Universidade de Aveiro
2019

Departamento de
Eletrónica, Telecomunicações e Informática

André Miguel
Lopes Coquim

Development of a BLDC controller for integration into a robotic manipulator
Desenvolvimento de um controlador BLDC para integração num manipulador robótico

Universidade de Aveiro Departamento de
2019 Eletrónica, Telecomunicações e Informática

André Miguel
Lopes Coquim

Development of a BLDC controller for integration into a robotic manipulator
Desenvolvimento de um controlador BLDC para integração num manipulador robótico

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Eletrónica e Telecomunicações, realizada sob a orientação científica do Professor Doutor José Luís Azevedo e do Professor Doutor Bernardo Cunha e colaboração do Professor Doutor Artur Pereira, Professores auxiliares do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president	**Professor Doutor Pedro Nicolau Faria da Fonseca** Professor Auxiliar da Universidade de Aveiro (por delegação da reitoria da Universidade de Aveiro)
vogais / examiners committee	**Professor Doutor António Paulo Gomes Mendes Moreira** Professor Associado da Faculdade de Engenharia da Universidade do Porto (arguente principal)
	Professor Doutor José Luis Costa Pinto de Azevedo Professor Auxiliar da Universidade de Aveiro (orientador)

agradecimentos / acknowledgements

Aproveito esta oportunidade para agradecer aos meus orientadores Professor Doutor José Luís Azevedo e Professor Doutor Bernardo Cunha e ao colaborador Professor Doutor Artur Pereira pela oportunidade de realizar esta dissertação, por todos os conhecimentos que me passaram e por toda a disponibilidade demonstrada. Quero agradecer especialmente ao Professor Doutor José Luís Azevedo pois foi desde o início quem me apoiou mais no desenvolvimento de todo este trabalho e esteve sempre presente nas fases mais críticas do mesmo. Mostrou sempre uma total disponibilidade em me ajudar no que fosse necessário e ensinou-me muitas coisas novas. Também agradeço por todo o tempo perdido a rever este documento, para que este apresentasse a melhor qualidade possível. Quero agradecer também aos meus colegas Filipe Renca e Francisco Pinto pois também me ajudaram a superar dificuldades em fases deste trabalho.

Por fim quero agradecer a todas as pessoas que ao longo dos anos que passei neste curso fizeram com que este percurso fosse mais fácil e me ajudaram a superar as várias dificuldades que foram aparecendo. A todos, o meu obrigado.

Palavras-Chave Motor, BLDC, PID, Perfil de velocidade, CAN

Resumo

Esta dissertação descreve o trabalho realizado na criação e desenvolvimento de um controlador do ombro de um braço robótico antropomórfico no âmbito do projeto CAMBADA@Home. No início deste projeto é feito um estudo sobre os vários tipos de motores elétricos existentes, bem como formas de obter *feedback* relativo a posição e velocidade por parte destes, por exemplo com recurso encoders. É posteriormente feito o projeto de uma arquitetura adequada para o controlador bem como uma seleção de componentes de modo a que fosse possível implementá-la. Isto deu origem a um protótipo que é posteriormente demonstarado. São depois apresentadas informações e resultados relativos ao desenvolvimento do mesmo, nomeadamente, sinais elétricos relevantes, a resposta do sistema quando colocado, sem carga, em malha aberta e em malha fechada sob efeito de um controlador PID, a forma como este implementa um perfil de velocidades e resultados relativos à comunicação via CAN. Por fim, é feita uma análise cuidada e são tiradas conclusões sobre o trabalho desenvolvido e aquele que ainda pode vir a ser feito futuramente.

Keywords Motor, BLDC, PID, Speed Profile, CAN

Abstract This thesis describes the work done in the creation and development of an anthropomorphic robotic arm shoulder driver under the CAMBADA@Home project. At the beginning of this project, a study is done on the various types of electric motors available, as well as ways to get feedback on the position and speed of some of these components, for example using encoders. Later, a suitable architecture for the robotic arm controller is made, as well as a selection of components so that it can be implemented with it. This gave rise to a prototype that was later demonstrated. Information and results related to its development are presented, namely, relevant electrical signals, the system response when unloaded, for both open loop and closed loop under the effect of a PID controller, a speed profile implementation and results related to CAN communication. Finally, an analysis is made and conclusions are taken about the developed work and what is left to be done in the future.

Contents

List of Figures

List of Tables

Acronyms

AC Alternating Current

BLDC Brushless DC

CAMBADA Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture

CAN Controller Area Network

CCW Counterclockwise

CMOS Complementary Metal-Oxide-Semiconductor

CW Clockwise

DC Direct Current

ECM Electronically Commutated Motor

EMF ElectroMotive Force

MOSFET Metal Oxide Semiconductor Field Effect Transistor

PC Personal Computer

PCB Printed Circuit Board

PD Proportional Derivative

PI Proportional Integral

PID Proportional Integral Derivative

PPR Pulse Per Revolution

PWM Pulse Width Modulation

RPM Revolutions per minute

UART Universal Asynchronous Receiver-Transmitter

USB Universal Serial Bus

Chapter 1

Introduction

1.1 Motivation and objectives

The modern word robot descended from the Czech word robota used in Karel Čapek's play R.U.R. in 1920. A robot can be seen as any automatically operated machine that replaces human effort. However, it does not perform functions in a human like manner. The first stationary industrial robot was created in 1954 and was an electronically controlled hydraulic heavy-lifting that repeated arbitrary sequences of motions. Later robots using artificial intelligence have been built[1].

Nowadays, artificial intelligence has already multiple applications in diverse activities such as healthcare, smart home devices, robot control, travel and navigation, etc.[2].

CAMBADA@Home is a multidisciplinary project in intelligent robotics that aims to develop a self-service robot capable of helping in domestic situations and/or helping people with special needs. The accomplishment of such tasks requires equiping the robot with manipulation capacity, through a robotic arm.

As a result of a previous work[3], a part of the robotic arm (6 of the 7 degrees of freedom) has been developed through the use of Dynamixel[1] servos. To control the first joint of the robot's arm, it is foreseen the use of a BLDC motor. The controller must communicate with the other modules of the system via a CAN network. This work objectives are then:

- Design and implementation of the BLDC motor controller responsible for the shoulder movement of the anthropomorphic robotic arm.
- Development of software to control the arm speed and position using speed profiles.
- Test and characterization of the developed motor controller for different load situations.
- Integration of the developd controller in the CAMBADA@Home CAN network in order to receive orders and operating parameters from the central computer.

1.2 Document structure

This document is divided in various chapters, where each chapter aims to show different development phases of the developed robotic arm controller.

[1]http://www.hizook.com/files/users/3/RX-28_Robotis_Dynamixel_Servo_UserGuide.pdf

- **Chapter 2** - This chapter provides an overview of the main characteristics and working principles of brushed and brushless DC motors, as well as a further analysis on BLDC motors control techniques, in order to maximize their performance and energy efficiency. Encoders are a fundamental piece to solve the proposed problem and so they are also studied in this chapter. Finally, a brief description of the CAN protocol is performed.
- **Chapter 3** - This chapter focus on the perception and understanding of the problem, the search of available resources as well as the definition of an architecture. It focus on a block diagram of the robotic arm controller's architecture and also on position and speed control perspective diagram to deeply explain the speed profile concept.
- **Chapter 4** - This chapter presents the entire process of developing a prototype of the robotic arm controller. Component choices are justified along the chapter. All preliminary tests performed are described. In the end, a fluxogram of the software algorithm implemented by the controller is also presented.
- **Chapter 5** - This chapter presents all the results obtained along the project and how the final system works.
- **Chapter 6** - This chapter focus on the conclusions that can be taken from the developed work, what's left to be done in the future and what has already been done that can be used as a starting point to it.

Chapter 2

Theoretical background

This chapter starts with a brief description of the existent types of electric motors and thet applications they are most suited for. Later in this chapter an in-depth study on BLDC motors is made explaining the theory behind their operation and methods to control them. A detailed comparison of these type of motors with the brushed dc motors is also made, pointing out the advantages and disadvantages of each other. Consequently, it is justified why BLDCs are better suited to carrying out the proposed work.

2.1 Electric motors

Eletric motors are devices capable of converting electric energy into mechanical energy. This type of motors can be divided into two categories depending on their power source. The ones that are powered by a continuous current are designated DC motors, on the other hand the ones that are powered by an alternated current are designated AC motors. There also exist solutions that mix these two types. In the DC motors there are two categories that will have more attention in this work, which are, brushed DC motors and brushless DC motors. However, before distinguish these two types of motors it is important to distinguish the AC motors from the DC motors in terms of their general characteristics.

2.1.1 AC motors

AC motors are characterized for having lower power demand on start, controlled accelaration, adjustable operational speed, controlled starting current, adjustable torque limit and reduced power line disturbances. However, AC motors often lose torque as speed increases which may be a problem if a specific operation requires high torque at high speeds.

These motors have a much larger installed base than the DC motors[4][5].

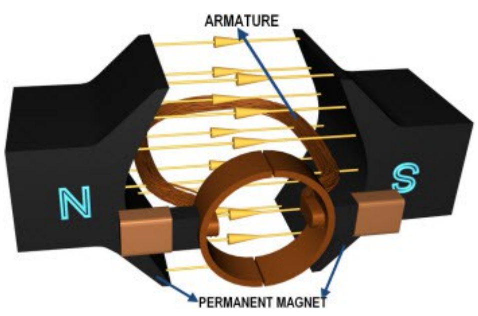
2.1.2 DC motors

For low power units these motors are usually less expensive than AC motors. DC motors speed can be controlled by varying the supply voltage and are available in a wide range of voltages. They also have easy installation, speed control over a wide range, quick starting, stopping, reversing acceleration, high starting torque and a linear speed-torque curve. DC motors can be powered by batteries or DC voltage sources.[4][5]. In mobile robotic applications it is desirable that robots are powered through batteries, as this allows greater mobility.

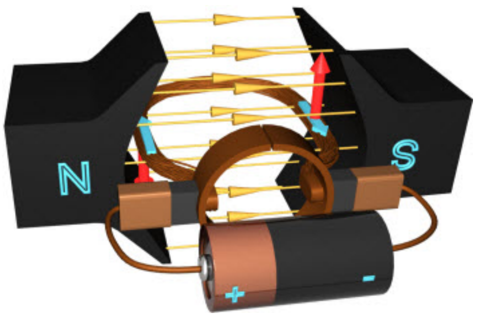
2.2 Brushed DC motors

First of all, it is important to understand what is a brushed DC motor and how it works[6][7]. Figure 2.1a represents a basic diagram. The brushed DC motor is formed by a rotating part designated armature, that is nothing more than a coil, and two permanent magnets, which are usually designated stator. The stator provides a constant magnetic field. In its turn the armature is connected to a DC power source through a pair of rings as ilustrated in figure 2.1b. When the current flows through the coil a magnetic force is induced on it according to the Lorentz Law so the coil will start to rotate. As the coil rotates there will be a time when the commutator rings (also called colector) represented in blue and bright blue in figure 2.1c will connect with the power source of the opposite polarity. As a result, on the left side of the coil, the current will always flow away from the commutated ring connected to the positive pole. On the other hand, on the right side, current always flow towards the commutated ring connected to the negative pole, so the coil will continue rotating in the same direction due to the torque action.

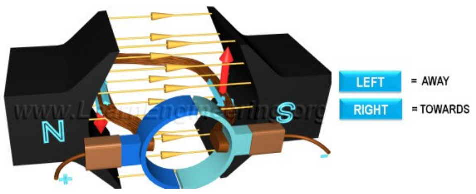
This is the basic idea of how a brushed DC motor works, however, this would lead to irregular motion of the motor. The irregular motion is due to the coil being perpendicular to the magnetic field, at some instants and so the torque that acts in it will be nearly zero as shown in figure 2.1d. The solution to this problem is to add another coil as shown in figure 2.1e. In this case when the first loop is in the vertical position, the second loop will be connected to the power source, so a motive force is always present in the system. In practical cases more coils are used as it is shown in figure 2.1f, so the motor rotation becomes smoother. Also, the armature loops are fitted inside slots of highly permeable steel layers that enhance magnetic flux interaction. Spring loaded commutator brushes help to maintain contact with the power source as shown in figure 2.1g.



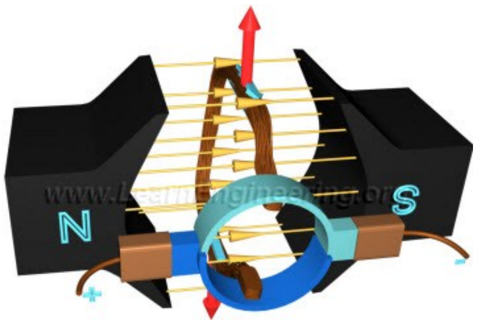
(a) Basic diagram of a DC motor.



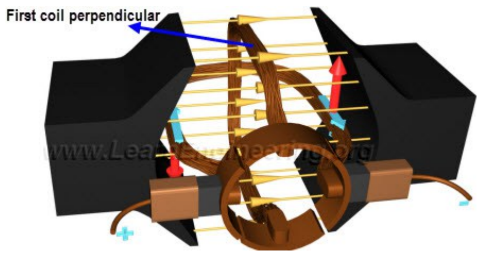
(b) The eletromagnetic field force induced on the coil makes it rotate.



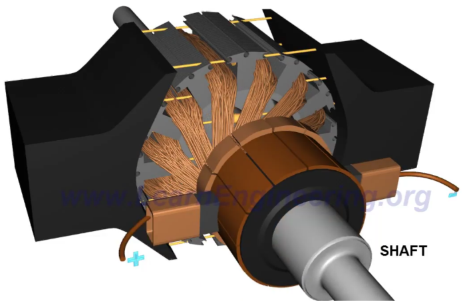
(c) The commutator rings make sure a uni-directional current flows through the left and right part of the coil.



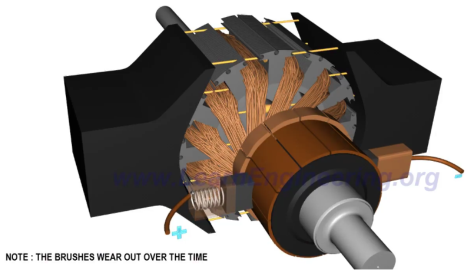
(d) When the coil nears perpendicular to the magnetic flux, the torque produced nears zero.



(e) Two coil armature arrangement.



(f) More coils generate a smoother rotation; to enhance magnetic flux interaction, coils are put between steel layer poles.



(g) Brushes help to maintain contact with the power source.

Figure 2.1: Brushed DC motor operation and its parts (From [6]).

The brushed DC motor is called this way due to its need of having brushes to transfer the eletric energy to the colector. The brushes can be made of graphite/carbon or precious metals. The type of material used is decided considering the type, quality and costs pretended. In figure 2.2 it is possible to understand the difference between using graphite brushes versus using precious metals brushes. As it can be seen in the graphite brushes case there is a higher high frequency noise in control circuits and so there also exists a higher eletromagnetic interference.

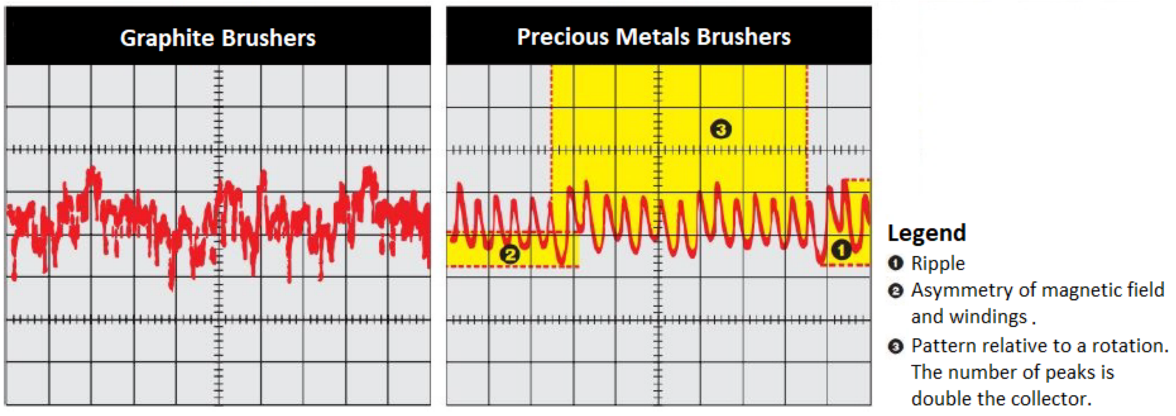


Figure 2.2: Switching current standard for two types of brushes (From [8]).

2.2.1 Common Problems of the brushed DC Motors

Brushed DC motors are not reliable for long life operations essentially because of their brushes. Over the time the brushes tend to wear out and may also make electric sparks. This can cause malfunctions in systems that operate for long periods. Therefore, they need periodic maintenance. As it will be referred in the next section there are aspects about this type of motors that can be improved and make them less widely used nowadays. The recent trend has been to use brushless DC motors because they seem to be more efficient, more reliable and less noisy than the traditional Brushed DC motors.

2.3 Brushless DC motors

As the name implies, brushless DC motors are motors that do not need brushes to operate. That already represents a big improvement when compared to the traditional brushed DC motors, because the problems related with the existence of brushes, referred in the last section, vanish. From now on, these type of motors will be referred in this document as they are usually known, that is as BLDC motors.

BLDC motors do not need brushes because they have their permanent magnets in the rotor, so they do not need a collector. The commutation is done electronically, commanding the correct sequence and direction of the stator windings. Due to this particularity, this kind of motors are also called Electronically Commutated Motors (ECM)[6][7].

These motors can be divided into two types depending on their construction. They can be inrunners if the stator windings surround the permanet magnets, or outrunners if the permanent magnets surround the stator as shown in figure 2.3. The outrunner design has a clear mechanical advantage over the inrunner design. At higher speeds the runner tends to expand slightly due to the centrifugal force. As a result, in inrunner designs a good amount of clearance should be given between the rotor and stator to avoid the collision. Such higher clearances increase the magnetic flux leakages and reduce efficiency of the motor, the outrunner design has no such limitation, as the runner at the outside is free to expand.

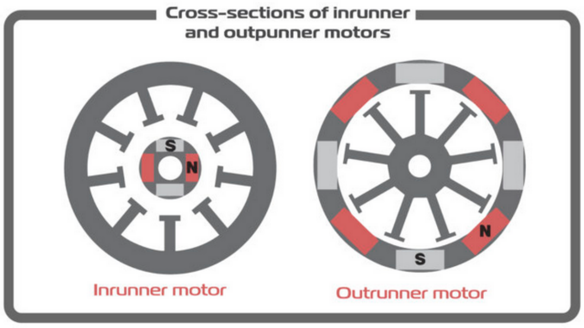
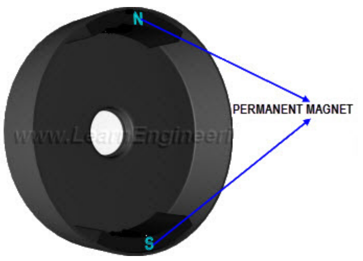


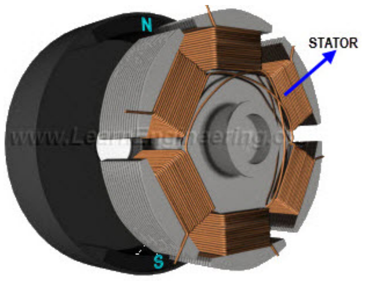
Figure 2.3: In inrunner motors the stator surround the permanent magnets while in outrunner motors the permanent magnets surround the stator (From [9]).

It is now important to understand how a BLDC motor works[10][7] and for that, the outrunner case will be presented. The rotor of a BLDC motor is a permanent magnet and the stator has a coil arrangement as shown in both figure 2.4a and figure 2.4b. In this example, the rotor has 3 coils A, B and C and the internal winding is depicted in figure 2.4c. When energized individually by a DC power source, the coils create a magnetic field as it is shown in figure 2.4d and become an electromagnet. The operation of a BLDC is based on the force interaction between both the permanent magnet and the generated electromagnet. When coil A is energized the opposite poles of the rotor and the stator are attracted to each other. This makes the rotor poles move closer to the stator poles due to the attractive force. As the rotor approaches coil A, coil B is energized, which will cause the rotor to approach coil B and coil C becomes energized. After that, coil A is energized but now with opposite polarity. The process is repeated and the rotor continues to rotate. This phenomenon is illustrated in figure 2.4e, where the green arrows represent the attractive force. One can think of six time intervals before coil A turn into its original position. This intervals and the DC voltage required in each coil are represented in figure 2.5a.

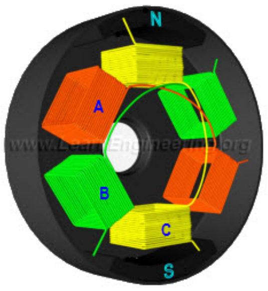
This is the basic explanation of how a BLDC motor works, however, this process has a drawback, which is, at any instant only one coil is energized and the other two coils are dead. This greatly reduces the power of the motor. However, there is a trick to overcome this problem. Looking back at the first position of figure 2.4e, if the coil behind the rotor is energized in such a way that it will push the rotor as described in figure 2.4f, with red arrows, this will make so that in one instant a current with the same polarity passes through the second coil. The combined effect produces more torque and power output from the motor. This effect also makes sure that a BLDC has a constant torque nature as it is presented in figure 2.6. This torque nature is difficult to achieve in other types of motors. The voltage scheme from figure 2.5a becomes now as presented in figure 2.5b. In addition, by connecting the free ends of the coils together, it is possible to obtain a current flow exactly as in the separately energized state, with the advantage that now it is not necessary to energize the two coils individually as before. The current flow of this final configuration is shown in figure 2.4g.



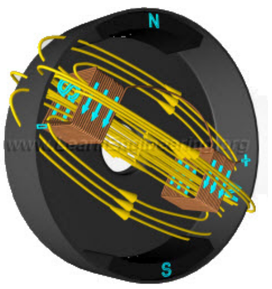
(a) The rotor of a BLDC is a permanent magnet.



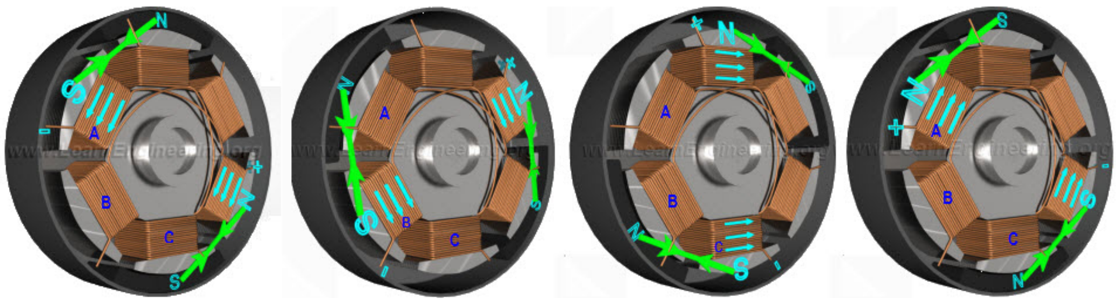
(b) The stator has a winding arrangement.



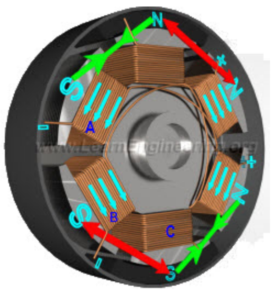
(c) The coil arrangement in a BLDC is shown here, with different color for different coils.



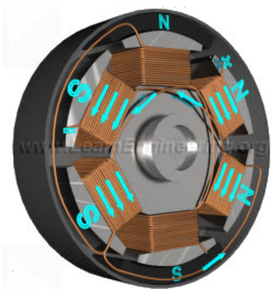
(d) The coil energized by a DC power source becomes an electromagnet.



(e) The rotor moves toward the energized coil, due to the attractive force and the next coil is energized.

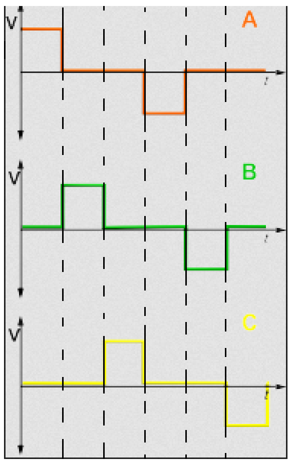


(f) One more coil is energized in practical motors.

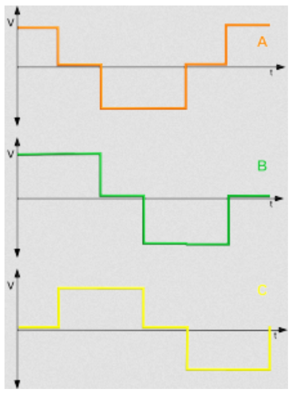


(g) Connected winding produces exactly same current flow as that of the separately energized state.

Figure 2.4: BLDC motor operation (From [10]).



(a) Voltage form required in each coil when a single coil is energized.



(b) Voltage form required in each coil when two coils are energized.

Figure 2.5: BLDC voltage forms (From [10]).

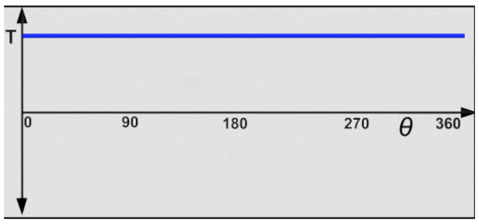


Figure 2.6: The BLDC motor has a constant torque nature (From [10]).

2.3.1 Energizing a BLDC motor

To energize a BLDC motor external electronic controllers are used. These controllers switch the current in the motor phases in order to generate motion. This process is called *Commutation.* Usually a sensor determines the position of the rotor and based on this information the controller decides which coils to energize and when to energize them. Figure 2.7 represents a simple diagram of this idea. The sensors used for this purpose are the Hall-effect sensors which generate a specific voltage value depending on the magnetic field that crosses them[11][7].

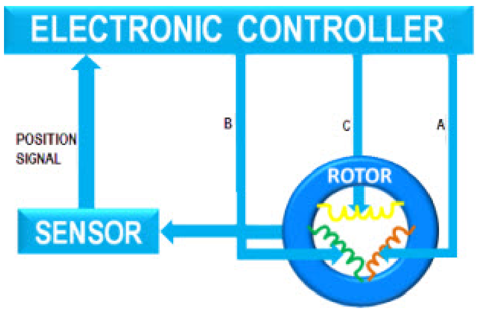


Figure 2.7: The controller determines which coil to energize and when to energize it (From [10]).

2.3.2 Commutation and position feedback

For the controller to maintain accurate control of the BLDC motor, it must always know the exact position of the rotor in relation to the stator. Any misalignment or phase shift in the expected and actual position may result in undesirable behavior and a decline in performance. Although there are sensorless switching techniques, the most common way to get this feedback is to use Hall-effect sensors, encoders, or resolvers. The Hall-effect sensors are embedded into the stator of the motor to detect rotor positon, which is used to switch the transistors in a 3-phase bridge to drive the motor.

A schematic of the 3-phase bridge circuit is presented in figure 2.8, where it can be seen 6 power MOSFETs forming three pairs, where each pair is connected to a different motor winding. One pair of MOSFETs consists of one low side MOSFET and one high side MOSFET, for example SW1 and SW2. A PWM signal is injected sequentially in the gates of all transistors, one low side and one high side transitor at a time, of differents pairs.

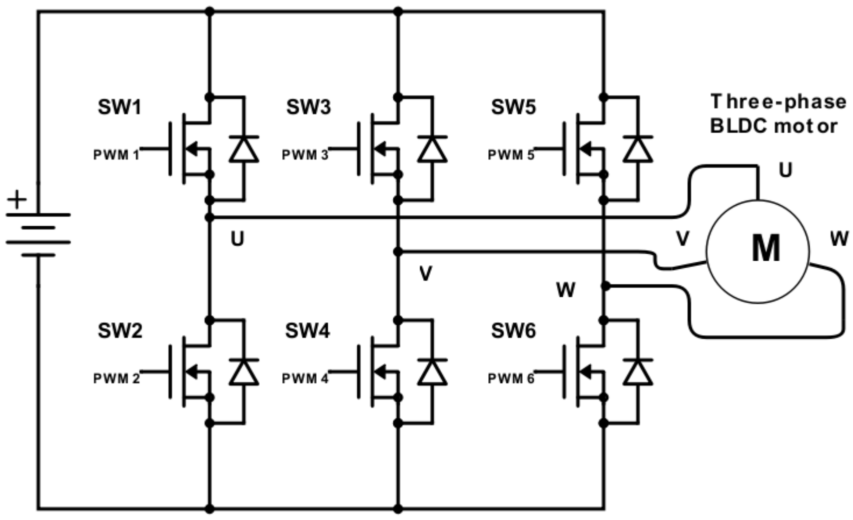


Figure 2.8: A three-phase BLDC motor is typically powered by three pairs of MOSFETs arranged in a bridge structure and controlled by PWM (From [12]).

While Hall-effect sensors are an effective solution for commutating BLDC motors, they only address half the needs of a BLDC system. Hall-effect sensors will allow a controller to drive a BLDC motor, but their control is limited to speed and direction. With a 3-phase motor, Hall-effect sensors can only provide angular position within each electrical cycle. To ensure a robust and complete solution, the BLDC system should provide real-time position information, so that the controller can track not only speed and direction, but distance travelled and angular position as well. To answer the need of tighter position information an absolute encoder is added to the BLDC motor along with the Hall-effect sensors. The Hall-effect sensors are used for motor commutation, and the encoder is used to track position, rotation, speed and direction with far greater accuracy. Since Hall-effect sensors only provide new position information at each Hall state change, their precision is limited to six states per electrical revolution. For a two-pole motor, this results in only six states per mechanical revolution, while absolute encoders offer resolutions in the thousands of PPR (pulses per revolution). Hence, it becomes obvious why it is useful to have both. However, for the motor to be efficiently commutated, the rotor and stator positions must be known. This means great care must be taken to ensure that the commutation encoder's U/V/W channels are properly aligned to the BLDC motor's phases.

2.4 Comparison between brushed and brushless DC motors

Now that is clear how both BLDC and brushed DC motors work it is important to make a full comparison between them in terms of characteristics, advantages, disadvantages and take conclusions about them. For that, many topics shall be analyzed [8][13].

2.4.1 Efficiency and losses

In electronic terms efficiency is a measure of the relation that exists between the total power input of a system and the useful power output of it. It can be expressed by:

$$Efficiency = \frac{UsefulPowerOutput}{TotalPowerInput} \tag{2.1}$$

Efficiency can never be more than 100%, being always less than this value in real systems. In real systems there are energy losses and that's the reason why efficiency is never 100%. The total power input can then be interpreted as the sum of the useful power output plus the losses. The useful power output in a motor can be interpreted as the electric energy that is transformed into mechanical energy.

Knowing this, it is now possible to make a comparison in terms of efficiency between brushed and brushless DC motors and conclude something about it. Various types of losses can be considered in motors, namely, electric losses, mechanical losses and magnetic losses. Next, some of them are sumarized:

- Electric losses
  - Copper losses in the inductor and in the inducted, resulting from the copper conductors resistance not being zero. Inducted copper losses are inexistent in BLDC motors because their rotor has permanent magnets;

  - Losses due to the voltage drop in the brush-collector contact. This doesn't happen in BLDC motors;
  - Losses due to the short circuit currents in the commutation.

- Mechanical losses
  - Friction losses in the collector and brushes. This does not happen in BLDC motors;
  - Losses due to aerodynamic drag and ventilation. Some motors need to be ventilated so that they can dissipate the generated heat. Those motors usually have a fan included in the shaft's back part which causes aerodynamic drag. The bearings are also drag sources.

- Magnetic losses
  - Hysteresis losses at the ferromagnetic materials;
  - Losses due to Foucault's currents at the inductor and at the inducted.

To know the real efficiency of a motor all losses must be considered. The thermal losses and noise are consequences of the referred phenomenon. In the case of BLDC motors, the heat can cause the demagnetization of the permanent magnets so it is important to maintain them in good temperature conditions.

Considering all of the points presented before it is intuitive that losses are greater in brushed motors and so this leads to a better efficiency of the BLDC motors, when compared with the brushed ones.

2.4.2 Noise

In terms of noise, it is very low in BLDC motors when compared to the brushed DC motors. This is essencially due to the friction between brushes and colector existent in the brushed DC Motors as referred before.

2.4.3 Speed

BLDC motors can generate greater speed values than the brushed case. The inexistant brushes in the BLDC motors is the main reason to explain this, because there is no mechanical limit imposed by them as exists in the brushed motors.

2.4.4 Costs

There are many factors that shall be considered in terms of costs as the efficiency and the power of motors. Usually BLDC motors are more expensive than brushed DC motors due to the power magnets and also due to a more complex and consequently more expensive control.

2.4.5 Lifetime and maintenance

The BLDC motors have a longer operating life than the brushed DC motors due to the lack of electrical and friction losses. While the brushed DC motors need periodical maintenance due to the wear out of the brushes, the BLDC do not have this problem so they do not need such maintenance.

2.4.6 Control

As referred before the control in BLDC motors is more expensive and the existence of a controller is always needed either for fixed motor speeds or variable ones. On the other hand, brushed DC motors do not need a controller for fixed speeds, only for applications that use variable ones, so the control in these motors is usually cheaper and simpler.

2.5 BLDC Control techniques

BLDC motors are not self-commutating and so they need more complex control than the brushed DC motors. It is essential to know the rotor position so that the motor can be commutated. In closed-loop speed control, measurements of the motor speed and current are needed and also PWM signals are used for speed control. The PWM signals used by the motor can be of two types, which are center-aligned and edge-aligned PWM signals. If the application demands dynamic braking, dynamic reversal or servo-positioning, complementary center-aligned PWM signals should be used. However, for applications where only variable speed operation is needed, six independent edge-aligned PWM signals should be used, to provide the highest resolution. As referred before, Hall-effect sensors are used to provide absolute position sensing. However, for low-cost variable speed applications these sensors are not needed and electromotive force (back-EMF) of the motor is used to estimate the rotor position (some of these applications are, for example, fans and pumps). Commutation that uses Hall-effect sensors is commonly designated *Sensored Commutation* while in the opposite case is designated *Sensorless Commutation.* There are two main commutation and control techniques that apply to the BLDC motors, which are *Trapezoidal commutation* and *Sinusoidal commutation* [8][14][15][16].

2.5.1 Trapezoidal commutation

Considering a three phase motor and the sensored commutation case, three Hall-effect sensors are disposed with a gap of 120° from one another and provide digital signals that track rotor position within 60° sectors. This data is then sent to the motor controller. When the motor turns, the current at the motor terminals is electronically commutated for each 60° of rotation. This way the current space vector can allways be within the nearest 30° of the quadrature direction. The current waveform for each winding assumes then the form of a staircase from zero, to positive current, to zero, and then to negative current as presented in figure 2.10. This current form generates a current space vector that approximates smooth rotation, as the rotor turns. A simplified block diagram of the trapezoidal controller is depicted in figure 2.9 where it can be seen that the output of the PI controler is the input of the PWM modulator that will act in the motor through the bridge. Pulse width modulation is used to apply a variable voltage to the motor windings being the effective voltage proportional to this signal's duty cycle.

For the sensorless commutation the rotor position is determined considering the zero crossings of the back-EMF inducted in each winding. The commutation is done 30° after detection, however due to the absence of the Hall-effect sensors, the time between commutation and detection becomes dependent of the motor rotation speed. The amplitude of the inducted back-EMF is also speed-dependent and so when the motor is stopped or at low speed the zero crossing detection is not very accurate. Consequently the estimated rotor position becomes little precise. To compensate for this, special boot algorithms may be required.

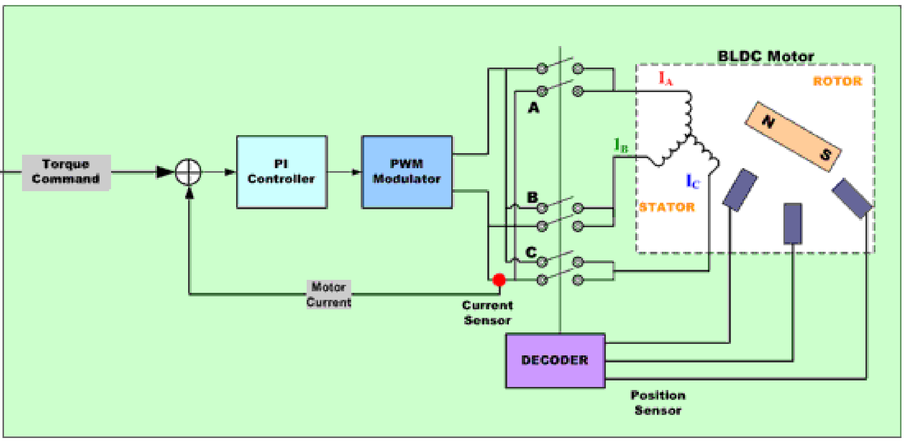


Figure 2.9: Simplified block diagram of trapezoidal controller for BLDC motor (From [14]).

Considering both the sensored and the sensorless cases it is easy to understand that the sensored case is much more accurate and more easily controllable at low/inexistant speeds. However, as referred before the sensored implementation is more expensive than sensorless implementation.

Theoretically in trapezoidal control the current presents a block like perfect form, however in practical cases the current waveform is not so perfect and presents what is usually referred as torque ripple. This phenomenon is obviously undesirable and it happens due to the incapacity of the current to be established instantaneously in a motor phase. Figure 2.10 represents the electrical waveforms (voltage and current) of each rotor phase as well as the generated torque for a practical case. The torque ripple phenomenon is evident in all waveforms.

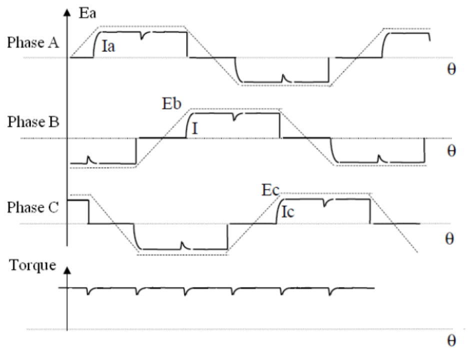


Figure 2.10: Electrical waveforms in the two phase ON operation and torque ripple (From [15]).

2.5.2 Sinusoidal commutation

For low speeds, trapezoidal commutation can not provide smooth and precise motor control of BLDC motors. Contrarily, sinusoidal commutation can solve this problem [14].

Sinusoidally commutated BLDC motor controllers attempt to drive the three motor windings with currents that change smoothly and sinusoidally as the motor rotates. To obtain a smoothly rotating current space vector the relative phases of this currents are chosen with care. This space vector has constant magnitude and should be always in the quadrature direction with respect to the rotor. This way any possible commutation spikes are then eliminated as well as the torque ripple, which were present in the trapezoidal commutation case.

An accurate measurement of rotor position is required so that smooth sinusoidal modulation of the motor currents is generated as the motor turns. This accuracy can only be found using angle feedback from an encoder or similar device as the Hall devices provide only an approximate measure of rotor position and so are not a viable option for this purpose. A block diagram of the sinusoidal controller can be seen in figure 2.11.

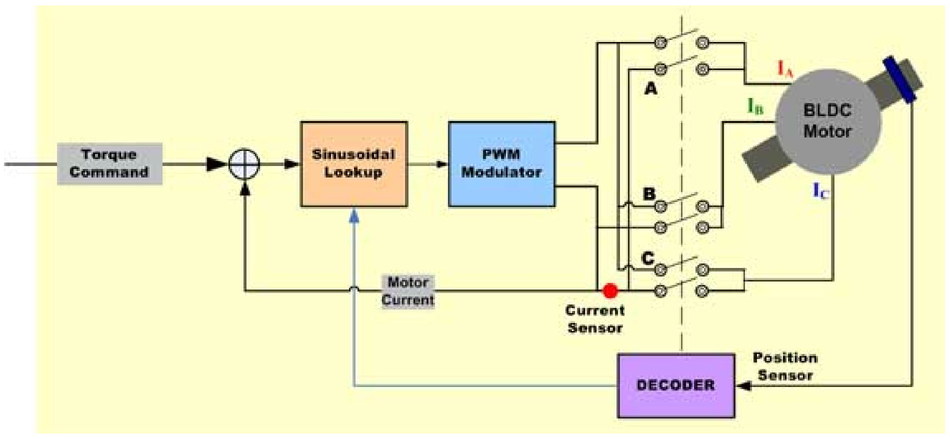


Figure 2.11: Simplified block diagram of sinusoidal controller for BLDC motor (From [14]).

Figure 2.12 shows how a sinusoid can be obtained through the PWM modulator.

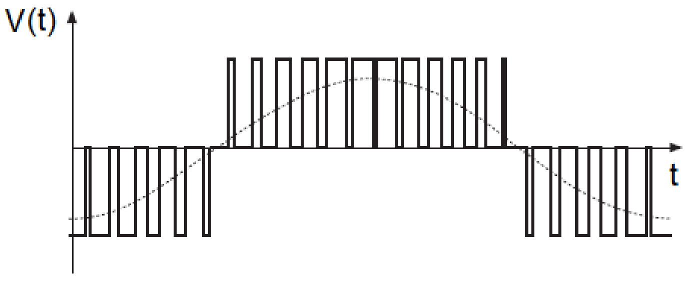


Figure 2.12: PWM modulation to obtain a sinuosidal signal (From [8]).

2.5.3 Comparison between trapezoidal commutation and sinusoidal commutation

Sinusoidal commutation provides smoothness of control that is unreachable in trapezoidal commutation. Sinusoidal control is effective at low motor speeds, however, it tends to not be so effective at high motor speeds. With the increase of speed the current loop controllers have to measure a sinusoidal signal of increasing frequency, also, they have to surpass the motor back-EMF which increases in amplitude and frequency as speed rises.

Higher speeds result in larger errors, due to the PI controllers limited gain and frequency response. The direction of the current space vector relative to the rotor is then disturbed, causing it to not go according to the quadrature direction. This leads to less torque being generated by a given amount of current. To maintain the same torque level more current is needed which leads to efficiency deterioration.

The cost is also an important factor. Trapezoidal commutation is less expensive due to the lack of an encoder. Even considering the sinusoidal commutation without an encoder there would be the need to use more complex algorithms, so the trapezoidal case is the winner in terms of simplicity. Figure 2.13 shows the difference between the control phase currents in both the trapezoidal and sinusoidal cases.

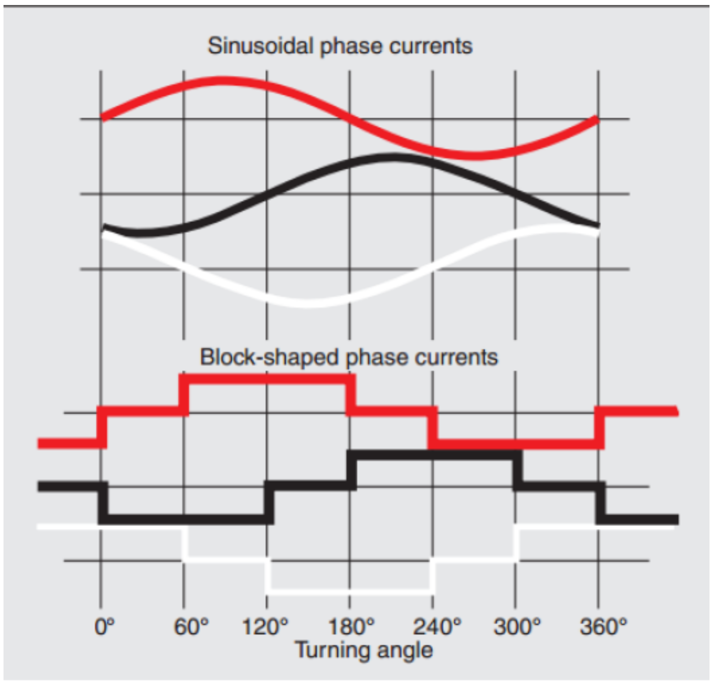


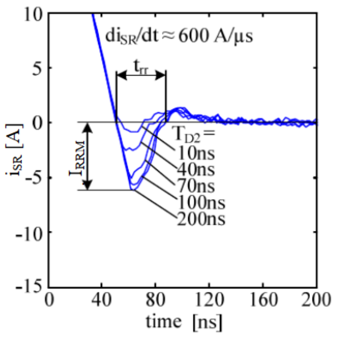
Figure 2.13: Sinusoidal and trapezoidal control currents (From [8]).

2.6 Drive circuit

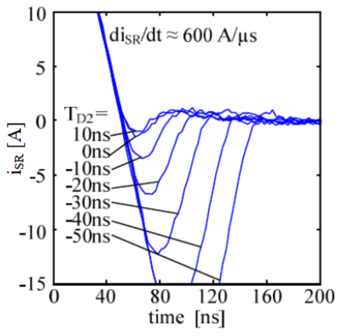
Usually controllers contain the logic needed to control motors, but do not handle the power to drive them. Their output voltages are not high enough to drive the MOSFETs gates and they also do not provide enough current to charge and discharge them fast enough. To address these problems a drive circuit is put between the BLDC controller and the bridge. The drive circuit is constructed based on the bridge structure. N-channel MOSFETS have a lower $R_{DS(on)}$ value so they are desirable for their lower losses and faster commutation times. Low side MOSFETs in the bridge are N-channel type because the voltage level needed for them to be cut or conducting is usually easily achieved with the available power supply. However, for the high side N-channel MOSFETs their drain needs to be at a higher potential than their source (otherwise their body-diode would open). This means that their source potential can be between the supply voltage and ground, depending if they are conducting or not. In order to turn one MOSFET on, its gate should be at higher potential than its source. This implies that the high side drive circuit has to be capable of generating a higher voltage than the available in the power supply. On the other hand, with high side P-channel MOSFETs the gate voltage value needed to turn them on and off is always below the one of the power supply.

The gate capacitance and the available drive current from the drive circuit determine how fast the transistor can be turned on or off. To make the time shorter one can either change the MOSFET to one with a lower gate capacitance or change the driver to one that can provide more current. On the other hand, to make this time bigger one can add a series resistor between the driver circuit and each MOSFET gate.

To prevent cross conduction (or shoot through), which is a phenomenon where two MOSFETs of the same pair get shorted due to the delay between one MOSFET turn off and the other turn on, which could lead both to be turned on at the same time, a suited dead time value must be chosen. The dead time can be defined as ”the time delay between the falling edge of the control signal of one MOSFET and the rising edge of the control signal of the opposite MOSFET”. This time should not be greater than its ideal value as the reverse recovery time (t_{RRM}) and reverse recovery current (I_{RRM}) rise and that might damage the MOSFETs. However, if this time is less than its ideal value, cross conduction occurs and the generated current spikes are much bigger than in the previous case for the same time deviation (in opposite direction). Figure 2.14 shows the results obtained for a 75 V-MOSFET (IRF2907, packaged in D2Pak) with the turn-off dead time as parameter (Measured at: VLV = 14 V, VHV = 40 V, iL(t3) = 30 A, TPCB = 30°C) [17].



(a) Reverse recovery (positive dead time variation to its ideal value).



(b) Shoot through (negative dead time variation to its ideal value).

Figure 2.14: Peak current comparison for the reverse recovery and shoot-through cases (From [17]).

Considering a half bridge composed only of N-channel MOSFETs, for the low side MOSFETS when the gate-driver voltage of the transistor is higher than the available digital supply, at least a level-shifter with a complementary drive stage is needed to drive the device [18]. One schematic to illustrate the needed circuit is presented in figure 2.15.

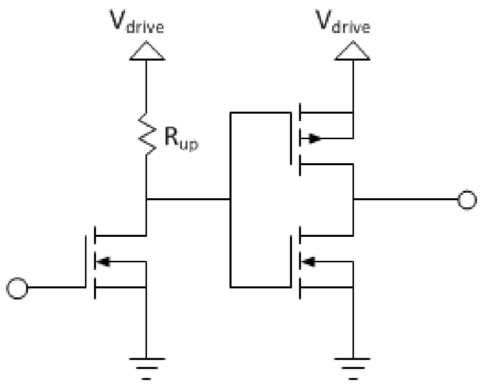


Figure 2.15: Low side drive example circuit (From [18]).

As referred before for high side N-channel MOSFETs to operate, there is the need to generate a voltage value higher than the one available in the power supply. For that, usually a charge pump in a boot-strapped configuration is used to generate that voltage level. The needed circuit is something like the schematic presented in figure 2.16. Still, this implementation has some drawbacks [18]:

- bridges driven by this type of driver can not operate at 100% duty cycle, there is the need to give some time in every cycle for C_{boot} to recharge.
- There is the need for C_{boot} to be at least an order of magnitude higher than the gate capacitance of Q1 so that the value of V_{gate} does not decrease along in some phases of the process.
- The charge pump process generates current spikes that might be larger than the average current flowing to the gate of Q1. This high current can stress the power delivery network and may be a strong noise source for the rest the design.

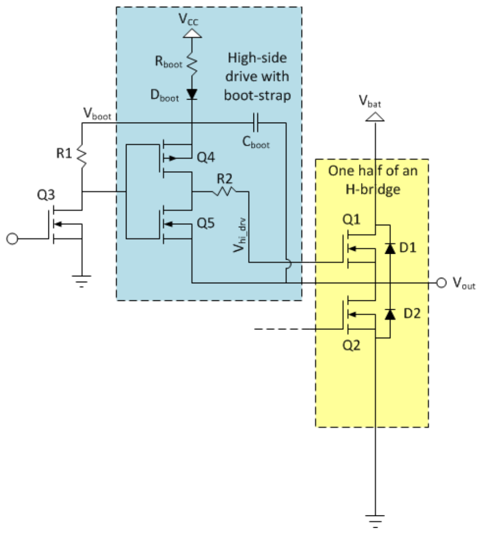


Figure 2.16: High side drive example circuit (From [18]).

2.7 Encoders

As mentioned in a previous section an encoder is needed to track position, rotation speed, and direction with far greater accuracy than what Hall-effect sensors can do. For that it is important to know the different types of encoders available in the market, the characteristics of each type, their advantages/disadvantages and how they work.

The encoders used with motors are known as *rotary encoders*. Rotary encoders can be classified into two types based on their output signal, which are *incremental encoders* and *absolute encoders*.

An incremental encoder is a simple encoder that is able to immediately report changes in position, which is an essential capability in some applications. However, it does not report or keep track of absolute position, it can only indicate relative position to a point. As a result, the mechanical system monitored by an incremental encoder may have to be moved to a fixed reference point to initialize the position measurement. [19][20].

An absolute encoder keeps track of absolute position and maintains position information when power is removed, which is not the case of an incremental encoder that needs additional logic to restore position when power is restablished. However, due to a more complex construction absolute encoders are more expensive than the incremental ones[19][20].

2.8 Feedback control

To control the robot arm, it will be needed a microcontroller that implements a control algorithm via software so that the speed that the robot arm moves and the final position it reaches are accurate. For that there are several control loop feedback alghoritms. However, there is one that is widely used in industrial control systems, which is the Proportional Integral Derivative (PID) controller [21].

2.8.1 Continuous time PID controller

A feedback control loop containing a continuous time PID is represented in figure 2.17. The controller is composed of 5 components, a subtractor block (responsible for obtaining the error signal), an integrator block (responsible for calculating the error signal integral), differentiating block (responsible for calculating the error derivative), a sum block and a gain block. The signal r(t) represents the setpoint that the system is expected to reach (for example a speed value); y(t) represents the actual value that the real output of the system reaches; e(t)=r(t)-y(t) is the error signal representing the difference between the expected output of the system and the real output. Finally u(t) is the output signal of the controller which will act as the input of the system.

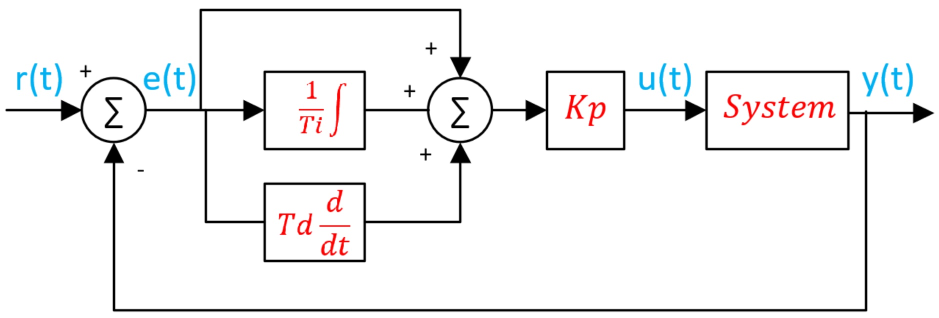


Figure 2.17: Feedback control loop using a Proportional Integral Derivative Controller.

The above block diagram leads to the signal u(t) being defined as:

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_\tau e(t) d\tau + T_d \frac{de(t)}{dt} \right) \tag{2.2}$$

Where $\mathrm{K_p}$ is the controller gain, $\mathrm{T_i}$ is the integration time and $\mathrm{T_d}$ is the differentiation time. These parameters can only have non-negative real values.

2.8.2 Discrete time PID controller

In the previous section the PID has been presented in its continuous time form. Nowadays, this type of controller is usually done via software using a microcontroller. This type of implementation is now of the discrete time domain instead of the continuous time domain, so what was showed in the previous section needs to be converted to this domain to be applicable. This can be done using the *Backward differences method* approximation which defines that[21]:

$$x(t) \simeq x(k) \tag{2.3}$$
$$\frac{dx(t)}{dt} \simeq \frac{x(k) - x(k-1)}{h} = \frac{1-q^{-1}}{h} x(k) \tag{2.4}$$
$$\int x(t)dt \simeq \frac{h}{1-q^{-1}} x(k) \tag{2.5}$$

Where k represents time in the discrete domain, h represents the sampling period and q^{-1} represents the delay operator. Applying this approximations to equation 2.2 turns out as follows:

$$u(k) = K_p \left(1 + \frac{1}{T_i} \frac{h}{1-q^{-1}} + T_d \frac{1-q^{-1}}{h} \right) e(k)$$
$$u(k) = K_p \left(\frac{1 + \frac{h}{T_i} + \frac{T_d}{h} - \left(1 + \frac{2T_d}{h}\right) q^{-1} + \frac{T_d}{h} q^{-2}}{1-q^{-1}} \right) e(k)$$

Considering the constants:

$$\begin{cases} s_0 & = K_p\left(1+\frac{h}{T_i}+\frac{T_d}{h}\right) \\ s_1 & = -K_p\left(1+\frac{2T_d}{h}\right) \\ s_2 & = K_p+\frac{T_d}{h} \end{cases}$$

The previous equation turns out to be:

$$u(k) = \frac{s_0 + s_1 q^{-1} + s_2 q^{-2}}{1-q^{-1}} e(k)$$

Rewriting this equation in order to u(k) but making it independent of the factor q^{-1} results in:

$$u(k) = u(k-1) + s_0 e(k) + s_1 e(k-1) + s_2 e(k-2) \tag{2.6}$$

This equation is now ready for the implementation of a PID controller on a digital device. Note however that for big error values the integral part of the controller should be deactivated so that the *integral windup* phenomenon does not occur. Also, it would be wise to add a low pass filter to the derivative term in order to avoid noise effects that may be amplified by it. Some systems do not require the use of all three terms (proportional, derivative, integral). In those cases, controllers like PI or PD might be enough. To obtain the equations for these cases the same reasoning must be applied but not considering the respective terms on the first equation.

2.8.3 PID tuning

The terms, h, K_p, T_i, T_d, are user defined so they must be chosen carefully in order to not damage the system. For that there are several theoretical techniques to find the most appropriate values of these parameters such as the *Closed Loop Ziegler-Nichols method.* This method states that, if the system is controlled only by a Proportional controller and the controller gain K_p is raised until the system output starts oscillating with a constant amplitude, when a step is applied on the input, that gain is called ultimate gain (K_{cr}) and the signal period is designated as oscillation period (T_{cr})[21][22]. Knowing these two values the PID parameters can be obtained by substituting these values on table 2.1.

Controller	$\mathbf{K_p}$	$\mathbf{T_i}$	$\mathbf{T_d}$	$\mathbf{K_I}$	$\mathbf{K_D}$
PID	$0.6K_{cr}$	$T_{cr}/2$	$T_{cr}/8$	$1.2K_{cr}/T_{cr}$	$3K_{cr}T_{cr}/40$

Table 2.1: Closed Loop Ziegler–Nichols PID tuning method (From [21]).

As an alternative, for systems whose dynamic behavior is known and the risk of damaging the system is very low or inexistant, these parameters can be set by trial and error. It is good practice to set K_p, T_i, T_d by this order. For that, starting with a value close to 0 of K_p and T_d and leaving T_i with a big value is a good way. These terms shall then be increased/decreased independently. For example, increasing only K_p is equivalent to apllying a Proportional controller, changing only K_p and T_d is equivalent to applying a PD controller and changing only K_p and T_i is equivalent to applying a PI controller.

2.9 CAN

The controller to be developed should communicate with the remaining robot parts via a CAN [23] network, so in this section a brief explanation about some important concepts of this protocol will be done.

The Controller Area Network (CAN) is a serial communication bus, originally created by Bosch in 1991, designed for robust and flexible performance in harsh environments, and particularly for industrial and automotive applications. Nowadays, CAN has multiple applications in areas such as communication between car subsystems, industrial applications, home automation, robotics, medical equipment etc.

CAN is a multi-master bus, where each bus node can start a transmission on its own and produce information. The communication is bidirecional half duplex, being information encapsulated in frames. Transmission is broadcast because one node can send messages to all other nodes at the same time. Each message has a unique ID that determines the message priority and consequently the bus access priority. One of the CAN architecture advantages is that it is easy to include a new device because there is no need to change the already existent structure. Only one new node has to be added to it.

As referred before CAN uses twisted pair differential communication. This has a big advantage, which is, it makes it more immune to electromagnetic interference, since the two wires are affected in the same way, so the diference of voltage in them remains the same. This is great because information in the signal is not affected. In order to create the signals of each wire, which may be denoted as CAN_H and CAN_L, usually is used a CAN transceiver, which may act as a transmitter or receiver. This transceiver is one of the two devices inside of each CAN node. The other device is a CAN controller, which may or not be contained in a microcontroller. In fact the CAN controller in one node is the one that transmits data to the transceiver (acting as a transmitter) through the pin Tx, being this signal converted to CAN_H and CAN_L by the transceiver. These two signals are sent to the other nodes through the CAN bus and received by that node transceiver (acting as a receiver) that converts CAN_H and CAN_L into a unique data signal sent to that node's CAN controller through the pin Rx. This concept is ilustrated in figure 2.18[24].

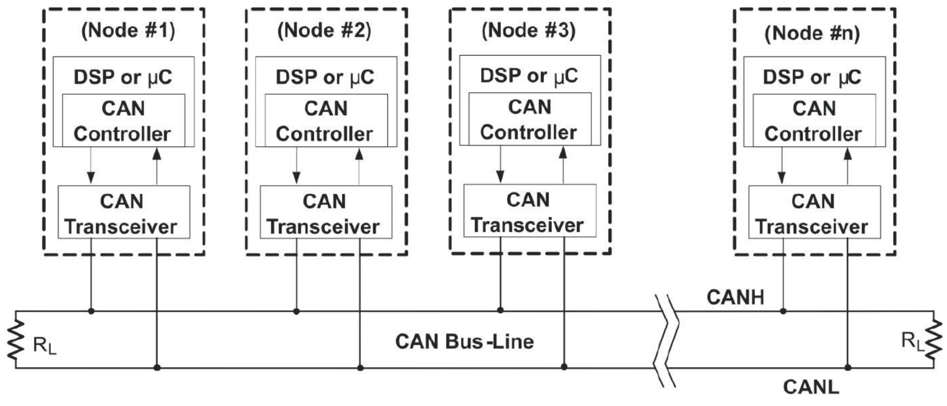


Figure 2.18: CAN network topology and node structure (From [24]).

Since CAN is a broadcast bus and a message transmitted through a CAN network is received by all the network nodes, that means that the CAN controller in each node has to read all the messages present in the bus. These messages are put in a temporary register designed MAB (Message Assembly Buffer). Each time a message is received by this register a filtering mechanism is applied so that only the relevant messages to a particular node are copied to the node's reception buffer, being the remainig ones discarded (this reduces the computational load on the microcontroller). Filtering is done by checking the message identifier bits. The valid bits are previously defined by software. Besides the referred buffers there also exists a buffer for messages to be transmited, a CPU interface with control registers, status and data buffers for communications. All of these "blocks" are represented in figure 2.19.

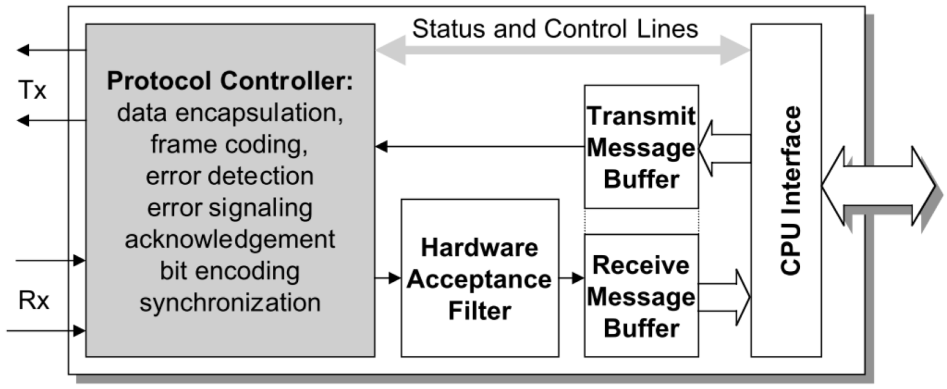


Figure 2.19: The inside of a CAN controller (From[24]).

Chapter 3

Architecture

This chapter describes the general architecture of the developed system meant to implement the robotic arm position controller.

3.1 System block diagram

This section provides an overview of the proposed solution to implement the arm controller, in terms of its main blocks and their interactions, starting with the block diagram presented in figure 3.1. A description of the function performed by each block is then presented in more detail.

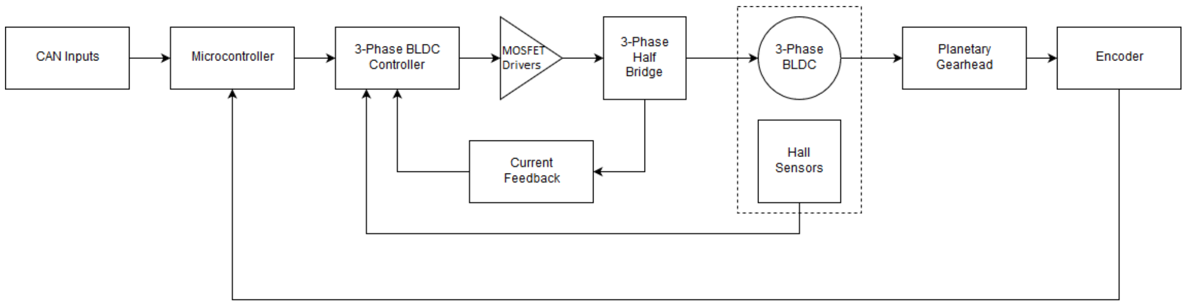


Figure 3.1: Block diagram of the robotic arm position controller.

3.1.1 CAN Inputs

This block represents the input variables passed through the CAN bus from the system main controller to the robotic arm controller. Since the purpose of this controller is to make the robot's shoulder move to a given position, within the range of 0 to 180 degrees, over a desired period of time, but with the speed being permanently controlled so that sudden movements do not occur, the controller needs three inputs. These are the maximum allowable speed the motor can reach, the time it must take to reach that value and the total time of the movement. These variables were chosen so that an adequate *speed profile* can be achieved. The concept of this profile will be further explained in the next section.

3.1.2 Microcontroller

The microcontroller is the central block of the system. It is responsible for controlling all of the system logic and ensure that the robot arm reaches its destined position in the required time. For that, it must use a PID controller algorithm that calculates the error value between the inputs and the value coming from the encoder. A desired PWM output is then produced based on the previous values.

3.1.3 3-Phase BLDC Controller

The 3-Phase BLDC controller receives information about position and direction of the motor from the Hall sensors, current feedback coming from the 3-Phase Half Bridge and the PWM signal sent by the microcontroller. Based on this information it sends control signals to the MOSFET Drivers so that they operate in the Half Bridge in such a way that it commutates the correct motor phases. The logic of this controller is in turn controlled by the microcontroller.

3.1.4 MOSFET Drivers

The MOSFET drivers are responsible for transforming low level signals such as the ones coming from the 3-Phase BLDC Controller into the required levels to operate the 3-Phase-Bridge.

There are many driver chips in the market but there are also 3-Phase BLDC controllers that come with built-in integrated drivers and that may be helpful, because they decrease the number of circuits which leads to a simplification of the the final system construction.

3.1.5 3-Phase Half Bridge

The 3-Phase Half Bridge is characterized for having 3 pairs of transistors that allow the motor to rotate clockwise or anti-clockwise, depending on which ones are ON and OFF at a given time instant. The state of the transistors is passed by the MOSFET drivers outputs.

3.1.6 Current Feedback

This block represents the current value that is sensed through a current sensor connected to the 3-Phase Half bridge and then compared to a reference value in the 3-Phase BLDC controller. This allows for the controller to limit the current to a maximum value and to control the relative amount of torque applied to the motor.

3.1.7 3-Phase BLDC and Hall Sensors

This block represents the 3-Phase BLDC motor itself. As the name suggests the motor has 3 phases due to its 3 windings. This motor has 3 integrated Hall-effect sensors and that is why in figure 3.1 the 3-Phase BLDC and the Hall Sensors blocks are represented within a dashed line rectangle.

The Hall-effect sensors allow the determination of the motor position, speed and direction in 60 degrees intervals. This feedback information is then used by the three-phase BLDC controller to commutate the motor windings.

3.1.8 Planetary Gearhead

The Planetary Gearhead is a mechanical arrangement which has the function of reducing the angular velocity and increasing the torque. The relevance of increasing torque is that it helps locomotion, otherwise the robotic arm would make sudden movements that would difficult the desired movement. Also, the increased torque allows to work with a higher weight limit. This block represents the physical output of the system. It is also at the gearhead output that the encoder, which provides the position of the robotic arm, is assembled.

3.1.9 Encoder

The Encoder block represents a rotary absolute magnetic encoder used to keep track of absolute position and control speed and direction of the motor with great accuracy. This information is then passed to the microcontroller that uses it to establish the main control of the system.

3.2 Position and speed control

Untill now the robotic arm controller was analyzed in a physical perspective. In this section it will be analyzed from a control perspective. For that, the previous block diagram can now be seen as the one represented in figure 3.2. The new blocks are individually explained below and related with the ones presented in the previous section for better understanding on how they relate with each other.

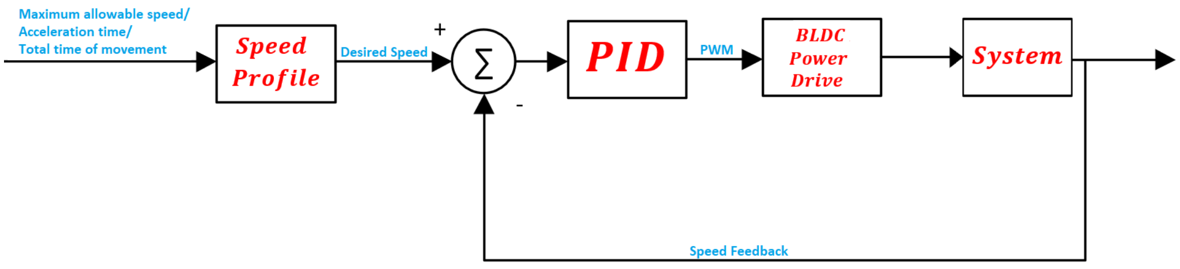


Figure 3.2: Closed loop speed control of the robotic arm.

3.2.1 Speed Profile

This block represents a software algorithm running in the microcontroler. This algorithm takes as inputs the maximum allowable speed the motor can reach, the time it must take to reach that value (acceleration time) and the total time of the movement. Note that this information is transmitted via CAN to the microcontroller and was represented by the CAN Inputs block back in figure 3.1. Thus, the algorithm implemented in this block is responsible for implementing a *speed profile* of the robot arm.

A speed profile consists in controlling the speed in such a way it starts at zero, rises to a certain level and decreases until it reaches the initial value, but doing all of this in a smooth way. The most common speed profiles try to achieve trapezoidal or sinusoidal forms. A sinusoidal profile is usually smoother than the trapezoidal case and tries to have a gradual increase until it reaches the maximum speed and then a gradual decrease until it reaches the starting value, being more complex to implement than the trapezoidal case. The trapezoidal case, also has gradual increases, by setting a stationary speed value, stays at the maximum speed usually for more than an instant and finally gradually decreases. The difference is that the group of gradual increases/decreases in the trapezoidal case ressemble a linear behavior. Figure 3.3 illustrates this case. This way this profile turns out to be less complex to implement than the sinusoidal case. In this work the speed profile that will be implemented is the trapezoidal case. Note that the speed and time values in figure 3.3 are not related to this work, being only an illustrative example.

In the profile example presented in figure 3.3 the blue dashed line represents the desired behavior of the speed in the speed profile. The red dashed lines divide the profile into three time slots. The first represents the time where the speed is rising from zero to a maximum value, the second represents a time where the speed has reached its maximum value and is constant and the third slot represents the time where the speed decreases until it reaches zero. In that example the first and third time slots have a duration of 10% of the time each, while the second time slot has a 80% duration of total time. There is no real controller that can set and achieve a desired speed instantly so the blue dashed line is a theoretical case. In practical cases an equivalent behavior to that is achieved using small steps. As represented by the black line, in each step the desired velocity is set, as well as the time during which that velocity must be maintained. The function of the speed profile block is to produce the desired setpoint velocity at a given time instant.

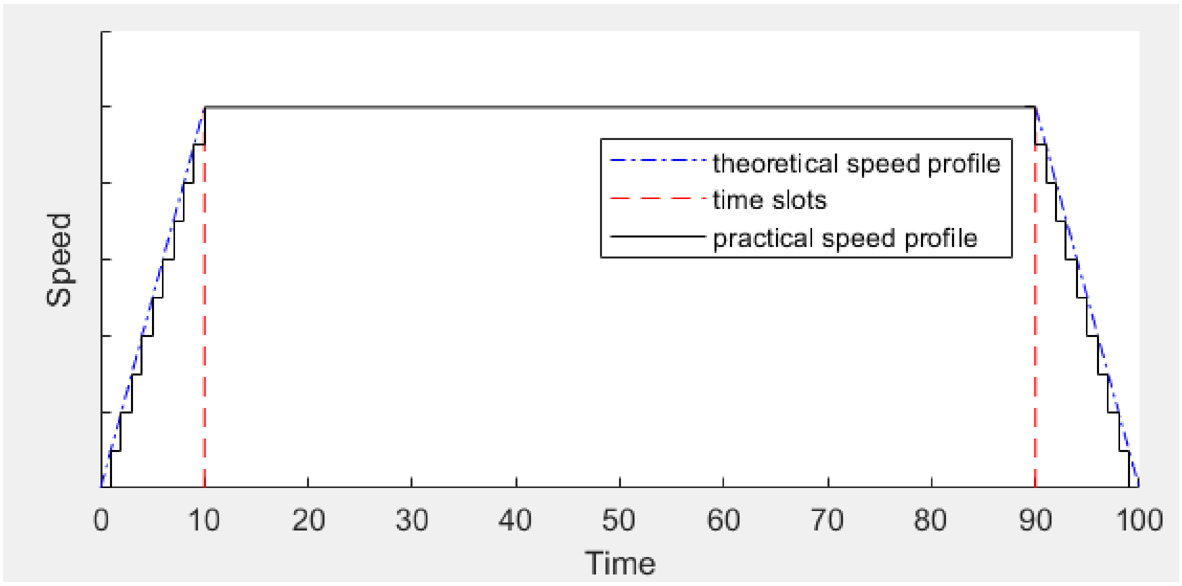


Figure 3.3: Trapezoidal speed profile.

3.2.2 PID

This block is also a software algorithm that implements a PID controller, which is responsible for setting the output speed value at the value desired by the Speed Profile block. This software algorithm is implemented by the microcontroller, which produces a PWM signal that will act in the BLDC Power Drive. A good tunning of the PID is essential for the speed profile to work as expected.

3.2.3 BLDC Power Drive

This block receives the PWM signal generated by the PID and transforms it in a sequence of signals that will act in the System and make it reach the desired stationary speed after a certain time. Back in figure 3.1 this block corresponded to the 3-Phase BDLC controller together with the Drivers and the 3-Phase-Half Bridge.

3.2.4 System to control

The System is composed of the motor, the gearhead and the associated load, so the System block can be seen as the 3-Phase BLDC together with the gearhead back in figure 3.1 because these need to be acted in such a way that they produce the desired output. The input of this block are the signals from the BLDC Power Drive that in the end will produce the desired speed value output.

Chapter 4

Implementation and prototype

This chapter starts by presenting the initial proposed setup to do this work. Following it, the choice of all the components used to implement the architecture mentioned in the last chapter and justifications are presented. After that, it shows the created prototype and preliminary results obtained with it. Finally the general algorithm of the robotic arm controller is presented. To complement this chapter, the Appendix A of this work presents a schematic of the final circuit that resulted from the tests done with the prototype.

4.1 Robotic arm setup

At the beginning of the proposed work, there was some initial setup that was presented to achieve this work objectives. This setup consists of a BLDC motor and its gearhead. They are both presented next.

4.1.1 Motor

For this project is used the motor EC-90 flat[1] which is illustrated in figure 4.1. This is a BLDC motor made by the company Maxon Motor, that has a power of 90W and comes with three integrated Hall-effect sensors. It can be powered from 4.5V to 24V. The main characteristics of this motor are represented in figure 4.2, which shows a part of its datasheet.

[1]`https://www.maxongroup.com/medias/sys_master/8825435389982.pdf`

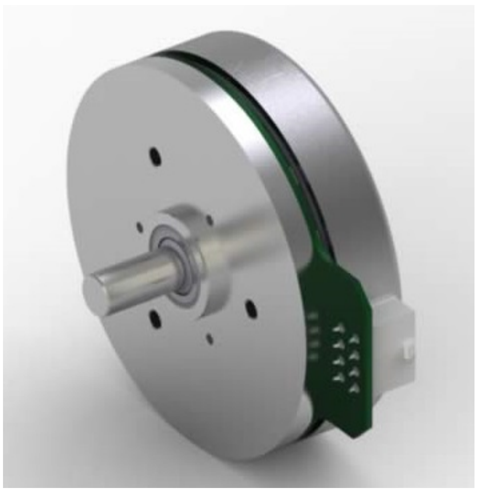
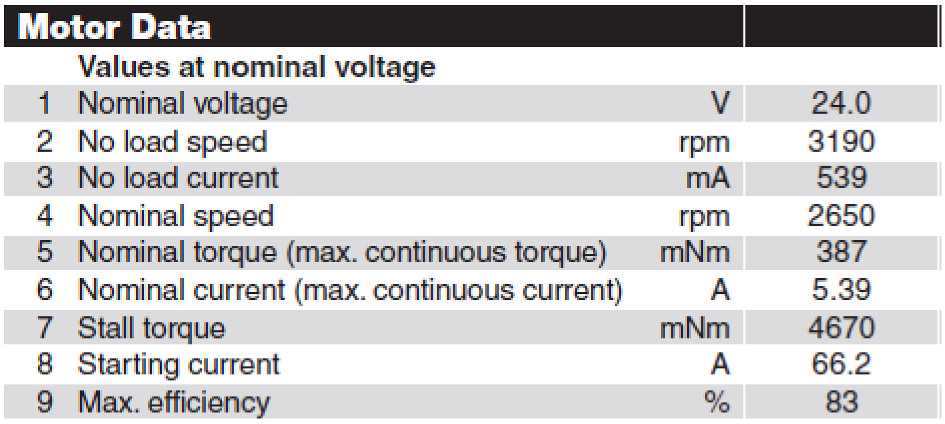


Figure 4.1: Representation of the BLDC motor EC-90 flat.



	Motor Data		
	Values at nominal voltage		
1	Nominal voltage	V	24.0
2	No load speed	rpm	3190
3	No load current	mA	539
4	Nominal speed	rpm	2650
5	Nominal torque (max. continuous torque)	mNm	387
6	Nominal current (max. continuous current)	A	5.39
7	Stall torque	mNm	4670
8	Starting current	A	66.2
9	Max. efficiency	%	83

Figure 4.2: EC90 Motor data.

To move the arm that is supposed to be integrated in the future, this motor must be able to produce enough torque. Torque is the applied force (F) times perpendicar distance from a pivot point (R) as shown in equation 4.1.

$$\tau = F.R = m.g.l \tag{4.1}$$

The arm has a weight (m) of 1 Kg and a length (l) of 0.7 m, so the required torque to move it without load, considering a gravitational acceleration of 9.8 m/s^2 is therefore $\tau =$ 6.86 Nm. Comparing this value with the nominal torque of figure 4.2 one can conclude that it is insufficient. This justifies the need of a planetary gearhead.

4.1.2 Planetary gearhead

The objective of the robotic arm controller is to accurately control the arm of the robot, so it is desirable that the arm does not make sudden movements. The planetary gearhead is ideal for that and so for this project is used one, which is represented in figure 4.3, that has a

66:1 reduction factor (N) and a maximum continuous torque of 30Nm[2]. That reduction factor means that, for the gear that supports the arm to do 1 revolution the motor itself needs to do 66 revolutions.

Figure 4.3: Representation of the gearhead.

With this gearhead the torque applied by the the motor together with the gearhead is expressed by equation 4.2 which gives a total torque value of 25.5 Nm, a value more than enough to move the arm without load (considering no losses in the process).

$$\tau' = N.\tau \tag{4.2}$$

Considering now the arm with load on its tip, equation 4.1 has to be substituted by 4.3, where the gravitational force applied in the load is now taken into account (considering no losses and that the perpendicar distance from a pivot point is the same for both the arm and the load).

$$\tau = F.R + F_{load}.R = m.g.l + m_{load}.g.l \tag{4.3}$$

The motor together with the gearhead could produce enough torque for a maximum load of 2.71Kg.

4.2 Component selection

The first step is to properly select the components to use. Nowadays, there is a great diversity and quantity of solutions available, however, there are solutions that prove better than others for a specific case.

There are solutions that present better quality than others, but this quality causes that their price is increased. Depending on the application and the intended purpose of the application, a high quality of the component may be required. Otherwise, if this quality is not required, it will be important to opt for a lower price component.

Next, the main components chosen for the robotic arm controller implementation are presented.

[2]`https://www.maxongroup.com/medias/sys_master/8831071289374.pdf`

4.2.1 Encoder

The chosen encoder is denoted as AEAT-6012-A06 and it is a 12 bit absolute magnetic encoder, which is equivalent to 4096 positions per revolution[3].

4.2.2 3-Phase BLDC controller and drivers

For the 3-Phase controller the option of a controller with integrated drivers was chosen in order to simplify the implementation process. For that an Allegro MicroSystems, LLC controller designated A3931 was chosen [25]. This controller is described as an "Automotive 3-Phase BLDC Controller and MOSFET Driver". Some of its main characteristics are.

- High current three-phase gate drive for N-channel MOSFETs
- Charge pump and top-off charge pump for 100% PWM
- Integrated commutation decoder logic
- Operation over 5.5 to 50V supply voltage range

This is a good solution since it saves the time needed to create a 3-Phase BLDC Controller or MOSFET Drivers and make them work together as expected. The cost is also affordable and less expensive than the previous alternative. Comparing it with some other similar alternatives available in the market, this device comes with extra control logic for diagnostics and protection that might help along the work. Also, more complex solutions come with internal microprocessors that would have the need to spend a good amount of time learning to program them, when they are not necessarily needed.

With this device the user's job is simplified, due to its integrated drivers the one designing the circuit only has to worry about choosing the correct MOSFETs for the 3-phase half bridge, an adequate sense resistor, and some external acessory components in order to have all the necessary hardware to control and drive the BLDC motor. A schematic that links all the referred components can be found in Appendix A.

4.2.3 3-Phase half bridge

To build the 3-phase half bridge there was the need to choose adequate transistors to fit the application demands.

Since the motor will be used essentialy with high speeds (at maximum 3190 RPM) so that the gearhead can move the robotic arm from positions of 0 to 180 degrees, this application demands that the motor is to be commutated fast, so the use of high speed MOSFETs is greatly advised to fulfill the requirements. Other aspect that must be taken into account is that these MOSFETs must be poweful enough to support both the power source voltage, which will be a 16V battery, the starting current of the motor (I_{Mmax}) which is 66.2A and a nominal current (I_M) of 5.39A.

Considering these aspects, the IRFS4115TRLPBF [4] power MOSFET was chosen. In its datasheet this MOSFET is classified for "High Speed Power Switching" applications such as this one. Also, it supports a maximum drain current of 195A and a drain to source

[3]`https://www.mouser.com/ds/2/678/Avago_05182016_DataSheet-1217255.pdf`

[4]`https://www.infineon.com/dgdl/irfs4115pbf.pdf?fileId=5546d462533600a401535636e5d2218f`

breakdown voltage of 150V. Both values greatly exceed the motor demands. Besides this, characteristics such as the maximum static drain-to-source on-resistance ($R_{DS(on)}$) of 12,1mΩ, input capacitance (C_{iss}), turn-ON and turn-OFF times were taken into account. All this four characteristics are desirable to be as low as possible. The times and the input capacitance are associated to the switching speed and the internal resistance is associated with power dissipation and the junction's temperature. When compared with other alternatives available in the market this MOSFET proves to be a good solution to this specific application due to its parameters.

Calculations done concerning the MOSFET power loss ($P_{Loss(MOSFET)}$) and the maximum MOSFET reached junction temperature (T_J) considering equations 4.4 and 4.5, respectively lead to values of $P_{Loss(MOSFET)} = 0.35W$ and $T_J = 39°C$ (considering an ambient temerature of 25°C and the junction to ambient thermal resistance $R_{\theta JA} = 40°C/W$).

$$P_{Loss(MOSFET)} = R_{DS(on)}.{I_M}^2 \tag{4.4}$$

$$T_J = R_{\theta JA}.P_{Loss} + T_{Ambient} \tag{4.5}$$

With the MOSFETs chosen, the values of the bootstrap capacitor (C_{Boot}) and the resistance (R_{dead}) that controls the dead time (t_{dead}) of the BLDC controller could then be calculated using the expressions 4.6, 4.7 and 4.8. These equations lead to $C_{Boot} = 154nF$, $t_{dead} = 0.16us$ and $R_{dead} = 3,67k\Omega$. Note that C_{Boot} was calculated considering $V_{Boot} = 10V$ and $Q_G = 77nC$ and t_{dead} was calculated considering $t_{d(off)} = 41ns$ and $t_f = 39ns$.

$$C_{Boot} = Q_G.\frac{20}{V_{Boot}} \tag{4.6}$$

$$t_{dead} = 2 * (t_{d(off)} + t_f) \tag{4.7}$$

$$R_{dead} = \frac{2000}{(\frac{33}{(t_{dead} - 0.1)} - 5)} \tag{4.8}$$

4.2.4 Current sensing

The 3-Phase BLDC controller chosen (A3931) comes with the sensing part integrated in it, being the person that is designing the circuit only responsible for choosing the adequate value of the sensing resistors. Considering 4 parallel sensing resistors of 25mΩ, equation 4.9 leads to a maximum power dissipation $P_{Loss(R)} = 182mW$. A solution with 4 parallel sensing resistors of 25mΩ, 7W each was adopted. These resistors are of the WSHM2818 series[5].

$$P_{Loss(R)} = R.\frac{I_M}{4}^2 \tag{4.9}$$

[5]`http://www.vishay.com/docs/30188/wshm2818.pdf`

4.2.5 Microcontroller

As referred in the previous chapter, the microcontroller is essential as it is directly involved in tasks such as the CAN communication, is responsible for generating an adequate PWM signal to make the motor rotate with specific speed and direction, read the values generated by the encoder and set the appropriate digital values in the BLDC controller pins.

To address the needs, the microcontroller chosen to operate was the PIC32MX795F512H[6], developed by Microchip. It is able to perform all required tasks. It provides enough digital pins to set or receive values from the other devices of the robotic arm controller; a UART for serial communications with a PC, timers to generate different periodical events such as reading the encoder output, send CAN messages, execute the PID algorithm and generate the frequency for the PWM signal that is to be applied to the BLDC controller; a PWM output to act in the motor and a CAN controller. Its operating conditions are 2.3V to 3.6V, -40°C to +105°C, DC to 80 MHz. Besides being able to perform all required tasks the author of this work already has experience with this microcontroller and it is also used in the working group. This helped on the choice against many other possibilities available in the market that could also address the needs.

In order to program this microcontroller through a computer the chip FTDI FT232RQ[7] is used to implement a USB to serial RS232 interface.

4.2.6 CAN Transceiver

A typical CAN node consists of a CAN controller that implements the CAN protocol and a transceiver that implements the interface to the physical medium. For that, the chosen circuit was the MAX3051[8] transceiver from MAXIM.

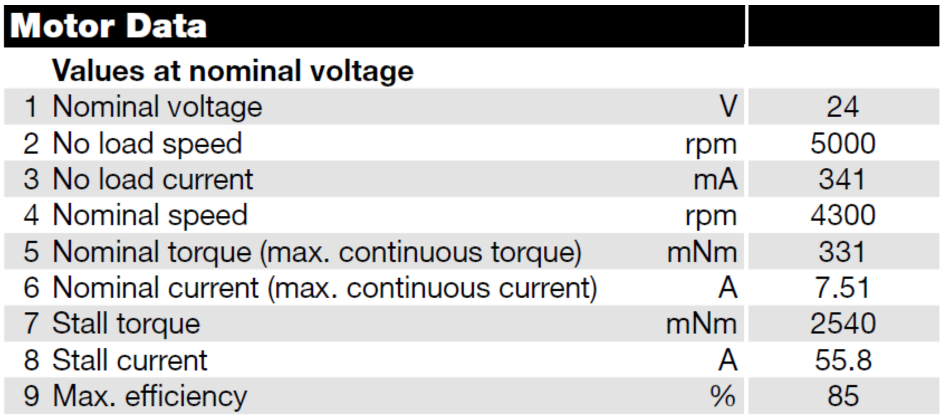
4.3 Setup assembly and preliminary tests

Due to some problems that ocurred during the work the proposed motor (EC90 Flat) and gearhead had to be substitued by the BLDC motor EC45[9], 250W without a gearhead, which is not suitable for the final application. However, it was the only solution available and it is still useful to obtain some of the work needed results. This motor was also made by Maxon Motor. When comparing it with EC90 Flat it presents less nominal torque and a bigger nominal current, which are worse than in the EC90 Flat case. However, it presents a little more efficiency and also operates in the same voltage range of 4.5V to 24V. Its main characteristics are presented in figure 4.4. Even with the change of the motor parameteres, this motor is still able to operate in the already designed circuit considering EC90 Flat.

All the preliminary results present in this section were obtained powering the circuit with a 16V battery.

The complete setup consists of the main board, where all components are connected, and externally the motor with integrated Hall sensors and the encoder. Figure 4.5 shows this setup with the main board on the left and the rest on the right.

[6]`http://ww1.microchip.com/downloads/en/devicedoc/61156g.pdf`
[7]`https://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf`
[8]`https://datasheets.maximintegrated.com/en/ds/MAX3051.pdf`
[9]`http://storkdrives.com/wp-content/uploads/2013/10/maxon-EC-motor-catalog-datas-1.pdf`



Motor Data			
	Values at nominal voltage		
1	Nominal voltage	V	24
2	No load speed	rpm	5000
3	No load current	mA	341
4	Nominal speed	rpm	4300
5	Nominal torque (max. continuous torque)	mNm	331
6	Nominal current (max. continuous current)	A	7.51
7	Stall torque	mNm	2540
8	Stall current	A	55.8
9	Max. efficiency	%	85

Figure 4.4: EC45 motor data.

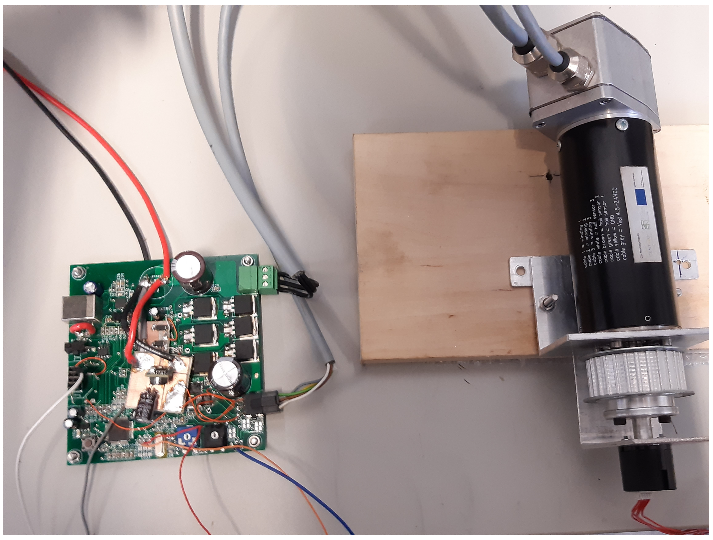
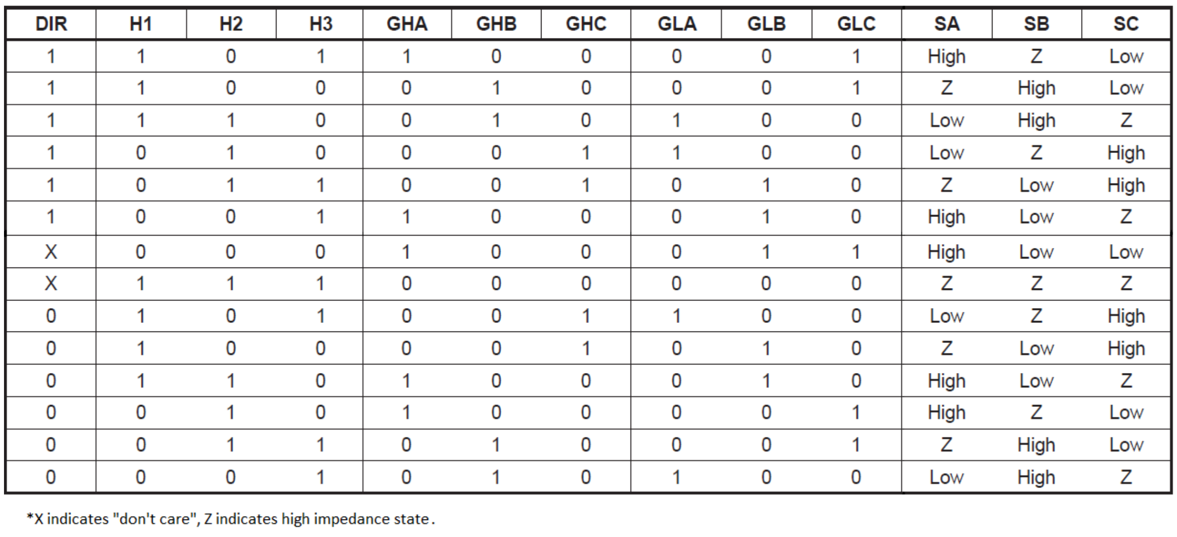


Figure 4.5: Setup used to test the robotic arm controller.

In order to verify that the various parts of the system were functioning correctly, several partial tests were performed. First was connected a computer to the FT232RQ via USB and verified that the computer recognized this device. Next, a bootloader was put into the microcontroller. Several mid level functions were created in order to configure timers, interrupts, PWM, UART, activate and read/write inputs/outputs. After this, the setup was tested without its power part connected. The major aspects of this test was to understand if, when rotating the motor manually, the Hall-effect sensors retrieved the expected values to the BLDC controller. The controller reacted putting the adequated value in its direction and commutation pins, connected to the microcontroller pins and LEDs were used to indicate their state. Besides this, the encoder and all remaining non-power devices were tested. This ensured that everything was fine with the non-power part of the circuit. For that, the table represented in figure 4.6, which was adapted from the BLDC controller datasheet was essential to analyze the obtained results of the preliminary tests. These results will be presented in the next subsections and are related to this table.

Looking to the table in a simple way, the rows in which DIR=1 represent the states, from top to bottom, when the motor is rotating clockwise. On the other hand, when DIR=0, the motor is rotating counterclockwise and the states must be viewed from bottom to top. Columns H1 to H3 represent the outputs of the Hall-effect sensors, GLA to GLC represent the inputs of the low side gates of the 3-phase bridge, GHA to GHC represent the inputs of the high side gates and finally SA to SC represent the state voltage of the motor windings. In a brief explanation of the table, considering the first row, Hall-effect sensors H1 and H3 report 1 and H2 reports 0, which represents a specific rotor position. According to this, GHA=1 and GLC=1, meaning that this two MOSFETs are conducting and so they can be seen as a low-resistance circuit creating a current path from the power source, crossing the high side MOSFET A, passing through the motor windings A and C, then passing through the low side MOSFET C and getting to ground, while all other MOSFETs are OFF. In this situation the motor winding A is energized, motor winding C is energized with opposite polarity and motor winding B is left in high impedance state making the motor rotate to the next position, as previously explained in Chapter 2. Note that there is no row in the table where the high and low side gates from the same pair of MOSFETs are ON, because that would lead to *cross conduction*. Notice however that the controller can receive a combination of all 1s or all 0s from the Hall-effect sensors and treats them as an error or pre-positioning code respectively.

For the power part of the circuit, the MOSFETs were assembled. Then a PWM signal with a specific duty cycle value was generated in the microcontroller and applied at the BLDC controller in order to make the motor rotate at constant speed.



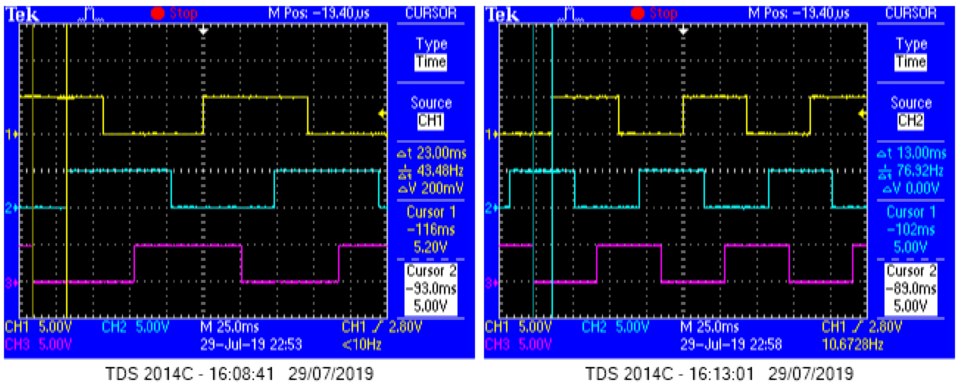
DIR	H1	H2	H3	GHA	GHB	GHC	GLA	GLB	GLC	SA	SB	SC
1	1	0	1	1	0	0	0	0	1	High	Z	Low
1	1	0	0	0	1	0	0	0	1	Z	High	Low
1	1	1	0	0	1	0	1	0	0	Low	High	Z
1	0	1	0	0	0	1	1	0	0	Low	Z	High
1	0	1	1	0	0	1	0	1	0	Z	Low	High
1	0	0	1	1	0	0	0	1	0	High	Low	Z
X	0	0	0	1	0	0	0	1	1	High	Low	Low
X	1	1	1	0	0	0	0	0	0	Z	Z	Z
0	1	0	1	0	0	1	1	0	0	Low	Z	High
0	1	0	0	0	0	1	0	1	0	Z	Low	High
0	1	1	0	1	0	0	0	1	0	High	Low	Z
0	0	1	0	1	0	0	0	0	1	High	Z	Low
0	0	1	1	0	1	0	0	0	1	Z	High	Low
0	0	0	1	0	1	0	1	0	0	Low	High	Z

*X indicates "don't care", Z indicates high impedance state.

Figure 4.6: BLDC controller commutation truth table (Adapted from [25]).

4.3.1 Hall-effect sensors

Figures 4.7a and 4.7b represent the Hall-effect sensors response for both the clockwise and counterclockwise cases, respectively. The two vertical lines (yellow for the clockwise case and blue for the counterclockwise case) represent the time interval of a state. For the clockwise case the state represented in this time interval, in the form H1-H2-H3, is 1-0-0, followed by 1-1-0, and so on, exactly matching the commutation truth table shown in figure 4.6. With the same reasoning the state represented in the counterclockwise case cursor interval is 0-1-0, followed by 1-1-0 and so on.



(a) Clockwise direction - Yellow: H1, Blue: H2, Purple: H3.

(b) Counterclockwise direction - Yellow: H1, Blue: H2, Purple: H3.

Figure 4.7: Hall-effect sensors commutation states.

4.3.2 Commutation and direction

TACHO and DIRO are two output pins of the BLDC controller which can be seen as the commutation and direction pins respectively, which provide speed and direction information. If DIRO reports the value 1 it means that the motor is rotating clockwise and if DIRO reports the value 0 it means that the motor is rotating counterclockwise. TACHO output changes state each time there is a winding commutation. This behavior can be observed in figure 4.8, where TACHO starts constant, which means the motor is stopped and DIRO is 0. Then TACHO commutates and DIRO is still 0, which means the motor started rotating counterclockwise. After a while the motor has stopped again and then started rotating clockwise because DIRO is now 1 and commutations exist in TACHO. Note that the value of DIRO is meaningless when TACHO is not commutating.

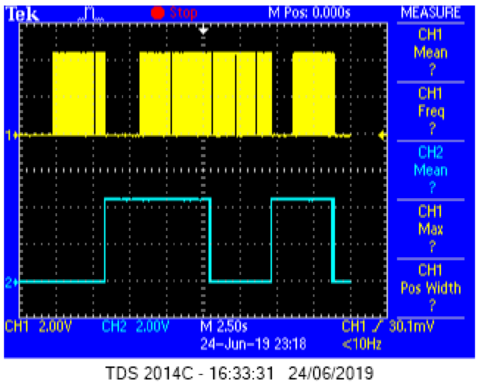


Figure 4.8: Commutation and direction outputs - TACHO: yellow, DIRO: blue.

4.3.3 Encoder

Three signals related with the encoder communication were obtained. These signals are represented in figure 4.9. The signals are the chip select (yellow), the clock (blue) and the serial data output (purple). After chip select is enabled (active low) the clock signal remains at logic state 1 during $1us$ and then starts acting like a clock for twelve periods. The data output value is validated at each falling edge of the clock. The combination of the twelve output bits is the value of the absolute position of the encoder. The first bit to be transmited is the most significant bit. So, the encoder position in the figure is given by 000000011010 in binary which corresponds to 26 in decimal.

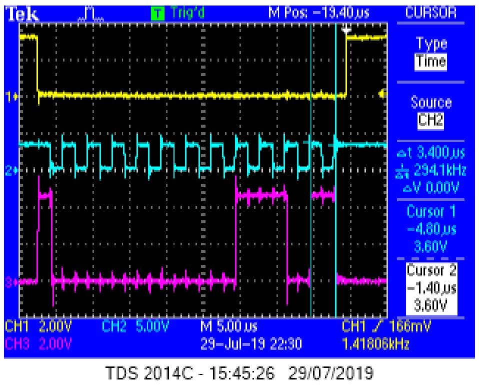


Figure 4.9: Encoder signals - chip select: yellow, clock: blue, data output: purple.

This absolute encoder has 12 bits, so the values provided are in the range 0 to 4095 (which means a resolution greater than the tenth degree). This presents a problem for this application since there is a discontinuity in the transition between 0 and 4095 or vice-versa. This problem can be perceived considering the following example. Considering that generally the encoder position increments when the motor is rotating clockwise and decrements when is rotating counterclockwise, the naive approach would be that if the actual position is greater than the previous position, the motor is rotating clockwise and in the opposite case it is rotating counterclockwise. However, this is not true because when it rotates clockwise and crosses from 4095 to 0, it decrements. The reverse would occur for the counterclockwise case. This problem can be solved converting an absolute encoder into an incremental encoder via software.

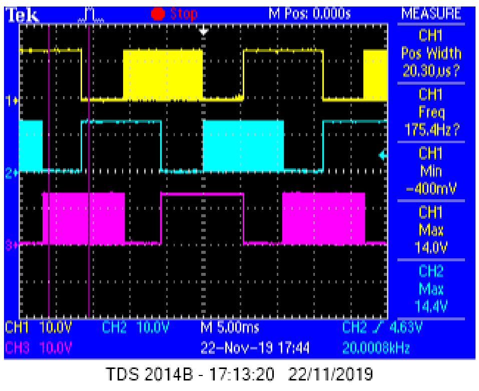
Figure 4.10 shows one possible solution. In the figure pseudocode consider that getAbsolutePositionFromEncoder() is a function that returns the position of the motor shaft given by the absolute encoder. DELTAT is bigger than the time needed to execute one cycle of code and smaller than the time needed for the motor shaft to make half rotation at the highest motor speed. At each interval DELTAT if the difference between the actual position (pos) and the previous position (prevPos) is less than half of the encoder resolution ($2^N/2$, where N is the number of bits of the encoder) no discontinuity ocurred. Otherwise, it is greater than half of the encoder resolution and a discontinuity is detected. Depending on the signal of the difference pos-prev being positive or negative one can know if the motor was rotating counterclockwise or clockwise, respectively.

```
$prevPos = getAbsolutePositionFromEncoder()$
$At\ Each\ DELTAT$
    $pos = getAbsolutePositionFromEncoder()$
    $delta = (pos - prevPos) + 2^N,\ \ if(pos - prevPos) < -2^N/2$          $//discontinuity\ ocurred$ (CW $case$)
    $delta = (pos - prevPos) - 2^N,\ \ if(pos - prevPos) >= 2^N/2$          $//discontinuity\ ocurred$ (CCW $case$)
    $delta = (pos - prevPos),\ \ \ \ \ \ \ if(pos - prevPos) \in [-2^N/2, 2^N/2[$   $//no\ discontinuity\ ocurred$
    $prevPos = pos$
$End$
```

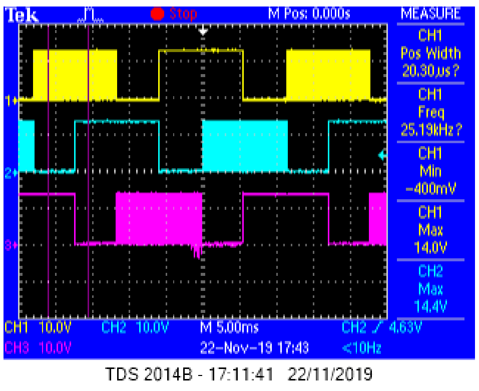
Figure 4.10: Pseudocode to transform absolute position information into incremental position information.

4.3.4 Motor windings

After assembling the MOSFETs, the power part of the circuit was tested. A PWM signal was generated in the microcontroller and applied to the BLDC controller in order to run the motor at constant speed commutating its phases correctly, according to the table of figure 4.6. Figures 4.11 and 4.12 represent the states of the low gates (GL), and motor windings (S) respectively for both clockwise and counterclockwise case. It is easy to verify that the states trasitions are in accordance with figure 4.6 where the sequence is A, B, C, A, etc. while rotating clockwise and C, B, A, C, etc while rotating counterclockwise. Figures 4.13a to 4.13c, show what is happening in the three high sides gates (GH) when each low side gate is operating. Figure 4.13d shows a zoomed view of the signals in 4.13a, where it can be observed that GLA and GHA are never active at the same time.

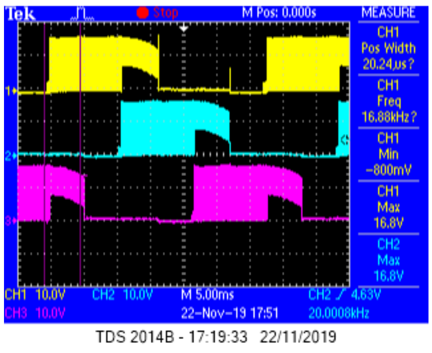


(a) Clockwise direction - GLA: Yellow , GLB: Blue ,GLC Purple.

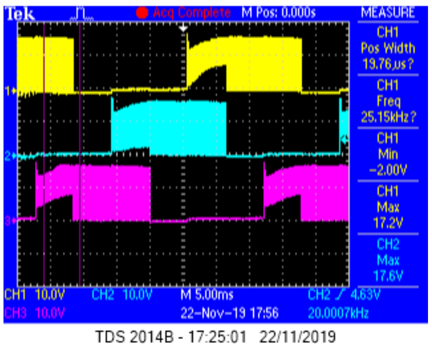


(b) Counterclockwise direction - GLA: Yellow, GLB: Blue, GLC: Purple.

Figure 4.11: Low gates (GL) commutation states.

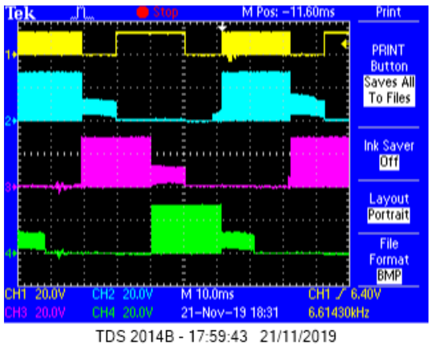


(a) Clockwise direction - SA: Yellow, SB: Blue, SC: Purple.

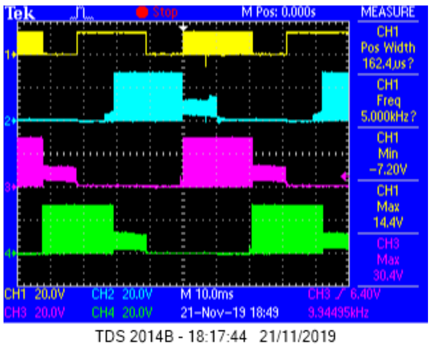


(b) Counterclockwise direction - SA: Yellow, SB: Blue, SC: Purple.

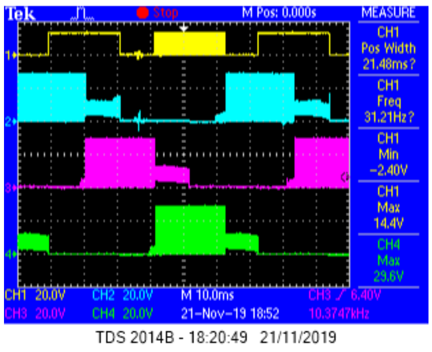
Figure 4.12: Motor windings (S) commutation states.



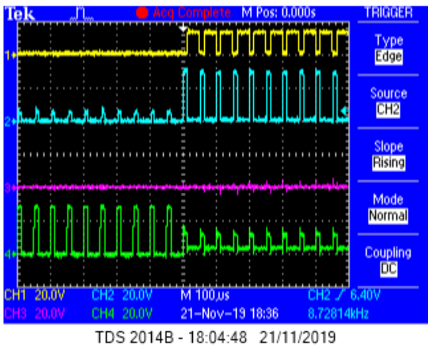
(a) GLA: Yellow, GHA: Blue, GHB: Purple, GHC: Green.



(b) GLB: Yellow, GHA: Blue, GHB: Purple, GHC: Green.



(c) GLC: Yellow, GHA: Blue, GHB: Purple, GHC: Green.



(d) Zoomed view - GLA: Yellow, GHA: Blue, GHB: Purple, GHC: Green.

Figure 4.13: High gates (GH) commutation states when each low side gate (GL) is operating.

4.4 General algorithm of the robotic arm controller

In order to make the robotic arm controller work as initially expected a suitable algorithm had to be created. This algorithm is represented in figure 4.14, in the form of a flowchart. The program has two different processes running in it, one consists in an infinite loop and the other consists in an interruption that occurs at each 1 milisecond interval. The code present in the interrupt takes care of the speed profile and PID algorithms and can change the state of the variable "flag". If flag=0, it indicates that a speed profile is in execution. In the speed profile the variable "i" is incremented, and when it finishes after a certain number of interrupt events (related with the total time of movement and a condition "i"="Imax" is verified) "i" is set to 0 and "flag" is set to 1 signaling that the speed profile that was in execution has finished. In addition, every 20 interrupt events the "tick_20ms" variable is set to 1. This signalizes that a CAN message should be sent to the CAN network. The infinite loop is responsible for sending, receiving and validate the received CAN messages. When "tick_20ms" is 1, a new message is sent from the robotic arm controller to the CAN network and "tick_20ms" is set to 0. For a CAN message to be received it must have one of two acceptable IDs. One ID is used for priority messages and other used for non-priority messages. Any other message in the CAN network that may have a different ID is rejected. Besides this, if it is received a message ID of one priority message, that message shall go to the validation check. This validation process is related with the fulfillment of the speed profile conditions. For example, if a message is received and its data field contents suggest that the received accelaration time is bigger than the total time of movement, the message is not validated. If it is valid "i" is set to 0, signalizing that the current speed profile being processed (if any) is to be aborted and a new one must start. Also, "flag" is set to 0 to signalize that a speed profile starts execution. For a non-priority message the procedure is identical, with the difference that if the current speed profile being processed has not finished, these messages are ignored and no other speed profile can start untill the current one is finished. Otherwise, if the current speed profile has finished and the message is valid a new one must start and flag is set to 0.

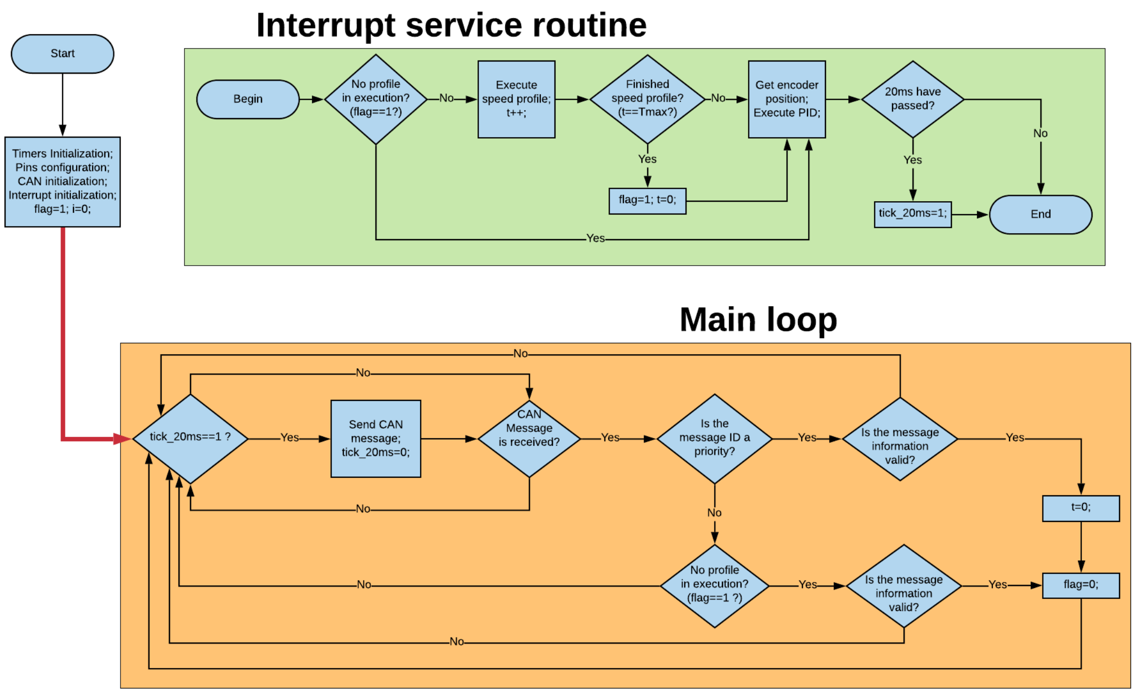

Interrupt service routine
Start
Timers Initialization; Pins configuration; CAN initialization; Interrupt initialization; flag=1; i=0;
Begin
No profile in execution? (flag==1?)
No
Execute speed profile; t++;
Finished speed profile? (t==Tmax?)
No
Get encoder position; Execute PID;
20ms have passed?
Yes
flag=1; t=0;
Yes
tick_20ms=1;
No
End
Yes
Main loop
No
No
tick_20ms==1 ?
Yes
Send CAN message; tick_20ms=0;
CAN Message is received?
Yes
Is the message ID a priority?
Yes
Is the message information valid?
Yes
No
t=0;
No
No profile in execution? (flag==1 ?)
No
Yes
Is the message information valid?
Yes
flag=0;
No

Figure 4.14: Flowchart of the implemented algorithm of the robotic arm controller.

A valid CAN message received by the robotic arm controller must contain information about the speed profile to execute. This information should be in the first 6 bytes of data. The information is divided in 2 bytes for the speed value in RPM, 2 bytes for the acceleration time in miliseconds and 2 bytes for the total time of movement in miliseconds. These values shall be interpreted as 2's complement numbers. What decides if a message contains priority priviledges is its ID, which means that the data structure of priority and non-priority messages is equal.

A CAN message sent from the robotic arm controller to the CAN network contains information about the arm's current position in degrees and also the logic level of the variable "flag", which indicates if a profile is being executed or not.

Chapter 5

Experimental results

This chapter presents the main results obtained in this work. It begins by presenting a characterization of the system while in open loop and the response of it in closed loop when controlled by a tuned PID. Next is shown the system response to different speed profiles. Finally, the general algorithm in which the robotic arm controller operates is presented together with a practical example.

5.1 System characterization

To characterize the robotic arm controller system a study was made by puting it in open loop without load. A PWM signal with a duty cycle value between 0 and 100% was applied to the system and 1 second was waited so that the system could reach a stationary state response for both the clockwise (CW) and counterclockwise (CCW) direction cases. Figure 5.1 shows an example of this procedure. This procedure was repeated for all possible duty cycle values. Note that in all experiences the motor starts from a rest state. This allowed the knowledge of the average speed value achieved in each case and also the maximum variation of the reached speed value. This variation is obviously undesirable, but also real and so should be taken into account.

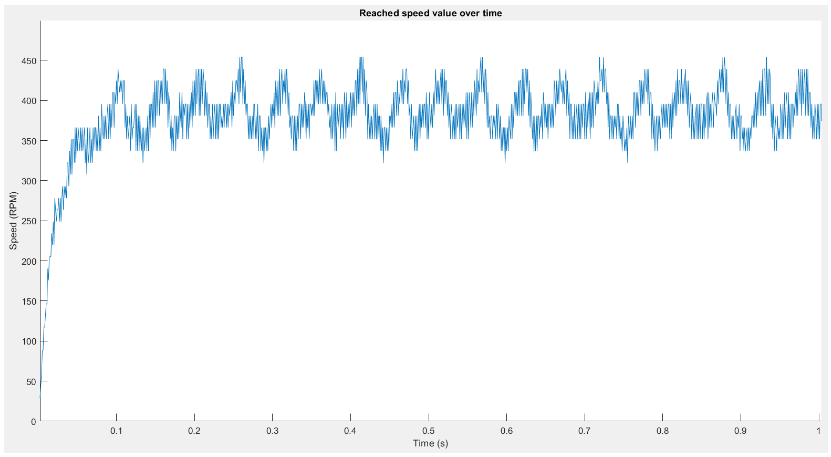
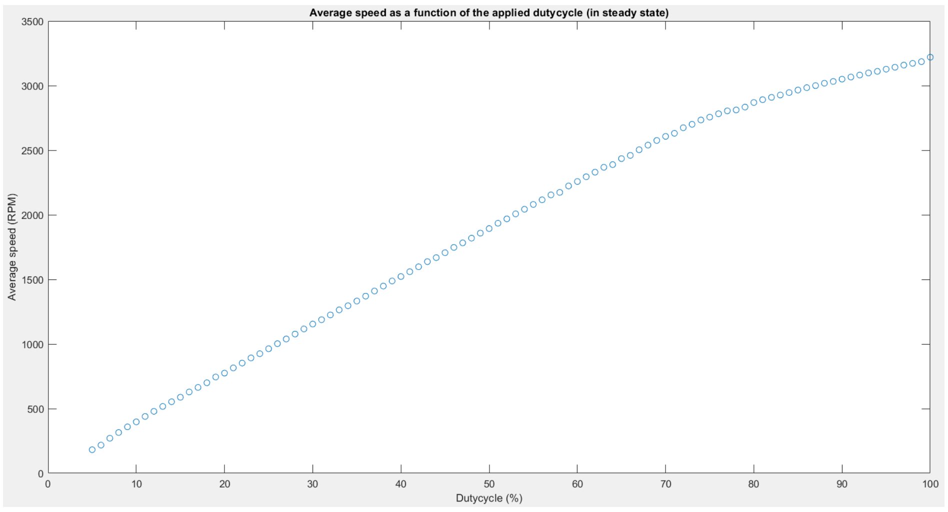
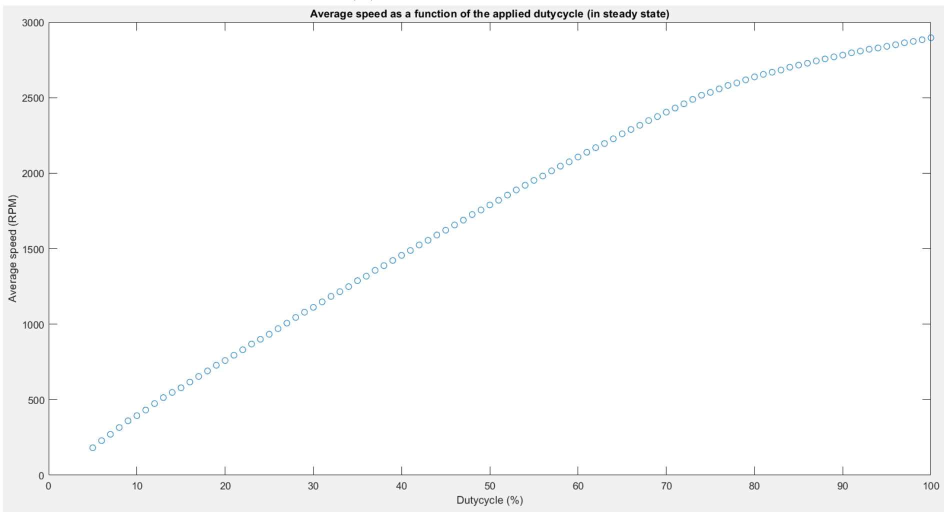


Figure 5.1: Step response for the 10% duty cycle case.

Figure 5.2 presents all the steady state speed values reached after applying the respective duty cycle value. Values below 5% are not represented as the motor can not move when parting from a idle state. To obtain these graphics, 100 samples were taken for each duty cycle value and the mean of these samples was calculated, so each point represents the mean value. Analyzing the obtained results, for both the CW and CCW cases, which are identical, it can be seen that for all values greater than 5% it exhibits approximately linear behavior. For the CW motion case the maximum value achieved is 3200 RPM. On the other hand for the CCW motion the maximum value achieved is 2800 RPM.



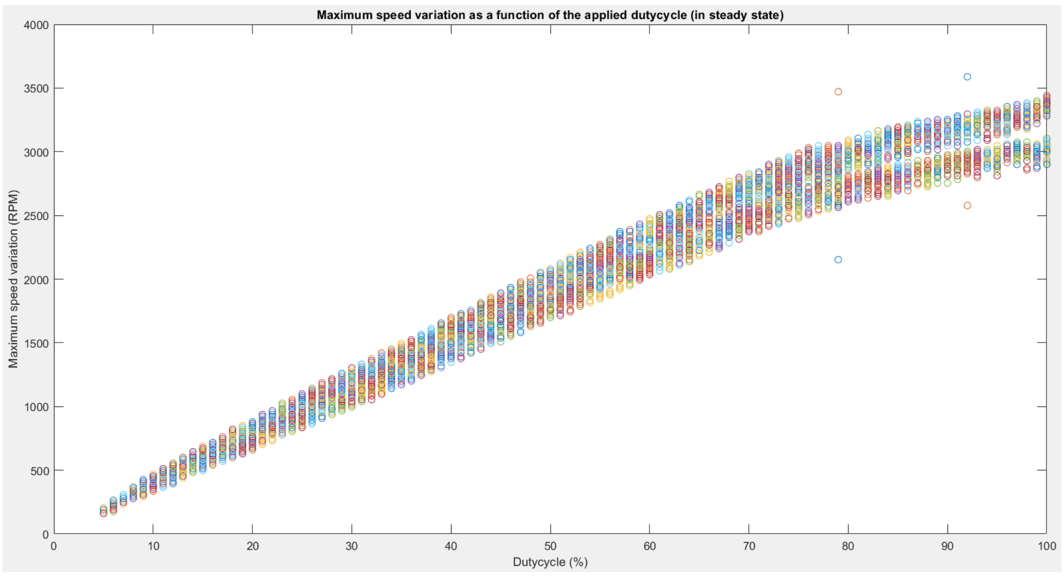
(a) Clockwise motion.



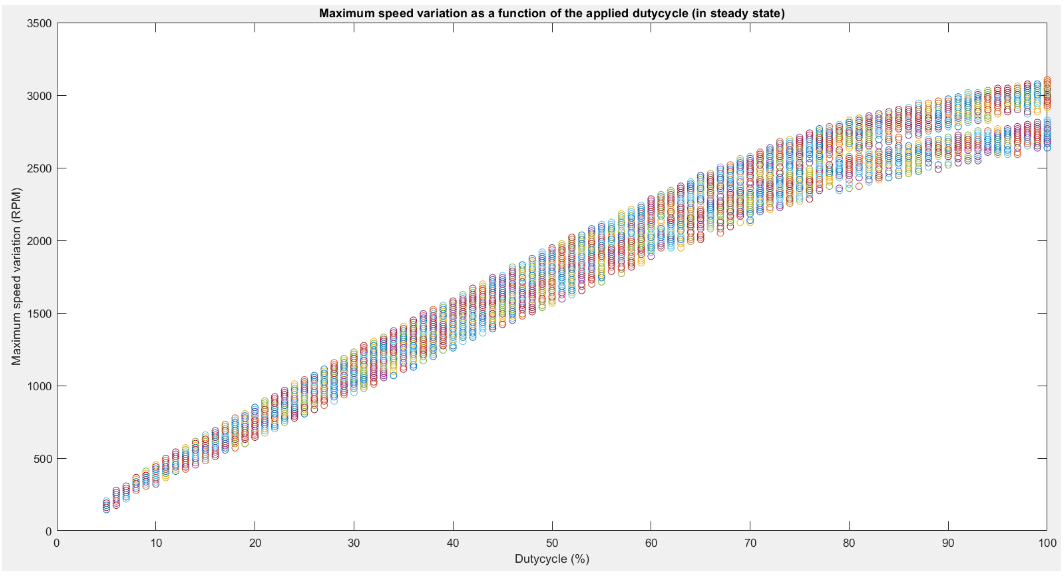
(b) Counterclockwise motion.

Figure 5.2: Average steady state speed reached for each duty cycle value between 5 and 100%.

Figure 5.3 shows that increasing the applied duty cycle value increases the variation in turn of the reached speed value for both the CW and CCW cases. The values represented in figure 5.3 are the raw speed values, that is, without any processing.



(a) Clockwise motion.

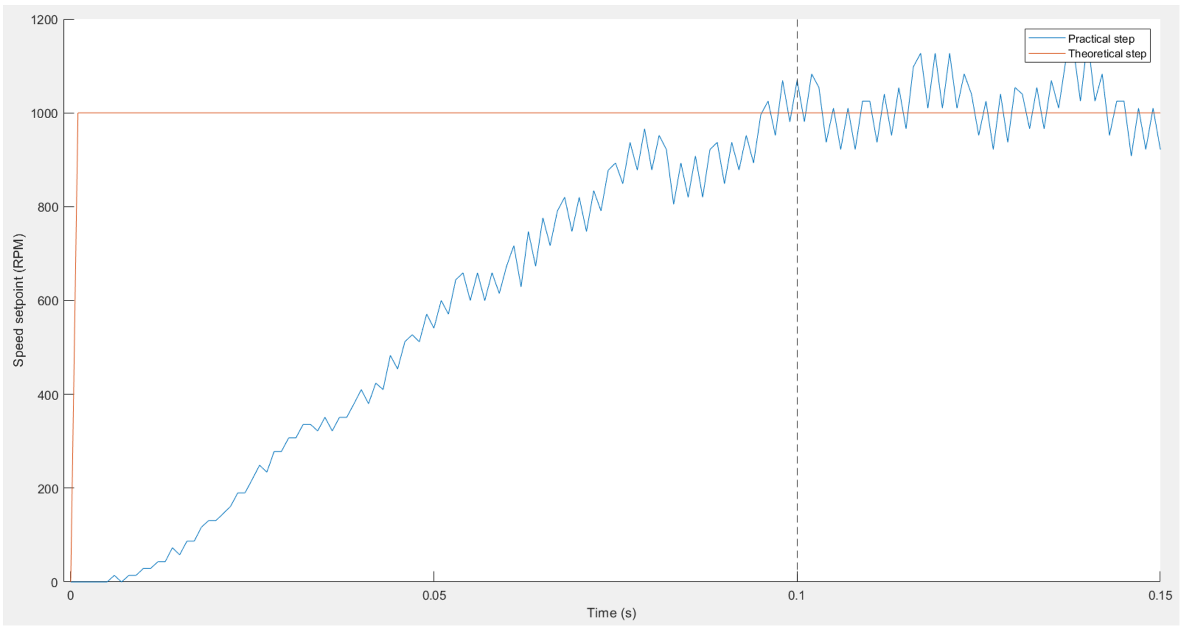


(b) Counterclockwise motion.

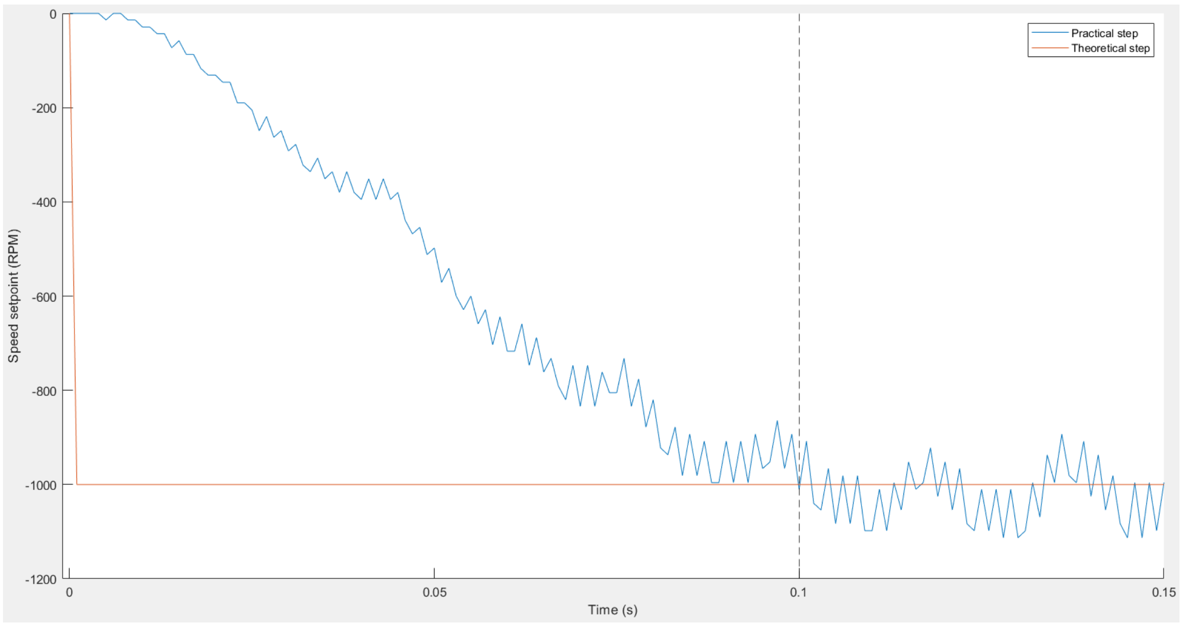
Figure 5.3: Variation of steady state velocity value.

5.2 Closed loop system response

Figure 5.4 shows the step response of the system when placed in closed loop, after a step (speed) of 1000 RPM is applied, for both CW and CCW cases. Attempting on both figures it can be concluded that the time of 0.1s (marked with a vertical dashed black line) is enough for the steady state to be reached. Note that positive values of setpoints represent the CW direction while negative values represent the CCW direction.



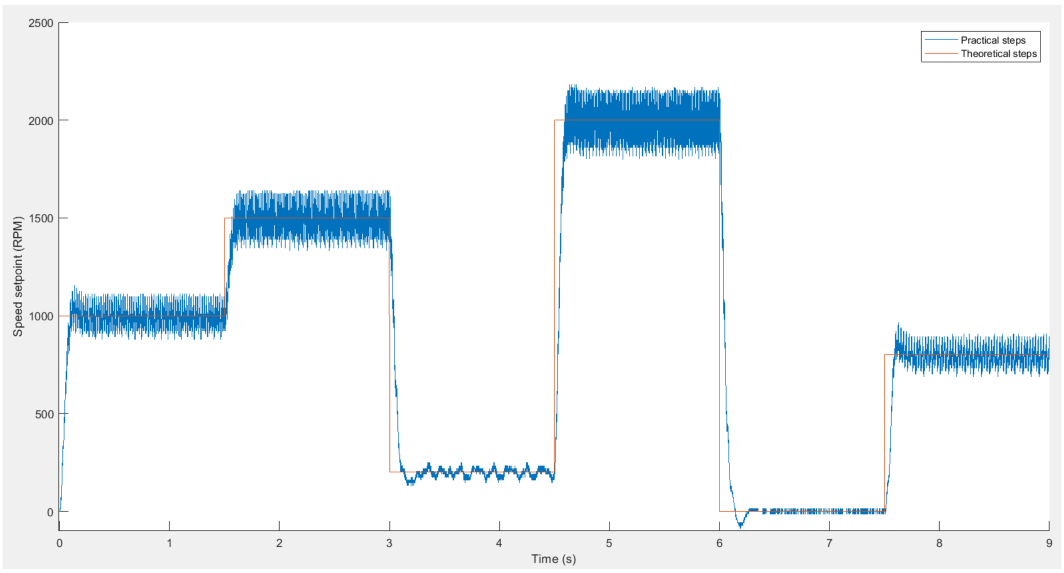
(a) Clockwise motion.



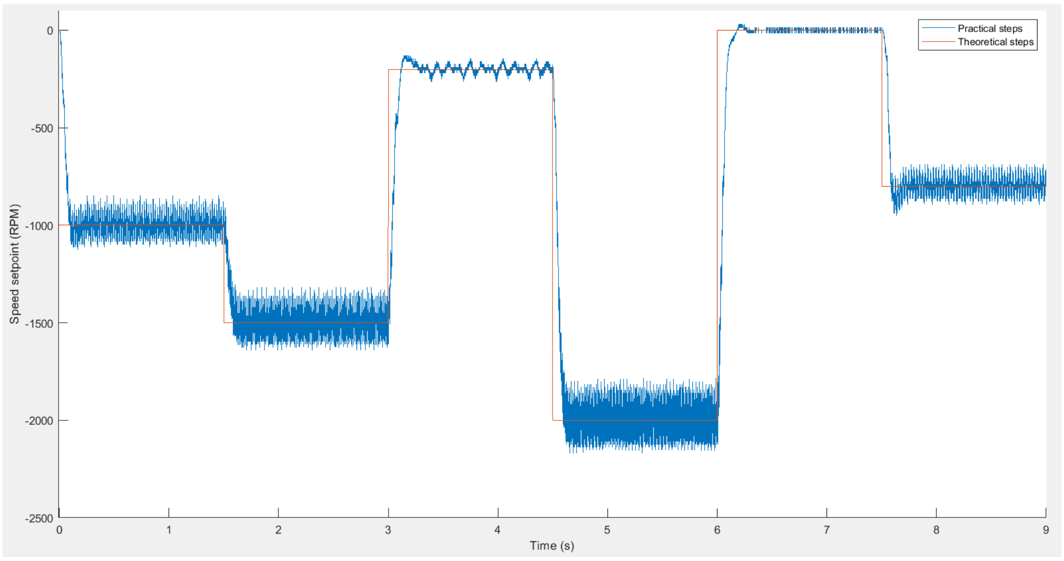
(b) Counterclockwise motion.

Figure 5.4: Closed loop system response when applying the limit setpoints.

To obtain a good closed loop response the PID was tuned with a sampling time of 1ms and the obtained results are presented in figure 5.5 for both the CW and CCW cases. These graphics present the system response for six different steps when the system is in closed loop with the tuned PID. These steps have 1.5s duration and have been chosen in such a way that both smaller and larger transitions were illustrated. This demonstrates the effective behavior of the tuned PID. As verified in the open loop case, the system response continues to be affected by some noise which produces some speed variation around the desired value after the steady state has been reached.



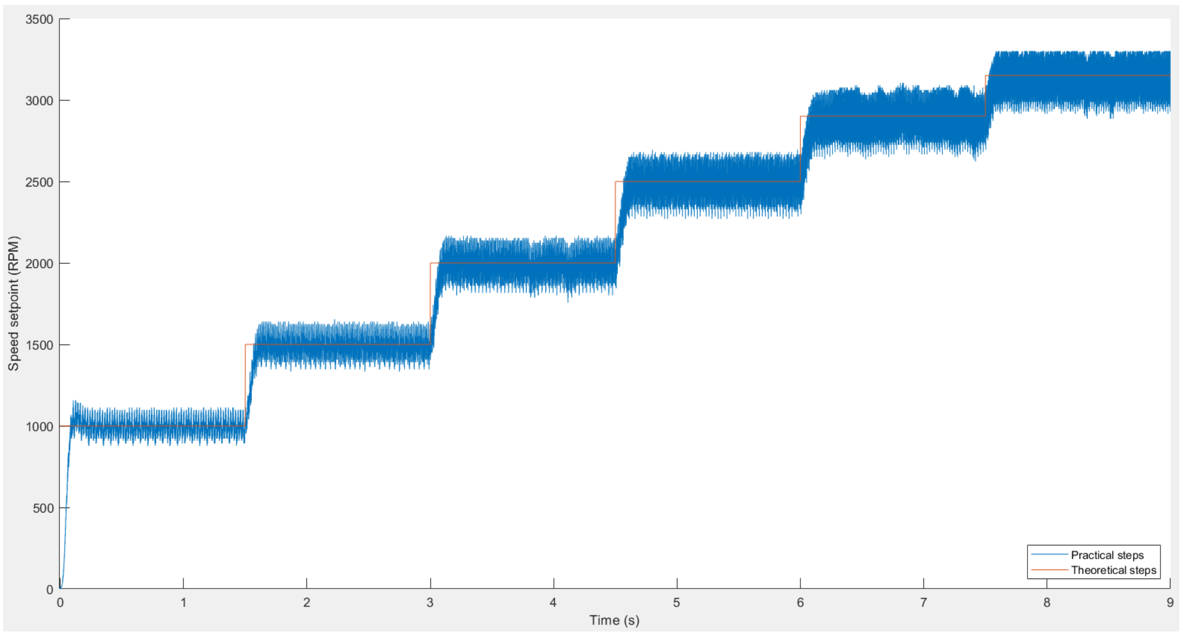
(a) Clockwise motion.



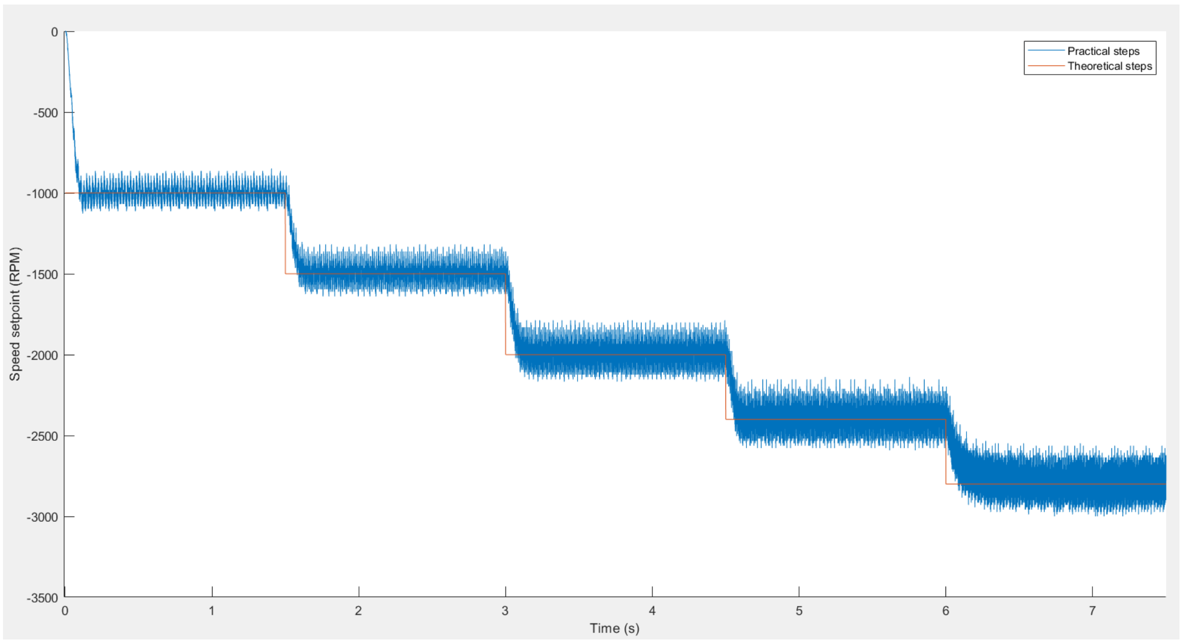
(b) Counterclockwise motion.

Figure 5.5: System behavior for larger and smaller transitions.

With the graphics of figure 5.6, where an increasing sequence of steps is illustrated is now possible to observe that, like in the open loop case, the greater the speed value, the greater the noise around the desired value. With these graphics is also possible to see that like in the open loop case, for the CW motion, the system reaches the 3200 RPM value in steady state. For the CCW case the system achieves the value of 2800 RPM.



(a) Clockwise motion.



(b) Counterclockwise motion.

Figure 5.6: Noise increase with the increase of the setpoint.

5.3 Speed Profile

When implementing the speed profile, some conditions had to be established for the algorithm to work as expected. Those conditions are that half of the total time of movement must be greater than the acceleration time to ensure that the maximum speed value is reached; the acceleration time must be greater than the time for the PID to reach the steady state

(0.1s); all times must be greater than zero. Figure 5.7 shows the obtained results with the implemented speed profile and the corresponding theoretical profile. The characteristics of these profiles are an acceleration time of 1000 ms, a total time of 4000 ms and a maximum speed of 1000 RPM. As it can be seen the obtained results achieve the expected behavior.

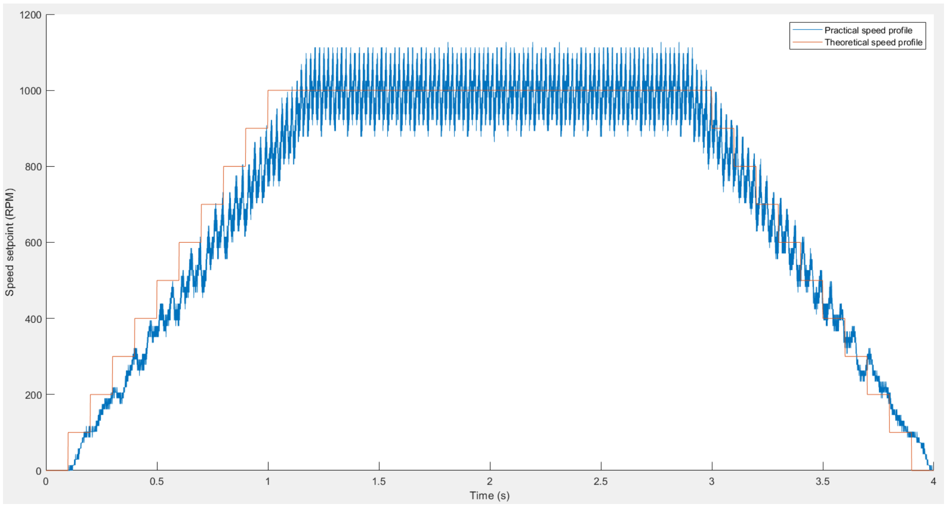


Figure 5.7: Comparison between a theoretical speed profile and an obtained practical speed profile with the same characteristics.

5.4 CAN communication

Figure 5.8 presents a test in which the robotic arm controller was connected to another device via CAN. In this test the other device sent four CAN messages, the first three with non-priority ID and the fourth with a priority ID. Table 5.1 contains the speed profile information sent in each message. The second, third and fourth messages were sent 2, 9 and 13 seconds after the first message, respectively. Figure 5.8 illustrates above the execution of speed profiles in the robotic arm controller and below the value received in another CAN node about a profile being in execution or not. A profile is in execution if the displayed value is 1 and is not in execution if the displayed value is 0. Looking at the figure, one can see that the first speed profile was executed sucessfuly and the second was ignored, since it was received before the end of the first one and the message contained a non-priority ID. In contrast, the third started execution normally as there was no profile being executed when the third message was sent. However, it has only done half of its execution as a new message with priority ID has been received in the middle of its execution. The third profile was then aborted and the fourth profile was executed.

Message number	**1**	**2**	**3**	**4**
Speed (RPM)	1000	1500	1000	1500
Acceleration time (ms)	1000	1000	1000	1000
Total time of movement (ms)	8000	4000	8000	4000

Table 5.1: Speed profile information in the sent CAN messages.

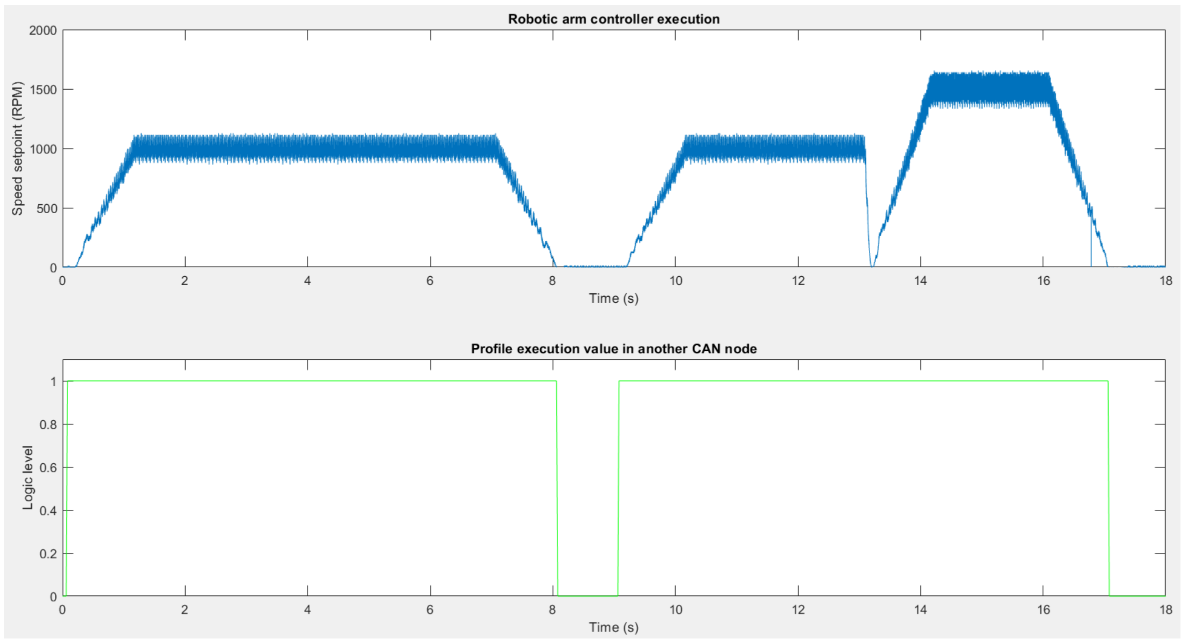


Figure 5.8: Robotic arm controller execution and information reported in another CAN node.

Chapter 6

Conclusions and Future Work

In this chapter conclusions are taken from the developed work and the results that were obtained. Next, some possible improvements are suggested. Finally, it is referred what is left to be done in the future and the challenges that are left to someone that migh continue this work.

6.1 Conclusions

The used BLDC controller presented itself as a possible solution to the problem, since in the end, the speed profile and PID results were satisfactory. However, there was some unexpected noise in the open loop experiences that may be provenient from this controller, which detiorates a little the results quality.

The trapezoidal control that the used BLDC controller implements works fine for duty cycle values of 5% or above however seems to be innefective for lower values. This prevented the motor from making a smooth start for smaller setpoints.

In its general behavior the robotic arm controller is able to communicate via CAN with other CAN module, produce the expected speed profile results and send to the network the required information.

In terms of costs the solution presented for the arm controller is cheaper than the alternatives available in the market which is good.

In terms of objectives met, this work could not meet all the requirements because tests of the controller with a suitable load could not be performed due to some problems that ocurred during the work that prevented the intended motor and gearhead from being used. Consequently, tests with the arm's weight could not be performed, because the alternative motor used did not have a gearhead and the available torque was not enough.

6.2 Future work

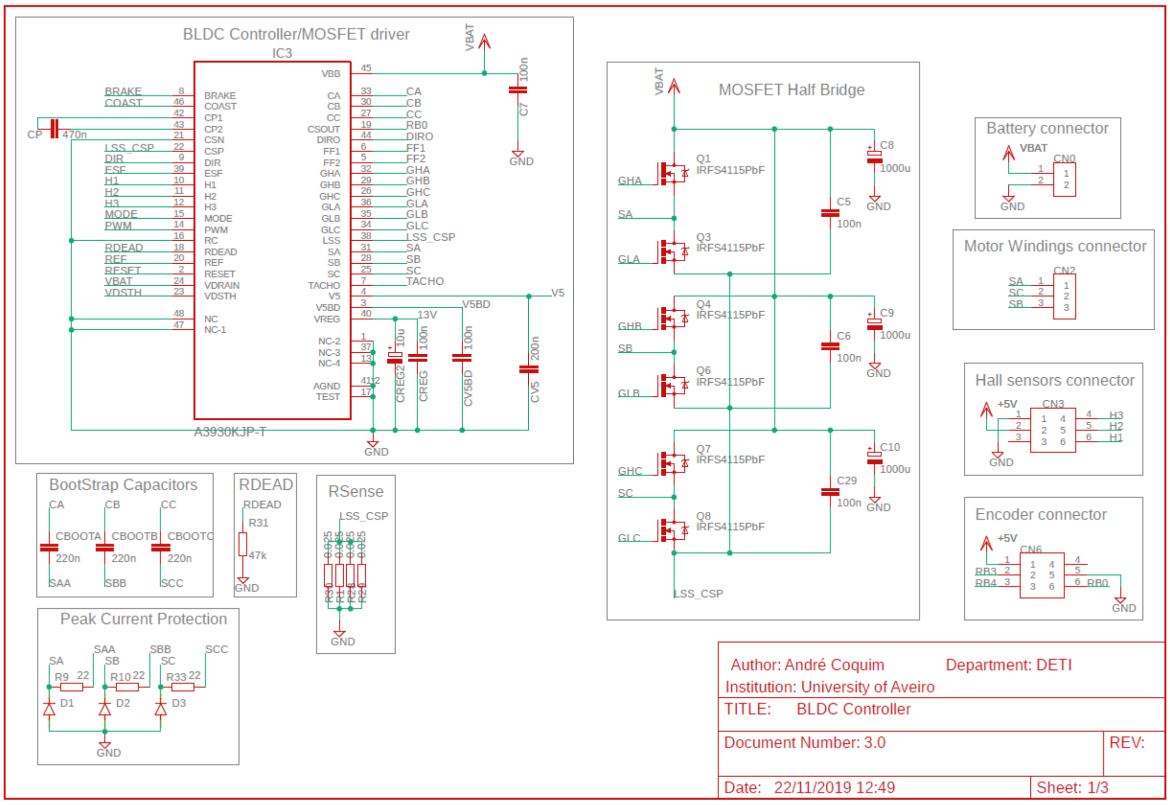
The arm integration and the use of a motor together with a gearhead that can lift it and move it smoothly is the first thing that should follow this work. For that the PID must be tuned again with the new equipment and tests with and without load must be performed. The arm and the developed controller must operate in a suitable way in order to perform the desired movements and so much work is left to be done in the software part.

The speed profile concept shall be extended to the control of all Dynamixel servos of the arm, having each one an independent profile. This way is possible to achieve a certain movement in a desirable time, one joint can move in different speed from the others and all reach their final destination at equivalent times. To do this, an interface electronics with the dynamixel servos needs to be done. Also, communications with these servos must be managed as well.

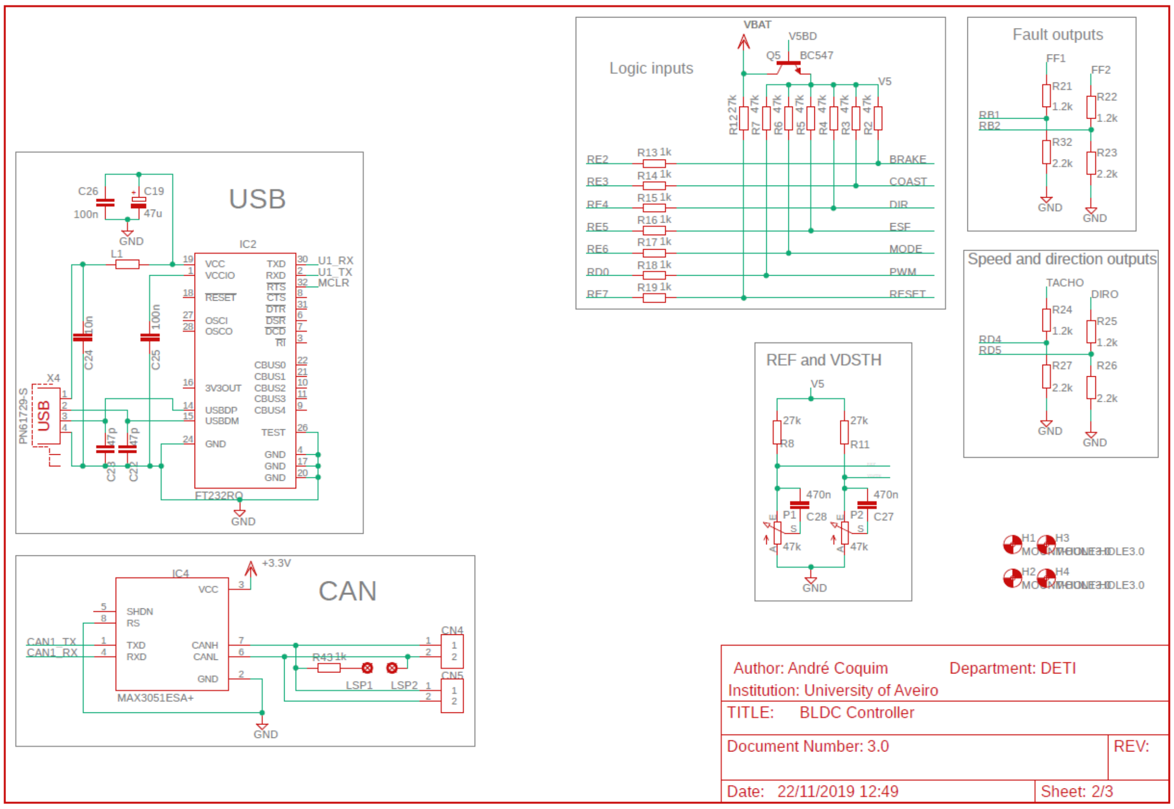
Appendix A

Implementation and prototype

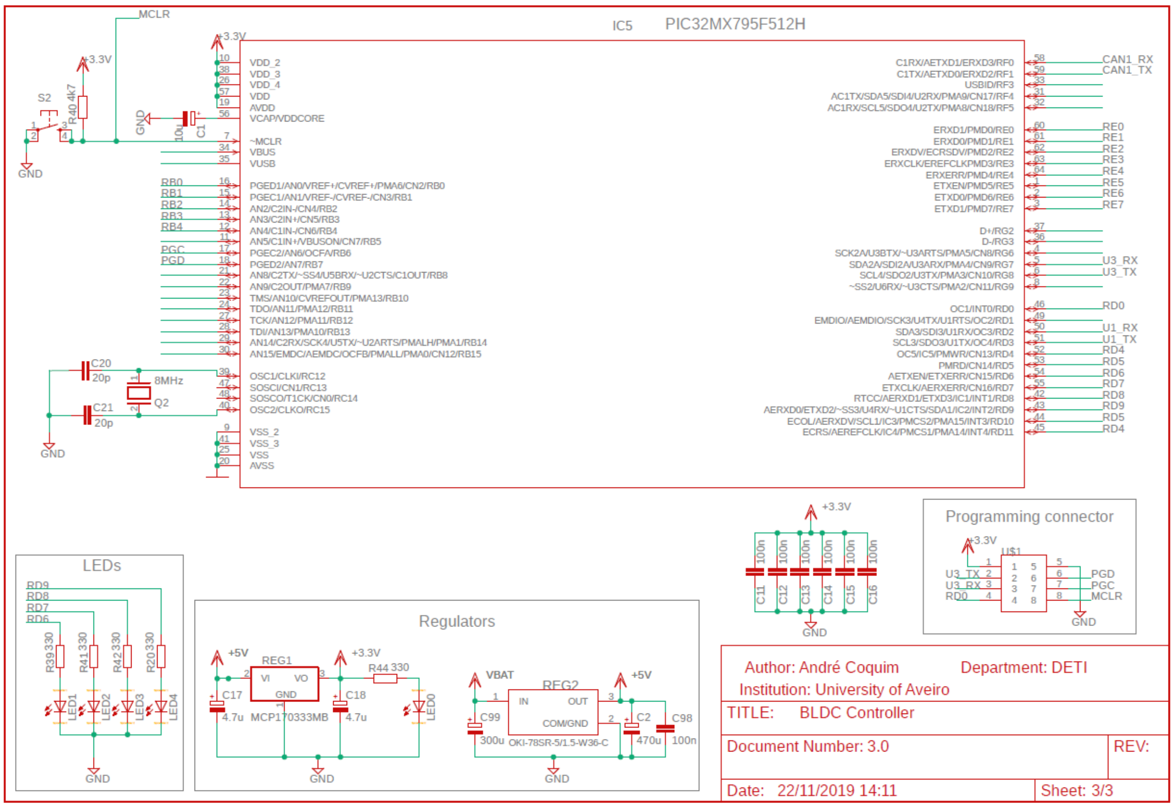
A.1 Robotic arm controller schematic



Author: André Coquim Department: DETI
Institution: University of Aveiro
TITLE: BLDC Controller
Document Number: 3.0 REV:
Date: 22/11/2019 12:49 Sheet: 1/3



USB
CAN
Logic inputs
Fault outputs
Speed and direction outputs
REF and VDSTH
FT232RQ
MAX3051ESA+
BC547
Author: André Coquim
Department: DETI
Institution: University of Aveiro
TITLE: BLDC Controller
Document Number: 3.0
REV:
Date: 22/11/2019 12:49
Sheet: 2/3



IC5 PIC32MX795F512H
LEDs
Regulators
Programming connector
MCP1700333MB
OKI-78SR-5/1.5-W36-C
8MHz
Author: André Coquim
Department: DETI
Institution: University of Aveiro
TITLE: BLDC Controller
Document Number: 3.0
REV:
Date: 22/11/2019 14:11
Sheet: 3/3

Bibliography

[1] H. P. Moravec, "Robot technology." https://www.britannica.com/technology/robot-technology/Robotics-research, November 27 2019. Accessed on 02-12-2019.

[2] "10 powerful examples of ai applications." https://becominghuman.ai/10-powerful-examples-of-ai-applications-553f7f062d9f, March 2014. Accessed on 20-11-2019.

[3] B. D. A. Teixeira, "Desenvolvimento de um braço antropomórfico para um robô de serviço," Master's thesis, University of Aveiro, 2017.

[4] "Different types of motors and their use." https://www.rs-online.com/designspark/different-types-of-motors-and-their-use, May 5 2016. Accessed on 21-07-2019.

[5] "Brushless dc motor vs. ac motor vs. brushed motor?." https://www.orientalmotor.com/brushless-dc-motors-gear-motors/technology/AC-brushless-brushed-motors.html, 2018. Accessed on 21-07-2019.

[6] Learnengi, "Dc motor, how it works ?." https://learnengineering.org/dc-motor-working.html, September 23 2014. Accessed on 01-10-2018.

[7] A. Hughes, *Electric Motors and Drives.* 1990.

[8] J. M. da Silva Pereira, "Sistema de monitorização elétrico para o veículo hammershark," Master's thesis, University of Aveiro, 2013.

[9] L. EITEL, "Motion control tips." https://bit.ly/3437Aja, June 5 2018. Accessed on 01-10-2018.

[10] Learnengi, "Brushless dc motor, how it works ?." https://learnengineering.org/brushless-dc-motor.html, October 17 2014. Accessed on 01-10-2018.

[11] J. Kelly, "What is the most effective way to commutate a bldc motor?." https://www.digikey.com/en/articles/techzone/2017/feb/what-is-the-most-effective-way-to-commutate-a-bldc-motor, February 16 2017. Accessed on 19-04-2019.

[12] D.-K. N. A. Editors, "How to power and control brushless dc motors." https://www.digikey.com/en/articles/techzone/2016/dec/how-to-power-and-control-brushless-dc-motors, December 7 2016. Accessed on 19-04-2019.

[13] T. Agarwal, "Brushless dc motor – advantages, applications & control." https://www.elprocus.com/brushless-dc-motor-advantages-applications-and-control/. Accessed on 09-10-2018.

[14] "Bldc motor control algorithms." https://www.renesas.com/kr/en/solutions/key-technology/motor-control/motor-algorithms/bldc.html. Accessed on 11-10-2018.

[15] B. Aki, M. Bhardwaj, and J. Warrine, "Sensorless trapezoidal control of bldc motors," *Texas Instruments*, 2015.

[16] J. D. F. da Costa, "Scooter elétrica - implementação de um controlador para motores bldc," Master's thesis, University of Aveiro, 2014.

[17] D. Polenov[1], T. Reiter[1], R. Baburske[2], H. Pröbstle[1], and J. Lutz[2], "The influence of turn-off dead time on the reverse-recovery behaviour of synchronous rectifiers in automotive dc/dc-converters," 2009.

[18] A. Tantos, "H-bridge drivers." https://www.modularcircuits.com/blog/articles/h-bridge-secrets/h-bridge_drivers/, 2011. Accessed on 21-11-2019.

[19] M. Douglas B. Leviton, Dunkirk, "Method and apparatus for ultra high-sensitivity, incremental and absolute optical encoding." https://patentimages.storage.googleapis.com/22/37/3d/59538cd8228f10/US5965879.pdf, October 12 U.S. Patent 5 965 879, 1999. Accessed on 30-11-2019.

[20] P. M. Chiau Woon Yeo, "Absolute encoder based on an incremental encoder." https://patentimages.storage.googleapis.com/2d/d1/d1/15cae55e39b25f/US6683543.pdf, January 27 U.S. Patent 6 683 543, 2004. Accessed on 30-11-2019.

[21] C.-T. Chen, *Analog and Digital Control Sytem Design.* 1993.

[22] K. H. Ang, G. Chong, and Y. Li, "Pid control system analysis, design, and technology.," 2007.

[23] R. B. GmbH, "Can specification." http://esd.cs.ucr.edu/webres/can20.pdf, September 1991. Accessed on 23-07-2019.

[24] SteveCorrigan, "Introduction to the controller area network (can)," *Texas Instruments*, August 2002.

[25] L. Allegro MicroSystems, "Automotive 3-phase bldc controller and mosfet driver." https://www.allegromicro.com/~/media/Files/Datasheets/A3930-1-Datasheet.ashx, July 27 2018.