve, on the other hand, represents the limit load torque at varying frequencies, after which a loss of synchronism occurs. The frequency range between the two curves represents the operating range of the stepper motor. This is shown in Fig. 2.8.

In this dissertation, these curves are not known previously as the transported load is unknown. Therefore, the acceleration curve is considered with respect to the frequency. This represents nonetheless a good approximation to its behaviour. Consequently, the two key parameters are acceleration and max velocity. Both of these values are considered at the

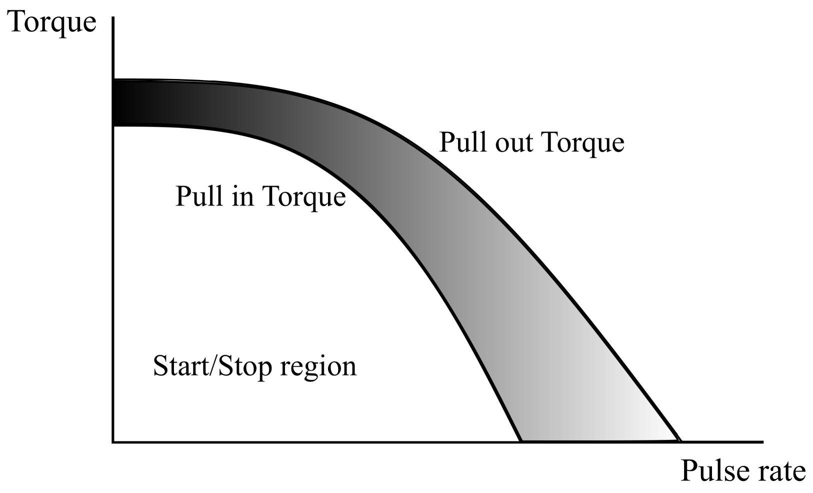


Figure 2.8: Qualitative Dynamic Characteristic Curves of Stepper Motor. The white area is called start/stop region, while the gray area is the working region. The latter is bounded by the pull-in and pull-out torque.

variation of the load. The tracing of this curve is carried out through experimental tests, and is treated in Chapter 5.

Resonance Effect

The functionality of the stepper motor is affected negatively when resonance occurs. This is a highly oscillatory phenomenon that causes vibrations and disturbances to the overall motion. If the amplitude of the oscillations is high, such instability might also cause a loss of synchronism and, consequently, of the step. Resonance phenomena also influence the pull-in and pull-out characteristic curves. In particular, in the pull-out curve, there are three types of resonances: low ($100 - 200\ Hz$), medium ($500 - 1500\ Hz$) and high ($2500 - 4000\ Hz$) frequencies. On the other hand, in the pull-in curve certain instabilities occur for specific stepping rates. They are indicated by dips of torque drop, typically visualised in the graph as small regions also known as small islands. For high-load applications, the use of speed ramps is crucial to reduce the risk of step loss. Unfortunately, this solution makes running into resonance almost unavoidable for the motor. The solution is then to ensure that the motor goes through the velocities at risk at a higher speed and as fast as possible, to reduce the undesired resonance. Finally, the micro-step mode is an

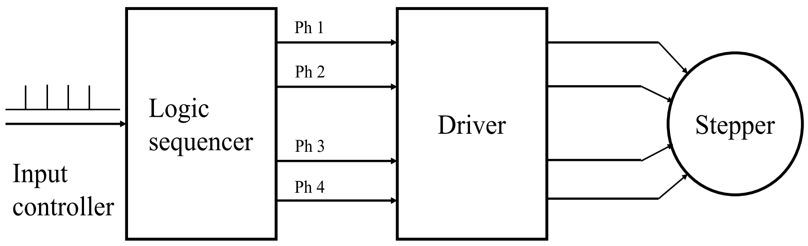


Figure 2.9: Block Diagram of the Driving System for Stepper Motor. The impulse signal enters in the Logic sequencer, that generates the eccitation signals for the Driver. The latter drive the bipolar stepper motor.

effective answer even for this harmful phenomenon.

2.2.3 Driver Model

In this subsection the working principle of the Driver is presented. This is a fundamental device for the correct driving of the stepper motor. Going back to [11], we find a driving system represented by a block diagram, where there is a train of pulses in input. These represent the number of $steps/s$ that the actuator must perform, which enter the Logic sequencer (the logic circuit). It will execute the sequential excitation signals of the motor windings. Subsequently, the hardware component receives them and translates them via the H-bridge into electrical excitation signals, which feed effectively the actuator windings. Depending on the appropriate switching sequence, the rotor will perform a series of fixed steps. A sketch of the block diagram is represented in Fig. 2.9. With reference to this article [2], it is possible to exploit the operation of this device according to different modes. The so-called full-step mode, which allows the motor to perform a full-step angle per second. The Half Step, instead, lets the rotor rotate at an angle equal to half of the step. Other modes are the one-fourth mode and the micro-step, the latter allowing the motor to execute smaller angles like: 1/8, a 1/16 and 1/32 of step.

As the stepper motor is a four-wire bipolar motor, the stator has the following phases: A $+$, A-, B $+$, B-. The representative diagram of the motor phases is represented in Fig. 2.10.
The different driving modes allow to excite these phases with proper out-of-phase voltage pulses. If we consider the full-step drive mode, for example, phases A and B are fed by two square-wave voltage signals which are

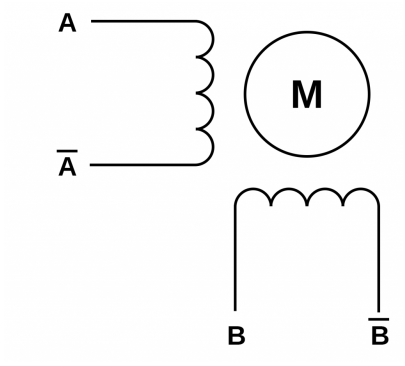


Figure 2.10: Bipolar Stepper motor with phase A and B.

$\frac{\pi}{2}$ phase-shifted between them. During the movement of the mechanical system, it is possible to obtain an unsatisfactory result from the dynamic viewpoint due to a damaging phenomenon called torque ripple, which causes mechanical vibrations and a loss of engine performance. The technique that moderates this harmful phenomenon is the micro-step mode, which allows to attenuate the mechanical vibrations of the moving mass reducing the step angle. The resulting motion is more fluid.
The basis of this mode is the possibility of controlling the current supply of the phases, so to produce a constant torque and rotor angles in stable positions. From the electrical standpoint, as illustrated in [8], both motor windings are powered by quantised sinusoidal and co-sinusoidal current signals, which can be described by the following relationships:

$$I_a = I_{pk} cos\left(\frac{h\pi}{2N}\right), \quad h = 0, 1, \ldots, 4N - 1. \tag{2.17}$$

$$I_b = I_{pk} sin\left(\frac{h\pi}{2N}\right), \quad h = 0, 1, \ldots, 4N - 1. \tag{2.18}$$

Where I_{pk} represents the peak current e N is the number of micro-steps. A qualitative example of the signals obtained is shown in Fig. 2.11, where the micro-step is set to an eighth of a step. Knowing the analogue signals in output from the driver, it is then possible to simulate them in a Simulink environment. The article [2] highlights how the latter are obtained through PWM by setting an appropriate fixed frequency, or $step/s$ executed, and a certain duty cycle, which divides the duration of on-off time. This is typically set at 50 %.
However, in the present project, these analogue input signals are simulated exploiting the so-called Embedded Matlab Function in Simulink. Thus, the realized algorithm allows to reconstruct square wave signals for the

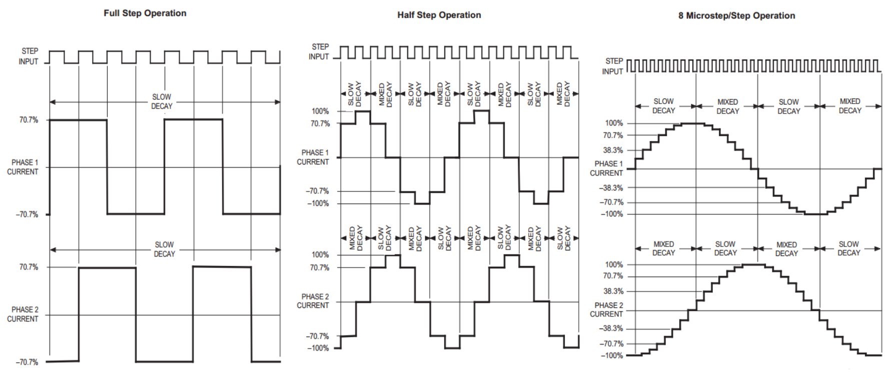
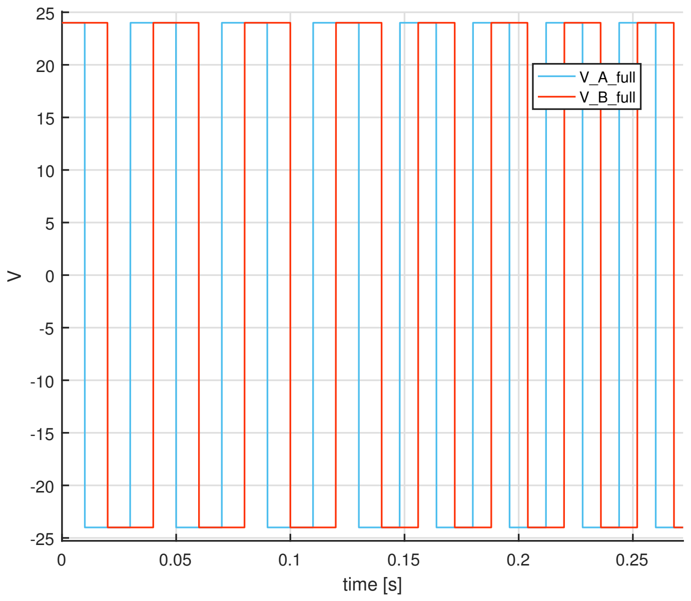
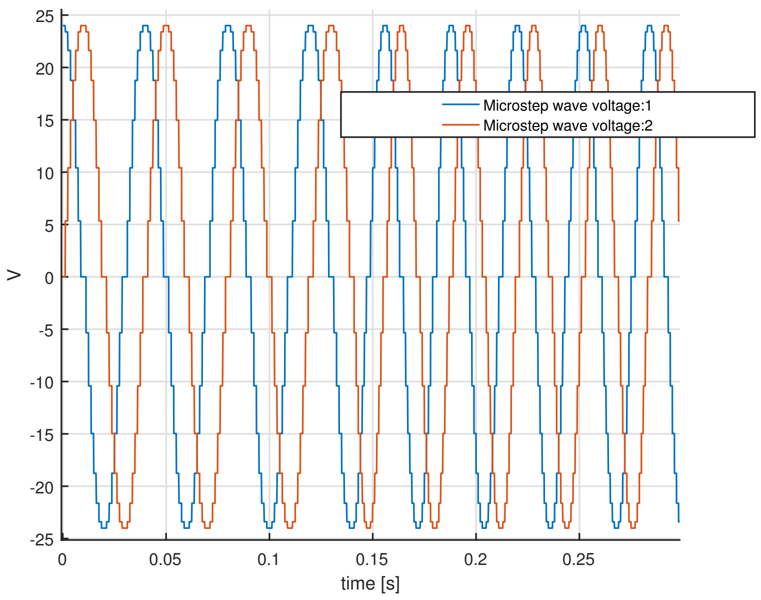


Figure 2.11: Current signals in Full Step mode at left, Half step mode in the center e Microstep mode at right

Full-Step and sine and cosine signals for the Micro-Step, also with variable frequency. These are not chirp signals, since the frequency must be varied every quarter of a period in order to realize linear speed ramps in real time. For a qualitative representation of the phase voltage signals for the two driving modes (full-step and micro-step), see Fig. 2.12.



(a) Matlab Plot of Phase Voltage Signals A and B at Full-Step Mode.



(b) Matlab Plot of Phase Voltage Signals A and B in Microstep Mode at one eight.

Figure 2.12: An example of voltage signals in input to the Simulink model of stepper motor.

2.3 Driving Logic

In the following section, the logics implemented for driving the stepper motor through the drafting of pseudocodes is presented.

The following pseudocode for driving the stepper motor has been realised:

```
Start code
Insert the acceleration
Insert the maximum velocity
Calculation of trajectory ($dt_i$) and theoretical number of steps $N_{th}$
profile realization
if $time > dt_i$ then
    Update velocity of PWM
    Update time interval $dt_{i+1}$
    Update $time$
end if
cicling...
if Read end-stroke then
    Measure number of steps $N_m$
    Calculation of error: $N_m - N_{th}$
    Adaptive Code
    Initialization
    Calculation of trajectory
end if
and so on ...
```

After launching it, it asks the acceleration and the maximum ramp speed in input. Then, it calculates the relative time intervals dt_i at which to increase speed, so to obtain the desired trajectory and the number of theoretical steps to be taken. When the time instant is greater than the dt_i, the frequency of the PWM is increased, and the next interval of time is considered. During the motion, the code works, and the PWM is updated with all the frequencies of the ramp. After n cycles, the carriage reaches the limit switch. Then, the code calculates the number of steps taken and the error made. Next, it activates the adaptive control function. Finally, it recalculates the trajectory for the new stroke, and starts again.

The pseudocode of the adaptive driving logic is reported:

```
Start adaptive code
if $error < limit\ value$ then
    Stepper motor not fails
    if $index_1 = pari$ then
        Increase acceleration
        $index_2$ write dispari
    else
        if $index_1 = dispari$ then
            Increase maximum velocity
            $index_2$ write dispari
        end if
    end if
    $index_1 ++$
end if
if $error > limit\ value$ then
    Stepper motor fails
    if $index_2 = pari$ then
        Decrease maximum velocity
    else
        if $index_2 = dispari$ then
            Decreas acceleration
            $index_1$ write dispari
        end if
    end if
    $index_2 ++$
end if
Calculation trajectory
```

In Fig. 2.13 the block diagram representing the operating principle of the adaptive driving logic is shown.

Once the limit switch is reached, the number of actual steps taken is calculated. Afterwards, it is possible to calculate the error, made between the number of theoretical and measured steps, through which the adaptive-control function modifies the trajectory. As it can be seen from the pseudocode above, if the error made is less than the limit value, the acceleration is increased and, if this occurs again, the speed is increased. For this reason, an index $index_1$ is used, which can take an even or odd

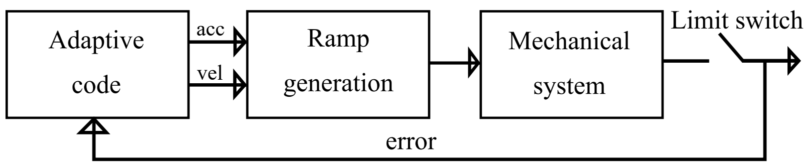


Figure 2.13: Block Diagram of the Adaptive Driving Logic.
The adaptive code receives the error as input, and based on it, the code sets the acceleration and the maximum speed to generate the ramp.

number. The index is increased every time it reaches the switch.
Notice that the $index_2$ belonging to the next case is immediately set as an odd number. Therefore, the code first reduces the acceleration in case of fail. Since the load curve decreases as the frequencies increase, it is possible that, for a given frequency, the motor will return to safe conditions reducing the acceleration. On the other hand, if the error persists, the velocity is reduced. Once the main parameters have been modified, the trajectory is recalculated and the motor restarted.
A qualitative example of what is expected from the code is shown in Fig. 2.14.

In conclusion, a simple schematisation of the working principle of the real code written in C has been given. It will be tested in Chapter 5, where the results of the experimental tests carried out in the laboratory are reported.
In the next chapter, the kinematics of the mechanical system on which the driving logic is based will be discussed.

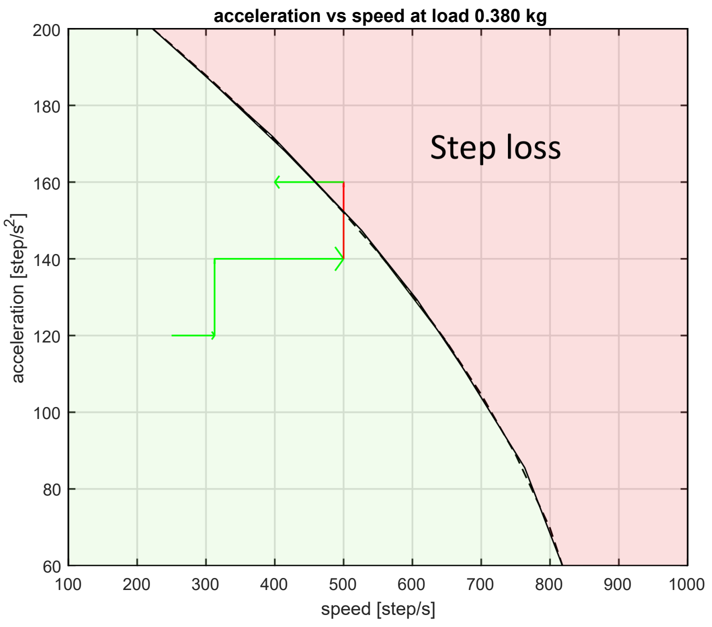


Figure 2.14: Qualitative representation of the adaptive driving logic working principle.

Chapter 3

Trajectory Planning and Kinematics

The aim of this chapter is to motivate the choices made, and how the trajectory can be implemented into the system. In Section 3.1, two ways of implementing speed profiles are discussed. Section 3.2 studies the kinematics of the mechanical system. The fundamental parameters that affect its dynamic behaviour are thus revealed.

3.1 Implementation of Speed Profiles

In order to move mass through a stepper motor, it must be noted that this actuator is sensitive to the starting and stopping phases, especially for heavy loads. Therefore, in order to avoid loss of steps or stall during the starting motion, it is necessary to create a linear speed profile, which allows the mass to start slowly and then speed up gradually. For this purpose, after setting the desired acceleration, two methods for the realisation of speed ramps are considered. Following this, comparison between them and the reasons for these choices are discussed.

3.1.1 Real-Time Speed Profile

This speed profile is obtained through an appropriate iterative algorithm, based on the physical working principle of stepper motors. This drive receives as input a train of pulses at a certain frequency and duty cycle, by means of which the rotor makes successive rotations of fixed angular step. As such, it generates an incremental movement, or discrete from a temporal standpoint.

Some kinematics fundamental parameters are listed below:

• Stepping-rate/frequency	f $[step/s]$;
• Speed	v $[rad/s]$;
• Acceleration	a $[rad/s^2]$;
• Step time/delay	δt $[s]$;

The rotor's rotational speed increases as the number of pulses received per second rises – that is, when the stepping rate increases. On the other hand, if the frequency grows, the time period between one pulse and the following is reduced. The algorithm's objective is to calculate the delay between consequent impulses moment by moment, so that the resulting velocity profile is a linear ramp. To implement that, the following references have been consulted [15, 12]. The time delay between one pulse and the next one is calculated by considering that, at each instant, the speed increases producing a rotation of a fixed angle α. The following relationship is thus evaluated:

$$\delta t_i = \frac{\alpha}{v_i} \tag{3.1}$$

Considering two time instants t_i and t_{i+1}, with good approximation, it is possible to say that the speed increases almost linearly according to the following relation:

$$v_i = v_{i-1} + a\delta t_{i-1} \qquad i \geq 1 \tag{3.2}$$

If the code is implemented by a microprocessor, it is useful to consider the so-called counter c_i, which counts the number of pulses at a given frequency f. This makes it possible to write:

$$\delta t_i = c_i/f \tag{3.3}$$

Taking into account the equation (3.1) and combining it with (3.2), the iterative algorithm is obtained.

$$c_i = \frac{\alpha f}{v_i} = \frac{\alpha f}{v_{i-1} + a\ \frac{c_{i-1}}{f}} \tag{3.4}$$

In order to execute the code, it is necessary to know the initial value of the counter c_0. To determine it, the first impulse has to be considered as having initial speed of zero. Therefore, the angular position is given by:

$$\alpha = 1/2\ a\ \delta t_0^2 \tag{3.5}$$

The speed at time istant t_1 is calculated as:

$$v_1 = a\delta t_0 = \sqrt{2\alpha a} \tag{3.6}$$

If a mean value between v_0 and v_1 is considered, the relation (3.6), combined with (3.4), allows to evaluate the value c_0.

$$c_0 = f\sqrt{\frac{2\alpha}{a}} \tag{3.7}$$

The code is thus completed and ready to operate. It requires the acceleration and the maximum speed to reach with a fixed number of steps in input. The number of steps also represents the number of iterations required for the algorithm. They are calculated keeping in mind that the integral of speed represents the displacement. The area of the triangle, or the ramp, divided by the fixed step α equals the number of steps.

$$N = \frac{1}{2}\,\frac{t_{acc}\,v_{max}}{\alpha} \tag{3.8}$$

A qualitative example is represented by Fig 3.1. Note that, between the reference speed and the one obtained with the algorithm, there is a certain offset. The latter can be reduced with appropriate corrections, but a small percentage of error remains.

The algorithm is written in C. It is then loaded in the Raspberry Pi card, running on Linux OS. For the following step, a small test bench is used, where no load and no other mechanical components are taken into exam. This is characterised by a stepper motor, the relative driver, and the card with a microprocessor. After starting the code, the motor runs a perfectly working ramp. If the micro-step mode is activated, the motor rotation results more fluid and precise.
Even if it represents a high-potential solution, the velocity profile is not realised in this thesis work with this method. The reason being the fact that, in order to realize the pulse train, C presents a function called *nanosleep*, which suspends the microprocessor's work for fractions of second equal to the time delay between one impulse and another. This was deemed not acceptable, since the application requires a communication system between the considered mechanical system and the central network, which simultaneously manages other multiple complex systems, the threads of which cannot be suspended. Moreover, the trajectory must be discretized with a certain sampling time. Therefore, the stepper motor driving is realised by writing a code, in which the pulse train is directly sent to

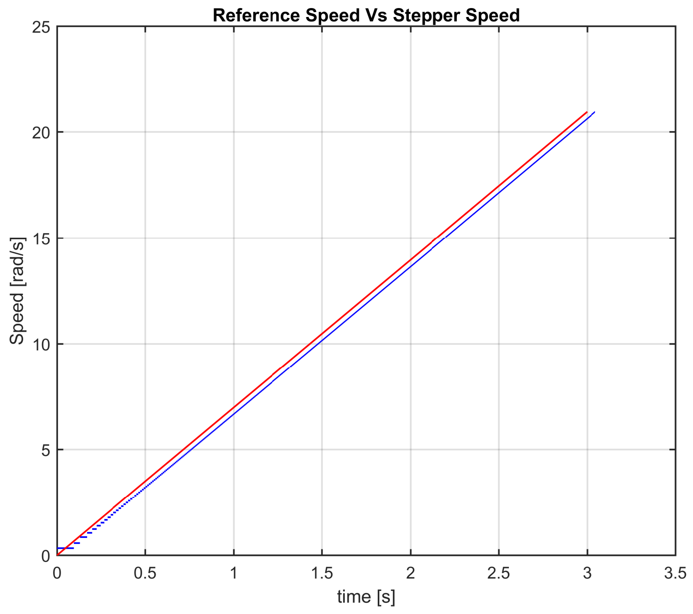


Figure 3.1: Matlab Plot: Reference Speed vs Stepper Speed generated by Real-Time Algorithm: Speed profile of stepper motor in blue. Reference speed profile in red.

the actuator by the PWM Software. This one can be activated at the beginning of the code, making it work independently of the other threads. However, it is important to update the frequency of pulse train when the speed of the ramp increases at the correct time.

3.1.2 Speed Profile with PWM Software

Since the aforementioned method was not optimal for this project, the design and implementation of another code is considered. This one is also written in C using PigPio library functions, (references on the topic can be found in [17]). The train of pulses is realized by PWM Software, which could only be exploited for fixed imposed frequencies and is dependent on the clock time set in the code. Under these conditions, the ramp profile presents the following stepping-rates:

$$ramp = [800, 1000, 1250, 1600, 2000, 2500, 4000, 5000, 8000] \quad [mustep/s] \tag{3.9}$$

Where $[mustep/s]$ refers to the active state of the micro-stepping mode; in this case one eighth. Therefore, a fraction of step is obtained. In order to reach the same velocity, the motor will do more step per seconds, precisely the value m of micro-steps multiplied by the stepping rates. In this way, a simple stepped ramp is obtained, which is acceptable for the present application. However, there are some issues due to the limitations of PWM Software. In fact, as in the previous method, there are no possibilities to increase the velocity instant by instant, but only in time intervals dt. These are calculated by trying to maintain the acceleration constant. Therefore, having set the acceleration, it is possible to consider the following relation to calculate the time intervals:

$$dt_i = \frac{V_{i+1} - V_i}{A} \qquad i = 1, 2, \dots, n-1; \tag{3.10}$$

Where n refers to the length of the vector *ramp*. The number of steps computed depends on the stepping rate, which permits to calculate the time pulse t_{pulse}, and the time interval according to the following relations:

$$N = \frac{dt}{t_{pulse}} \tag{3.11}$$

$$t_{pulse} = 1/f \tag{3.12}$$

The PWM works properly if the motor performs an integer number of steps. Therefore, the time interval dt must be a multiple value of t_{pulse}. In the C algorithm, is implemented a code where, if the number of steps is not integer, the time intervals dt are corrected. However, the acceleration is also varied, as can be seen in the Fig. 3.2b, although its value remains close to the initial one

The next section presents the kinematics of the mechanical system. It also presents an example of trajectory planning realised as such.

3.2 Mechanical System Kinematics

In this section, the study of the kinematics of the mechanical system as introduced in 2.2.1 is used.
It is important to introduce two hypotheses on the system under examination. In the first, the angular velocity is transmitted from the drive shaft to the pulleys by means of rigid joint. The other one concerns the belts connected to the cart, which are in tension, so that the effect of elasticity can be considered negligible. They can be considered as inextensible rope. Under these assumptions, the fundamental kinetics parameters are introduced considering a practical example.

An acceleration value of the cart equal to 250 $[step/s^2]$ and a maximum speed of 1000 $[step/s]$ are given as input to the algorithm for trajectory planning introduced in the section 3.1.2. Following this, using the code previously explained, the proper time intervals dt are evaluated. The shaft velocity could be expressed in [m/s] or in [rpm] by the use of the following relation:

$$v_{rpm} = \frac{60\ f}{200\ m} \tag{3.13}$$

$$v_{rad} = v_{rpm}\ \frac{\pi}{30} \tag{3.14}$$

Knowing the shaft velocity, the speed of the cart is evaluated. Since the belt and pulley transmission are connected with the shaft through a rigid joint, the velocity is calculated as:

$$v_{abs} = v_{rad}\ R_p \tag{3.15}$$

Where R_p is the primitive radius of the pulley. Since the PWM is updated each dt seconds, the time spent by the ramp is calculated as:

$$t_{rampa} = \sum_{i=1}^{n} dt_i \cdot f_i \tag{3.16}$$

Where n is the size of vector rampa, which gives the stepping-rates.
When the time needed to perform the speed profile is elapsed, it is possible to estimate the time spent by the cart to reach the end stroke with the following relations:

$$S_t = \frac{1}{2} t_{rampa} v_{max} + v_{max}\left(t_{corsa} - t_{rampa}\right) \tag{3.17}$$

$$t_{corsa} = \frac{S_t}{v_{max}} - \frac{1}{2} t_{rampa} + \frac{t_{rampa}}{v_{max}} \tag{3.18}$$

Where the relation (3.17) refers to the cart displacement during the trajectory (the length of the stroke), which was measured in the laboratory. The results obtained from the practical example are collected in Tab. 3.1. The speed profiles obtained are shown in Fig. 3.2 and Fig. 3.3.

Table 3.1: Kinematics data

Quantity	Unit	Module
t_{rampa}	s	3.61
t_{stroke}	s	3.91
S_t	m	0.38

The time elapsed to reach the limit switch is therefore measured through a code. The time measured can be used to calculate the real number of pulses made during the stroke.

This number is calculated as:

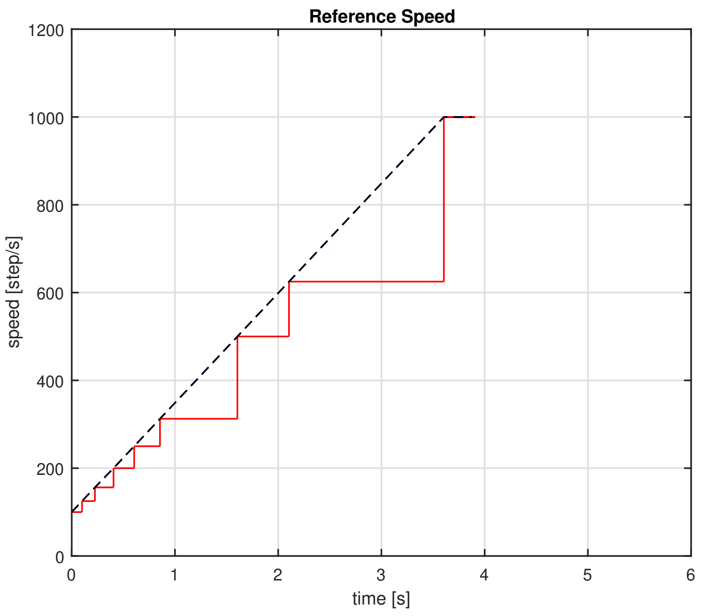
$$N_{rampa} = \sum_{i=1}^{n-1} f \cdot dt \tag{3.19}$$

Where f is the stepping rate in $step/s$, and dt is the time interval spent to update the speed in seconds. The number of steps made at constant velocity is evaluated with the following relations:

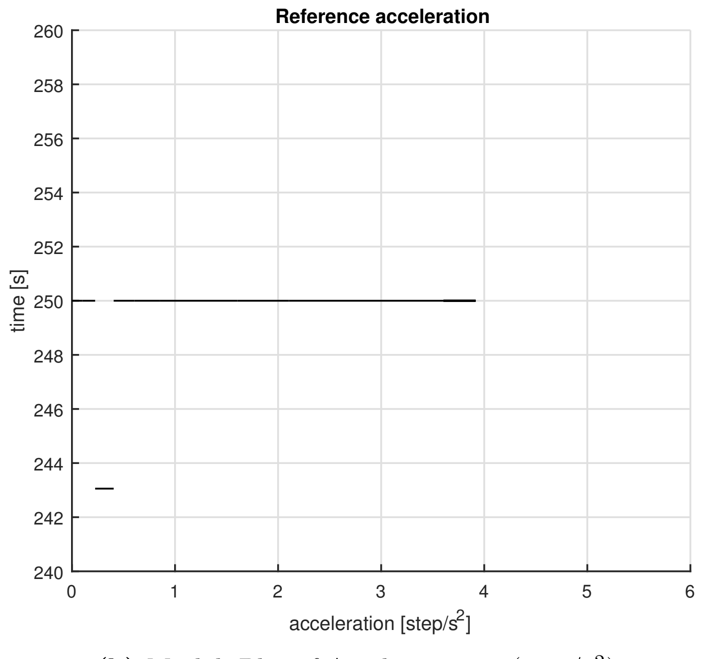
$$N_{cost} = (t_{corsa} - t_{rampa}) * f_{max} \tag{3.20}$$

$$N_{corsa} = N_{rampa} + N_{cost} \tag{3.21}$$

The main property of stepper motors is the possibility to carry out motion with a series of fixed steps at a certain frequency. However, if the velocity imposed to the cart transporting a heavy load is too high, the motor might lose steps. This compromises the motion by causing undesirable slippages. To avoid this, once the number of measured steps is higher than the ideal one, it is necessary to decrease the acceleration and maximum velocity. Whence, the two important parameters are acceleration and maximum speed.
In conclusion, the study of the mechanical system's kinematics allows to develop a non-traditional feedback based on the comparison between the measured number of steps and the theoretical one. If the error made is higher than a limit value (yet to be defined), the loss of steps may occur. Finally, the two parameters to be changed by the adaptive driving logic in order to prevent step loss have been highlighted.
In the next chapter, the numerical analysis performed on the mechanical system, driven by both trajectories, will be presented. In chapter 5, the limit value of the error, based on the results of the experimental tests carried out in the laboratory, will be estimated.

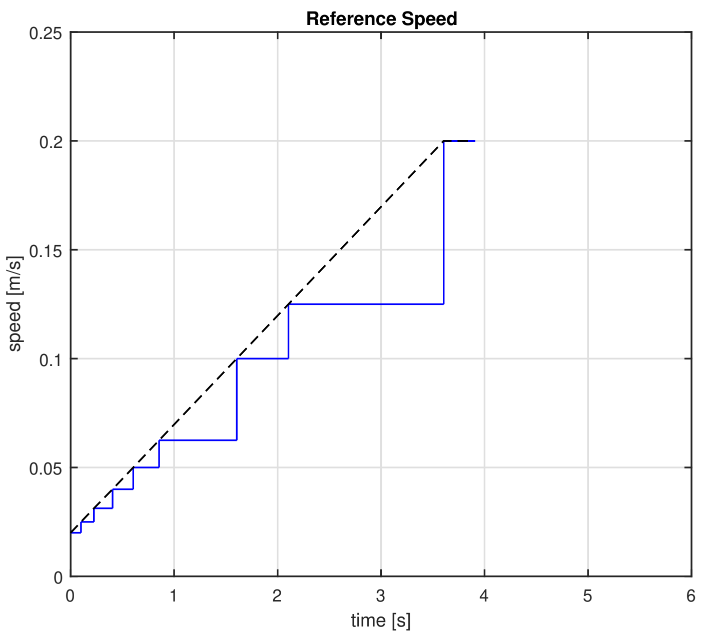


(a) Matlab Plot of Speed Ramp in $(step/s)$.

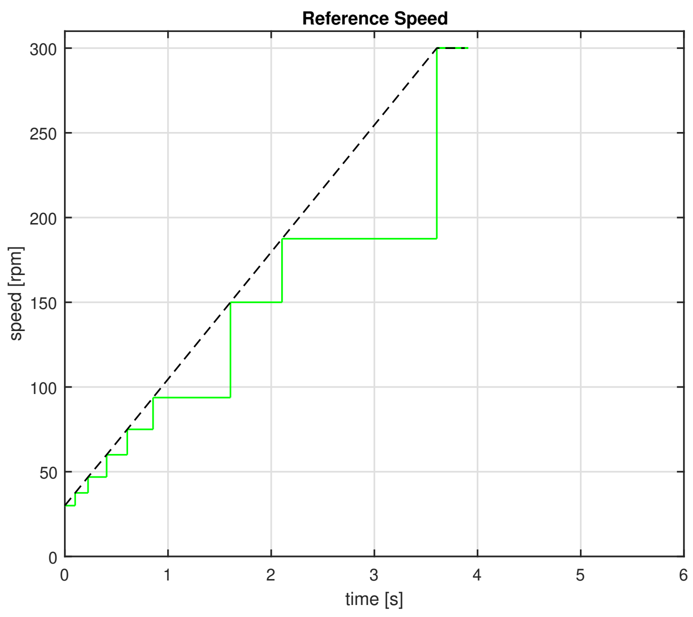


(b) Matlab Plot of Acceleration in $(step/s^2)$.

Figure 3.2: An Example of Speed Profile: in figure(a) the dotted line stress the linearity of the speed profile.



(a) Matlab Plot of Speed Ramp in (m/s).



(b) Matlab Plot of Speed Ramp in (rpm).

Figure 3.3: An Example of Speed Profile: In the two figures, the dotted line stress the linearity of the speed profile.

Chapter 4

Numerical Analysis

The results of the simulations carried out considering the theoretical model examined in section 2 are presented in this chapter.

In the present dissertation project, given the unfeasibility of using sensors, the theoretical model has not been validated. Therefore, the results of these simulations cannot be compared with the experimental ones to give proof for the loss of steps. However, the simulations were carried out to show the potential of the model, and to give an idea of the mechanical system behaviour. The simulation software chosen is Simulink. Suitable for modelling, simulation and analysis of dynamic systems, this tool was developed by the American company MathWorks. The open-loop block diagram is represented by Fig. 4.1. The block called *Trajectory* is responsible for the realisation of the reference trajectories. In input, it receives acceleration, maximum speed of the ramp, the micro-step fraction. Given the trajectory in input, the block called *Driver*, realises proper phase-voltage signals suppling the motor. Following this, the block diagram of the stepper motor generates the cart position at the output.

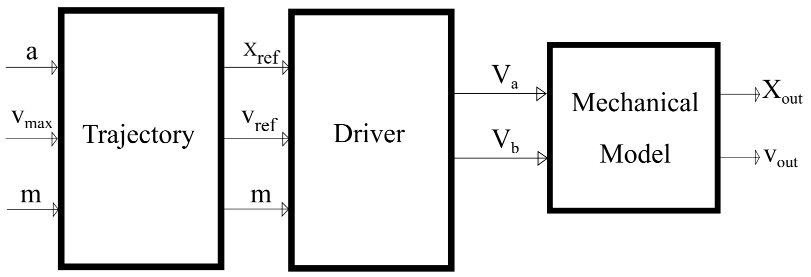


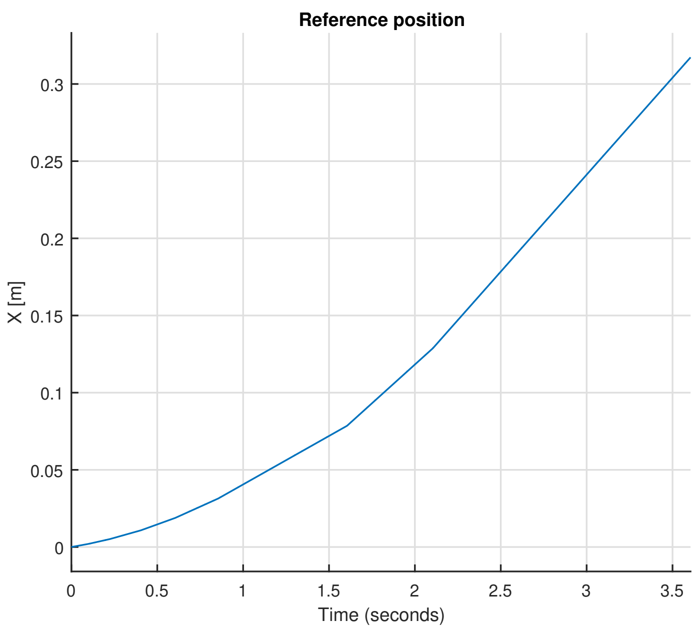
Figure 4.1: Open-loop Block Diagram.

Simulations of PWM Software Speed Profile

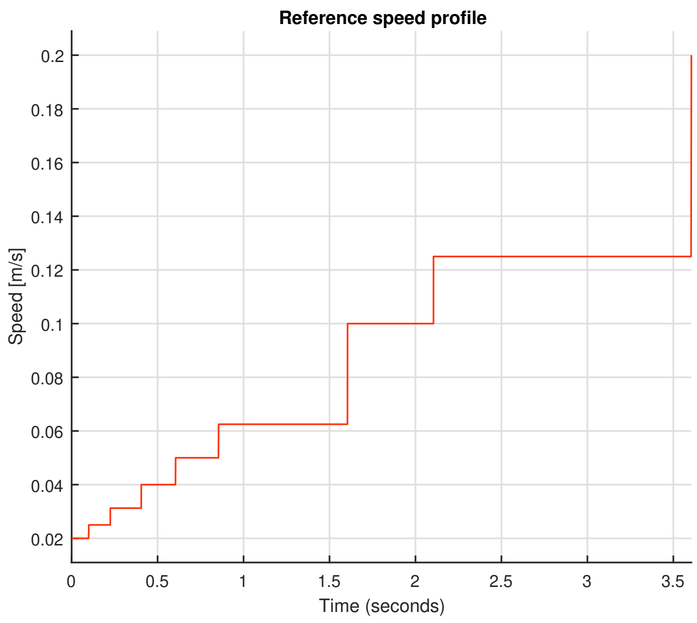
These simulations are carried out by setting as a solver ode-4 (Runge-Kutta). On the other hand, the iteration step is fixed with a sampling time of $t_s = 10$ [μs]. The Embedded Matlab function *Trajectory* deals with the realisation of the stepped speed profile by setting the acceleration $a = 250$ [$step/s^2$], the maximum speed $v = 1000$ [$step/s$], and the micro-step fraction set at one eighth. The results of the simulation are shown in Fig. 4.2. The simulation time is equal to the time required to execute the ramp, that is $t = 3,605$ [s]. Taken the velocity profile as reference, the motor model outputs the position assumed by the cart, as shown in Fig. 4.3. If the output is compared with the aforementioned reference, an error is obtained. As shown in Fig. 4.4, after covering 0.3 [m], the maximum error is about 4[mm], which, after being divided for n steps, is still very limited, thus demonstrating the validity of the micro-step. The other mechanical quantities under examination are reported in Fig. 4.5. The mechanical angular velocity shown in Fig. 4.5a respects the stepped profile. The phase currents shown in Fig. 4.5c, are reduced in form as speed increases, and present discontinuities caused by the rapid change of speed. The mechanical torque oscillations are reduced in module when speed increases. This does not guarantee path accuracy for high load torques.

Simulations of Speed Profile with Iterative Code

The model of the mechanical system is simulated under the same conditions as the previous case. This is done by applying the reference trajectory generated through the iterative algorithm. The results obtained are shown in the following Fig. 4.6. The mechanical system generates the position of the cart represented by Fig. 4.7. It can be noted that, compared to the previous case, the position curve is softer without changes in slope. In fact, the error made on the reference is smaller than in the previous case, as shown in Fig. 4.8. Other mechanical quantities under examination are reported, as shown in Fig. 4.9. The mechanical angular velocity is shown in Fig. 4.9a. It follows a linear trend. The phase currents shown in Fig. 4.9c are reduced in form as the speed increases. The mechanical torque oscillations are reduced in module as the currents are reduced. Hence, the path accuracy for high load torques is not guaranteed.



(a) Simulink Plot of Reference Position X_{ref} (m).



(b) Simulink Plot of Reference Speed (m/s).

Figure 4.2: Simulation of Trajectory planned with PWM Software.

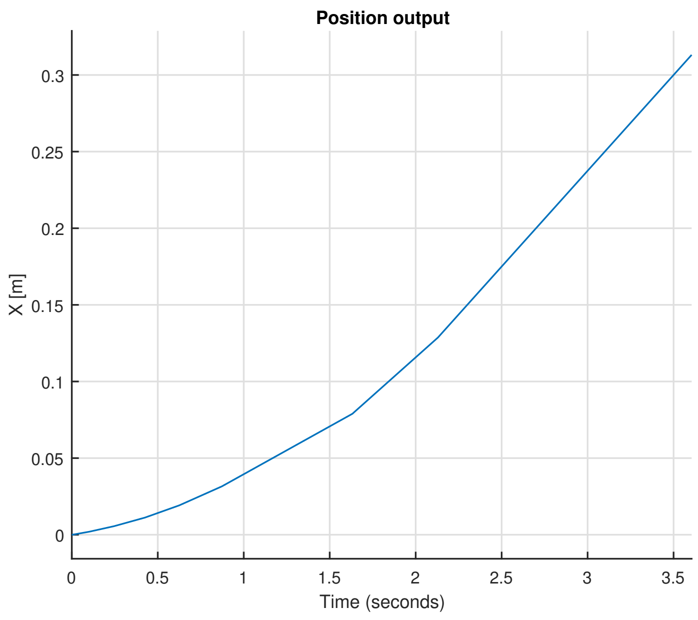


Figure 4.3: Simulink Plot of Position Output X_{out} (m) of Trajectory planned with PWM Software.

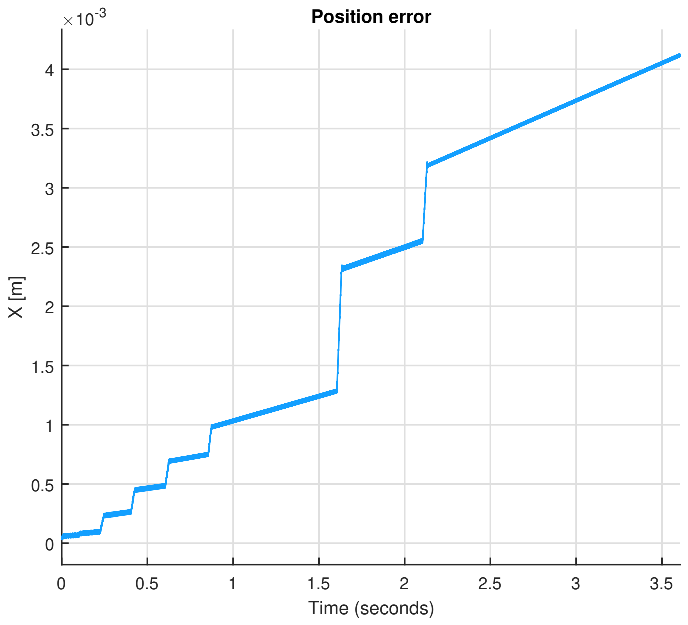
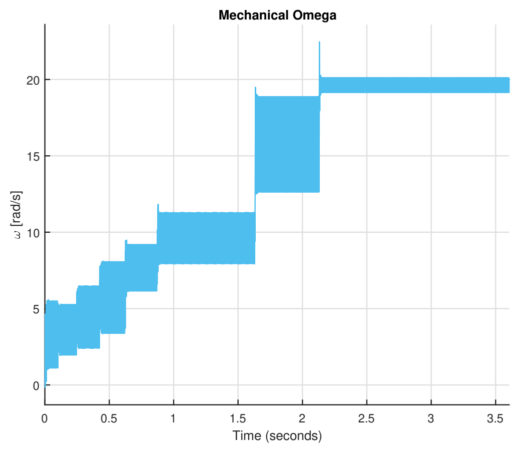
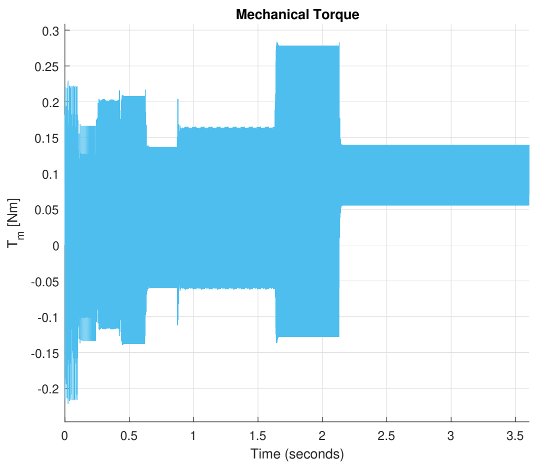


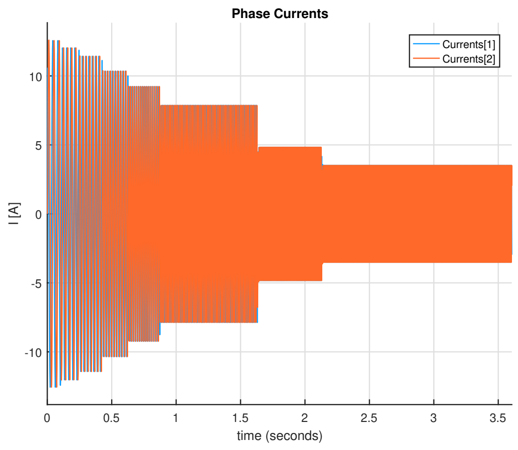
Figure 4.4: Simulink Plot of Position Error (m) of Trajectory planned with PWM Software.



(a) Oscillations ω (rad/s).

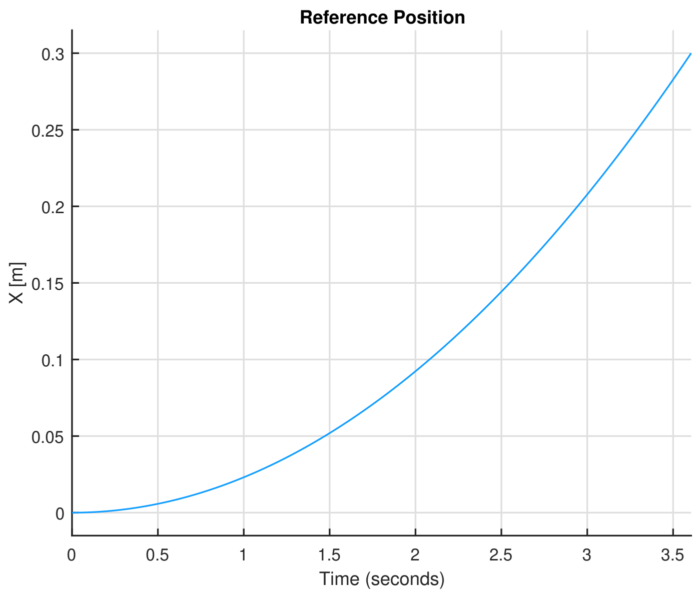


(b) Oscillations of the Mechanical Torque T_m (Nm).

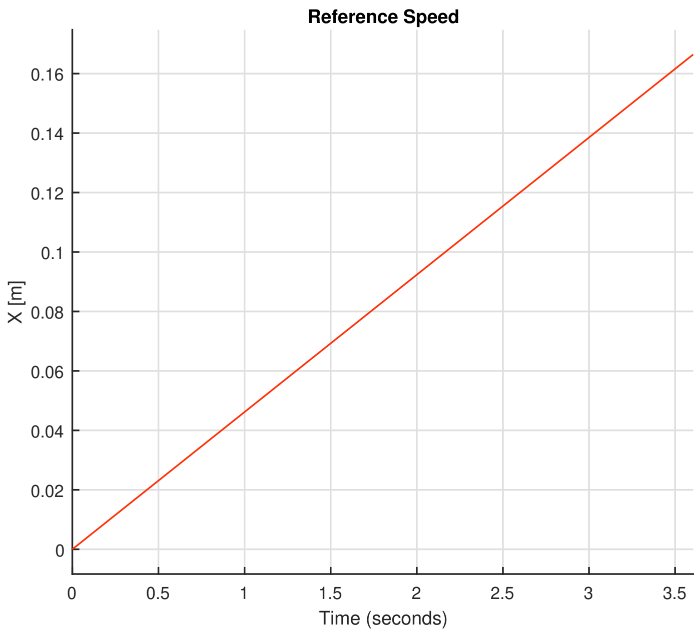


(c) Phase Currents: I_a (A) in blue and I_b (A) in red.

Figure 4.5: Simulink Plots of Trajectory planned with PWM Software.



(a) Simulink Plot of Reference Position X_{ref} (m).



(b) Simulink Plot of Reference Speed in (m/s).

Figure 4.6: Simulation of Trajectory planned with Iterative Algorithm.

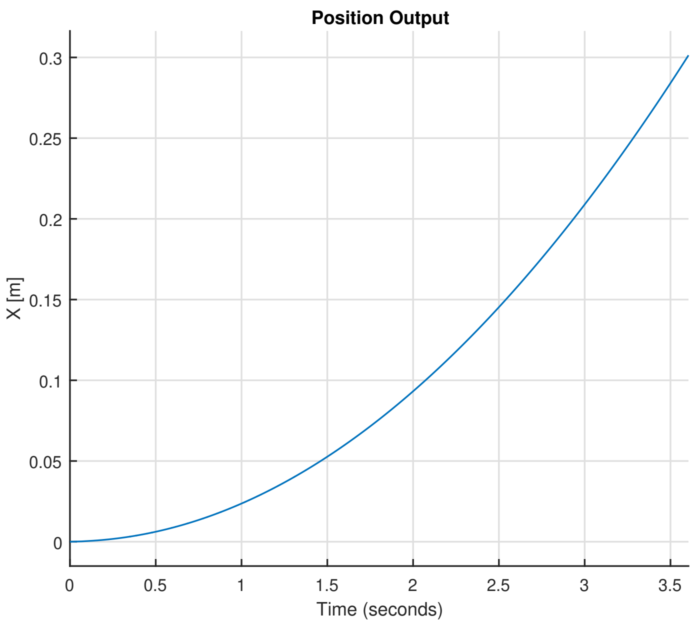


Figure 4.7: Simulink Plot of Position Output X_{out} (m) of Trajectory planned with Iterative Code.

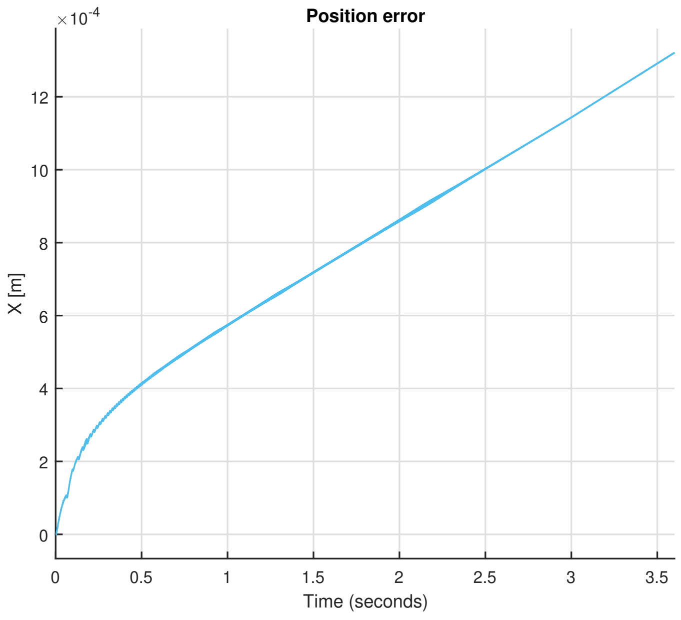
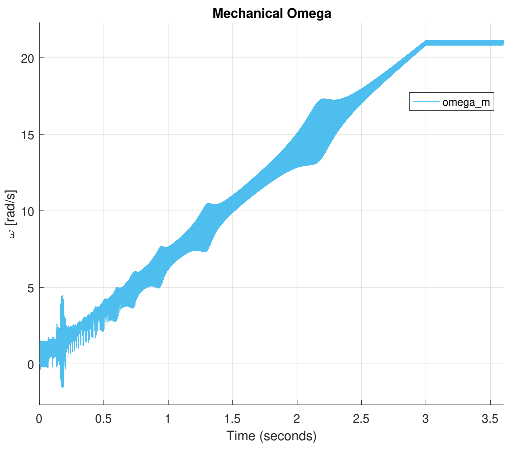
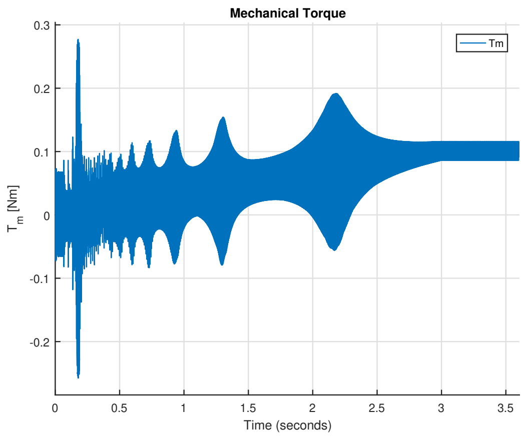


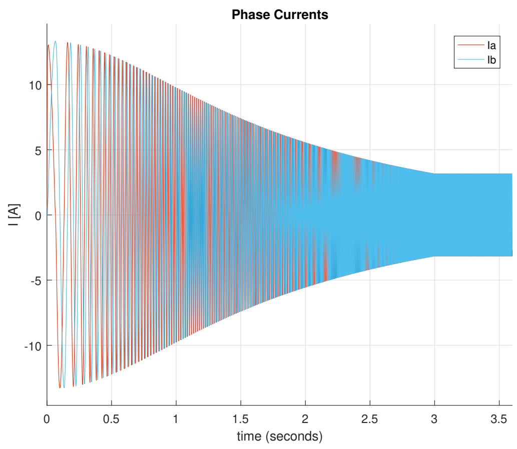
Figure 4.8: Simulink Plot of Position Error (m) of Trajectory planned with Iterative Code.



(a) Oscillations of the Mechanical Angular Velocity ω (rad/s).



(b) Oscillations of the Mechanical Torque T_m (Nm).



(c) Phase Currents: I_a (A) in blue and I_b (A) in red.

Figure 4.9: Simulink Plots of Trajectory planned with Iterative Code.

Chapter 5

Experimental Tests

The aim of this thesis is to realise an adaptive algorithm capable of change the trajectory in a way that would prevent step losses with variable and unknown loads.

In order to realise the code properly, it is necessary to understand whether step losses occur. Sensors have not been employed in the present thesis project. The electric measurements registered with an oscilloscope are not useful due to the presence of noise. Therefore, it has been decided to observe carefully the mechanical system during the motion.
To do that, some experimental tests have been carried out. By increasing the load and the velocity, the cart should present vibrations, slips and metallic noises. These conditions confirm step loss. Following this premise, a proper code for the experimental tests can be written.
Its operating principle was introduced in section 2.3. It calculates the number of steps theoretically needed to performed by the trajectory, as explained in 3. The code then compares this number with an estimate of the actual number of steps, which is obtained by taking into account the measures of time spent by the cart to complete the stroke. The code recognises the step loss from the error made. In order to properly correct the trajectory, the algorithm changes two variables, acceleration and maximum velocity. The experimental tests are performed by inserting a mass into the cart, and then driving it at different speeds and accelerations via an open-loop control mode. Once the cart reaches the end-stroke, if the algorithm records no step loss, it will increase the cart's maximum velocity for the following stroke. If it does not fail, the acceleration is also augmented in the next cycle. Conversely, if the code records a step loss, it reduces the acceleration for the next stroke. If this situation is repeated, the velocity is decreased in the following cycle. Once the step is lost, it is

expected that the code is able to return the cart to safety conditions, and that is keeps oscillating around an optimum point.

However, some issues arise. First, the uncertainty affects time measurements. Tests made about this issue demonstrate that uncertainties depend on the code written to prepare the ramp profile. The time intervals needed to upgrade the speed present an error of several microseconds. Another factor is the implemented PWM software, which presents delays of microseconds in updating pulse frequency. Other problems could be caused by the structure of the code and the time elapsed to recall the functions. Finally, there are effects generated by the elasticity of the belt, which are not considered in this thesis. The elasticity of the belt could compromise the precision of the movement.

5.1 Experimental Results

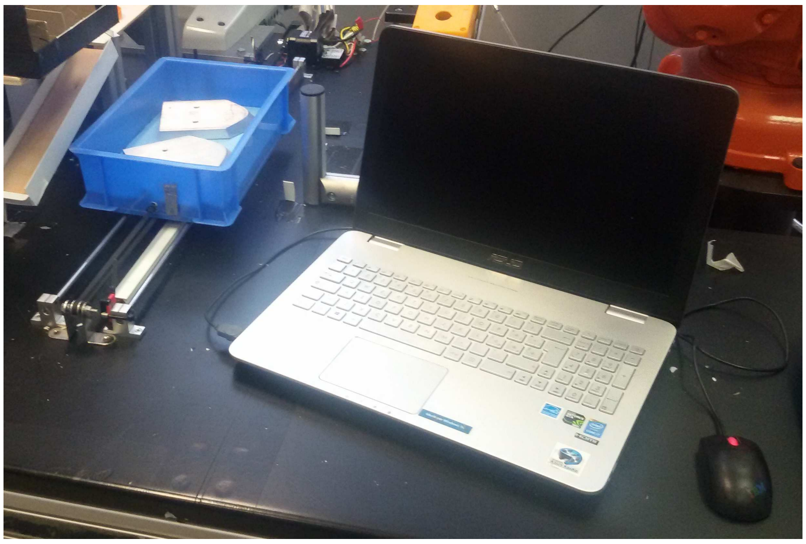


Figure 5.1: Experimental Setup.

The experimental tests were carried out in the Group MERLIN Mechatronics and Robotics Laboratory.
For the first set, a load $p = 0.380\ kg$ was chosen to be inserted into the cart. Then, by setting different accelerations and maximum velocities of the profile, the cart is moved from an end-stroke to another. Once the latter reaches the limit switch, the number of steps are measured, and the

number of theoretical steps are calculated, as explained in Section 3.2.
In each test, the maximum velocity and the acceleration are increased. For any combination of acceleration and maximum velocity, the numbers of steps measured are registered into an Excel sheet for both empty and loaded carts. Having observed the results recorded, it is possible to notice the presence of uncertainty. Hence, step loss is difficult to understand from the error made.
The last possibility to understand motor failure is to carefully observe the motion of the cart and look for slips or metallic noises. In fact, when acceleration increases at fixed speed, the number of steps performed decreases. After making the tests for any combination of speed and acceleration, a curve was obtained. This is represented in Fig. 5.2.
On the y-axis, the acceleration is in $[step/s^2]$, while on the x-axis the velocity is in $[step/s]$. From the figure, it is possible to see that the curve is outlined by the points in which the stepper motor fails. It is quite similar to the characteristic curve of the motor. Therefore, it could be employed to realign the code to the behaviour of the mechanical system.
From these results, it is possible to notice a repeated fixed error that increases alongside the velocity. It is possible to compensate for the former by doing an initial calibration with a series of experimental tests using an empty cart. Once the algorithm is realigned with the behaviour of the empty system, the error made is caused by the step losses, with the presence of $\pm$ 2 step uncertainty.
Therefore, the minimum error value could be set 3. The implemented code is thus able to modify the trajectory planned for the following stroke, in order to avoid step losses.

The next experimental tests were carried out with the aim of validating the adaptive driving logic developed.

Experimental Test with 0.380 kg

In this phase of experimental tests, an acceleration value of $a = 100$ $[step/s^2]$ and a maximum velocity of $v = 200$ $[step/s]$ were given as input to the algorithm. After Setting these initial values, it is ensured that the motor does not lose steps. Then, the micro-stepping mode at one eighth of step was activated. In this way, the trajectory planned at the initial state generated a slow travel of the cart and a fluid start.
Once the end-stroke is reached, the speed is increased to $v = 250$ $[step/s]$ while the acceleration is kept constant. Cycle after cycle, the two parameters are modified as shown in Fig. 5.3.

The adaptive driving logic continues to increase the performance of the speed profile, minimising the time cycle until the motor does not lose steps. As represented in Fig. 5.3a, the motor loses steps at a maximum velocity of $v = 625$ $[step/s]$, and acceleration of $a = 160$ $[step/s^2]$.
However, once the end-stroke is reached, the algorithm reshapes the trajectory and brings the cart back to safe conditions in the following stroke, as is shown in Fig.. 5.3b.
Then, the code continues to modify the two parameters, remaining in the area of the qualitative characteristic curve intercepting an optimum point in $v = 625$ $[step/s]$ and $a = 140$ $[step/s^2]$.
The comparison between the experimental curve, represented in Fig. 5.2, and the results just obtained prove the validity of the code.

Experimental Test with 0.215 kg

The same test was conducted with a load of $p = 0.215$ [kg]. Since the load inserted into the cart is lower than the previous case, a higher load curve was expected. The stepper motor should fail at higher maximum velocity and acceleration values. The kinematics parameters given as input to the algorithm, at initial state, are the same as the previous case. The results obtained are represented in Fig. 5.4. It is possible to observe that the motor loses steps at a maximum speed profile of $v = 1000$ $[step/s]$ and acceleration of $a = 180$ $[step/s^2]$. However, the load curve is only a qualitative representation. Once the step is lost, the algorithm modifies the trajectory and brings the cart back to safe conditions. Then, it continues to orbit around an optimum point equal to $v = 625$ $[step/s]$ and $a = 180$ $[step/s^2]$.

Experimental Test with 0.526 kg

The last test of the code was carried out with a load of $p = 0.526$ [kg]. In this case, the total weight is higher than that of the first test performed. It is therefore expected that the motor fails at maximum speed and acceleration lower than in the previous cases. The kinematics parameters given as input to the algorithm, at initial state, are the same of the previous experiment. The results obtained at the end of these laboratory test are shown in Fig. 5.5. The cart loses steps already at a speed of $v = 312.5$ $[step/s]$ and acceleration of $a = 120$ $[step/s^2]$, thus confirming the initial assumptions. Once the step is lost, the algorithm modifies the parameters, generating a velocity profile that orbits around an optimum point of $v = 250$ $[step/s]$ and $a = 120$ $[step/s^2]$.

5.2 Discussion about the Present Tests

The aim of this thesis is to realise an adaptive driving logic able to revise the trajectory in accordance with the step loss without the use of sensors and with a variable load. Observing the results of the tests, the goals seem to have been met. In the first part of the tests, since the load curve could not be measured, a decision was taken to estimate the curve acceleration vs speed at fixed load. This represents a good approximation of a load curve. Therefore, a series of experimental tests with variable maximum velocity and acceleration were performed. The results show that the test target was met. The measurements taken seem to reconstruct the load curve sought. It should be noted that the speeds imposed by the PWM Software are a limiting factor for the execution of these tests. Hence, the curve has empty spaces. As for the tests performed on the adaptive code, the expectations were also met. The results show that the code, even with some variability, is able to modify the trajectory of the cart to return it to safety conditions and thus staying in the area of an optimum point. Under this condition, the speed profile has the minimum time cycle without losing steps. At this point, costs are minimised and productivity maximised.
Then, the tests, carried out by inserting higher or lighter loads, demonstrate that the code is able to reconstruct load curves also in different scenarios. Negative aspects, however, are the presence of uncertainty in the time measurements, which compromises the calculation of the number of steps made. Another aspect is the impossibility of checking for step loss. This is done by observing the behaviour of the system during these experimental tests.

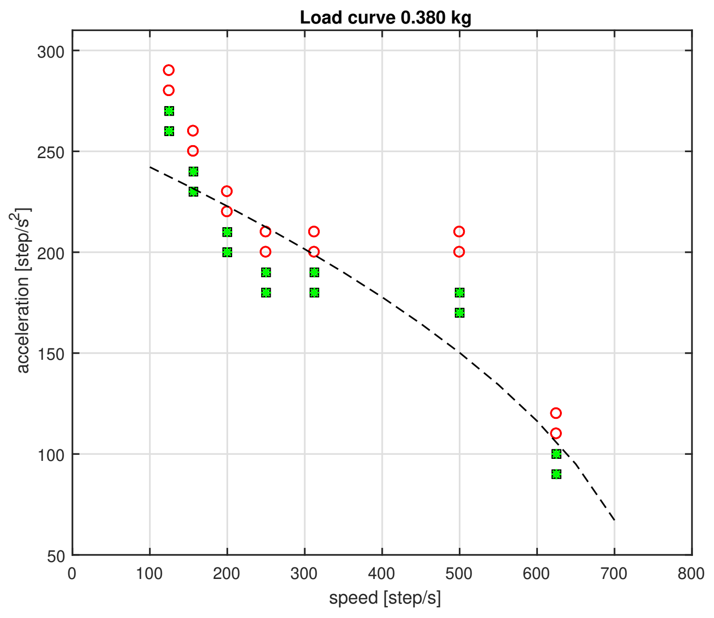
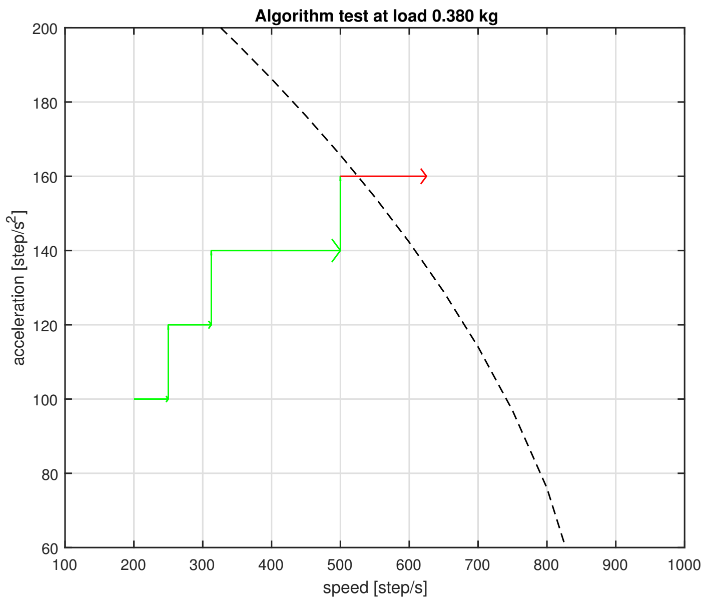
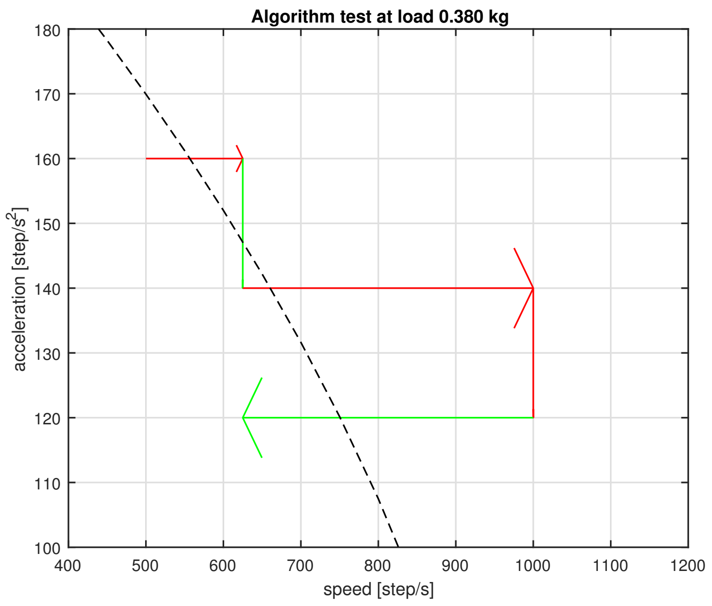


Figure 5.2: Experimental Test with Load of 0.380 kg.
○ Step loss; □ No step loss; the dashed curve represents the qualitative load curve.



(a) Execution Code up to loss of Steps.



(b) Code answer to the loss of Steps

Figure 5.3: Test Adaptive Algorithm with a Load of 0.380 kg: The dashed curve represents the qualitative load curve. The green arrows show safety condition, while the red arrow show step loss.

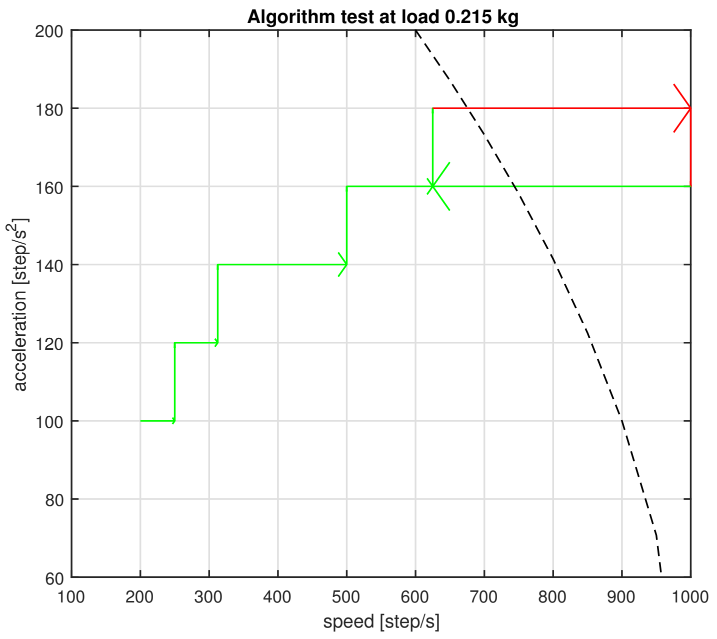


Figure 5.4: Test Adaptive Algorithm with a Load of 0.215 kg: The dashed curve represents the qualitative load curve. The green arrows show safety condition, while the red arrow show step loss.

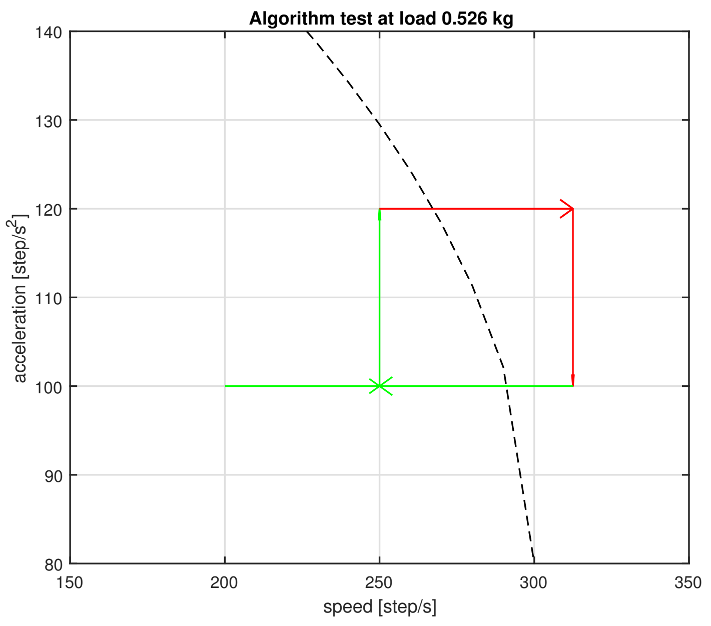


Figure 5.5: Test Adaptive Algorithm with a Load of 0.526 kg: The dashed curve represents the qualitative load curve. The green arrows show safety condition, while the red arrow show step loss.

Conclusions

This thesis project dealt with the experimental study of an open-loop control strategy, without the use of sensors, for a mechanical system set in motion by stepper motors.

In order to properly implement the control logic, it was necessary to study the different components of the mechanical system examined one by one. Its main features and characteristics have been identified. As far as the mechanical apparatus is concerned, its dynamic behaviour has been studied through the Lagrange equation. The same thing was done for the electric actuator. A system of non-linear differential equation was derived for it. The driver's behaviour has also been represented by embedded Matlab functions that reproduce the voltage signals of the full-step and micro-step modes. It was thus possible to implement a block diagram through which numerical analyses were performed in a Simulink environment. Knowing the characteristics of the project, it was possible to implement the adaptive driving algorithm. For the trajectory to be given as a reference to the system to execute motion, the speed profile most commonly used in these cases is the speed ramp. Two possible solutions have been examined, the profile realised by PWM Software and the one computed by an iterative algorithm. Numerical simulations were performed in Simulink on both trajectories. The results managed to detect useful information on the dynamic behaviour of the overall system. The profile realized with an iterative code is better in fluidity and precision than the one with PWM Software. However, it was decided to drive the stepper motor via PWM Software, seen that the trajectory is discretised according to the sampling time chosen. It is hence not possible to provide information moment by moment. The analysis of the kinematics considering the chosen trajectory allowed to calculate the number of theoretical steps performed by the cart to cover the path. Knowing the trajectory, it was possible to drive the cart in open-loop. Afterwards, an experimental test campaign was carried out in the test bench of the Merlin Group Mechatronics and Robotics

Laboratories, with the aim of validating the adaptive driving logic. To perform the tests, a C code was created. This presents functions capable of measuring the number of steps taken and the time spent by the cart to move along the path until the limit switch is reached. Tests were carried out by inserting a certain load on the cart, and by varying the acceleration and the maximum speed. These allowed to identify a curve very similar to the load curve of the system, which gives useful information on the loss of steps. Observing the results of the tests, a certain variability of the error made between the number of theoretical and effective steps was observed. Through other experimental tests on the cart unloaded, it was possible to align the code with the behaviour of the system taken into exam. Thus, the variability has been greatly reduced. Then, a series of experimental tests confirmed that the adaptive code works effectively. In the event of a loss of step, the adaptive driving logic is able to change the trajectory of the cart. In this way, the latter can be returned safely for the next cycle.

The study presented showed that it is interesting to investigate further the driving of stepper motors via open-loop control strategy. In fact, by adopting it, it is possible to have a considerable saving on the control instruments. The total mechatronic system is less expensive. For any future developments, it is advisable to try to reduce the variability of time measurements performed by software as much as possible.

Bibliography

Publications and Manuals

[1] P. Acarnley. *Stepping Motors: A Guide to Modern Theory and Practice, 4th ed.* London, U.K.: The Institution of Engineering and Tecnology, 2002 (cit. on pp. 8, 18).

[2] A. Bellini et al. “Mixed Mode PWM for High-Performance Stepping Motors”. In: *IEEE Transactions on Industrial Electronics* 54.6 (Dec. 2007) (cit. on pp. 9, 25, 26).

[3] M. Bendjedia et al. “Position Control of a Sensorless Stepper Motor”. In: *IEEE Transactions on Power Electronics* 27.2 (Feb. 2012) (cit. on pp. 8, 21).

[4] N. Bianchi et al. “Comparison of PM motor structures and sensorless control techniques for zero-speed rotor position detection”. In: *IEEE.* June 2006 (cit. on p. 8).

[5] M. Boussak. “Implementation and Experimental Investigation of Sensorless Speed Control With Initial Rotor Position Estimation for Interior Permanent Magnet Synchronous Motor Drive”. In: *IEEE Ttransactions on Power Electronics* 20.6 (Nov. 2005).

[6] S. Derammelaere, Florian Verbelen, and K. Stockman. “Open loop control of a stepping motor with step loss detection and stall detection using back-EMF based load angle estimation”. In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics* (July 2014) (cit. on p. 9).

[7] S. Derammelaere et al. “Sensorless load angle control for two-phase hybrid stepper motors”. In: *Elsevier Ltd.* 43 (Feb. 2017), pp. 6–17 (cit. on p. 9).

[8] D.R. Gaan, M. Kumar, and S. Sudhakar. “Real-Time Precise Position Tracking With Stepper Motor Using Frequency Modulation Based Microstepping”. In: *IEEE Transactions on Industry Apllications* 54.1 (Jan. 2018) (cit. on p. 26).

[9] H. Le-Huy, P. Brunelle, and G. Sybille. “Design and Implementation of a Versatile Stepper Motor Model for Simulink’s SimPowerSystems”. In: *IEEE* (2008).

[10] Texas Instruments. *DRV8825 Stepper Motor Controller IC*. Texas Instruments. Texas Instruments, Post Office Box 6553053, Dallas, Texas 75265, July 2014 (cit. on p. 13).

[11] T. Kenjo. *Stepping motors and their microprocessor control*. New York: Oxford University Press, 1984 (cit. on pp. 8, 18, 21, 25).

[12] M.Y. Stoychitch. “An Algorithm of Linear Speed Control of a Stepper Motor in Real Time”. In: *Annals of Faculty Engineering Hunedoara - International Journal of Engineering* 9.3 (2013) (cit. on p. 34).

[13] N.M. Tomy and J. Francis. “Modelling and Simulation of a Hybrid Stepper Motor in Microstepping Mode”. In: *International journal of Advanced Technology in Engineering and Science* 3.9 (Sept. 2015).

[14] M. S. Zaky. “A self-tuning PI controller for the speed control of electrical motordrives”. In: *Electric Power Systems Research* 119 (Nov. 2014), pp. 293–303 (cit. on p. 7).

Online Resources

[15] D. Austin. *Generate stepper-motor speed profiles in real time*. Dec. 30, 2004. URL: `https://www.embedded.com/design/mcus-processors-and-socs/4006438/Generate-stepper-motor-speed-profiles-in-real-time` (cit. on p. 34).

[16] MICROMO. *Stepper Motor Technical Note: Microstepping Myths and Realities*. 2019. URL: `https://www.micromo.com/technical-library/stepper-motor-tutorials/microstepping-myths-and-realities` (cit. on p. 9).

[17] pigpio. *pigpio library*. URL: `http://abyz.me.uk/rpi/pigpio/index.html` (cit. on pp. 8, 36).

[18] R. Ramsdale. *Engineer’s Handbook: Coefficient of Friction*. 2019. URL: `http://www.engineershandbook.com/Tables/frictioncoefficients.htm` (cit. on p. 12).

[19] Rototron. *Raspberry Pi Stepper Motor Tutorial*. 2017. URL: `https://www.rototron.info/raspberry-pi-stepper-motor-tutorial/` (cit. on pp. 8, 13).