Final Master Thesis

M.Sc.Automatic Control and Robotics

Development of a MATLAB/Simulink - Arduino environment for experimental practices in control engineering teaching

MEMORY

Author:	Eneko Lerma Macaya
Director:	Ramon Costa Castelló
Co-director:	Robert Griñó Cubero
Date:	July, 2019

Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

[This page intentionally left blank]

Abstract

This project presents the steps followed when implementing a platform based on MATLAB/Simulink and Arduino for the restoration of digital control practices. During this project, an Arduino shield has being designed. Along with this, a web page has also been created where all the material done during all this project is available and can be freely used. So anyone interested on doing a project can have a starting point instead of starting a project from scratch, which most of times this results hard to implement.

Taking all this into account, the document is structured in the following manner. The first chapter talks about the hardware used and designed. The second one explains the software used and the configurations done on the laboratory's PCs. After that, the web page Duino-Based Learning is explained, where you can find the five projects carried out in the "Control Automàtic" subject with their corresponding results. In this section too, as an additional research, the implemented indirect adaptive control will be explained, where the parameter estimation has been done by the Recursive Least Square algorithm. The last four sections before presenting the conclusions of the work, correspond to a satisfaction questionnaire done to the teachers that have used the setup, the costs and saves of the project, the environmental impact and the planning of the project respectively.

Contents

Glossary

ADC	Analog to digital converter
DAC	Digital to analog converter
DC	Direct current
GND	Ground signal
Hz	Hertz (frequency measure)
K_{mot}	Motor's static gain
LTI	Linear time invariant
MIMO	Multiple Input Multiple Output
PID	Proportional-integral-derivative controller
RLS	Recursive least square
SISO	Single Input Single Output)
THT	Through Hole Technology
ZOH	Zero order holder
$\mathbf{u(t)}$	Control signal
$\mathbf{p(t)}$	Perturbation signal
$\mathbf{y(t)}$	Process output
$\mathbf{s}$	Variable of the Laplace transform
$\mathbf{z}$	Variable of the Z transform
t_s	Settling time of the closed loop system with the 2% criterion
f_s	Sampling frequency
T_s	Sampling time of the control system
T_g	Sampling time of graphication (analog input block's sampling time)
τ_{mot}	Motors time constant
ξ	Damping ratio
ω_n	Natural frequency

Preface

State of the art

In the field of engineering teaching, the realization of experimental practices continues being of vital importance for the correct formation of the students. Unfortunately, the acquisition of equipment that allows to perform experimental practices presents a high economic cost situation that is not always easy to assume by the teaching centers.

The appearance of low-cost hardware platforms, such as Arduino and Rasberry Pi, has favored the development of practice environments that can be used in almost all areas and in particular in automatic control field [23], [9], [16], [20], [10]. This type of devices can be programmed by their own simulation environments or using automatic code generating environments like those incorporated in MATLAB/Simulink [23], [20]. All this has reduced substantially the costs of data acquisition, generation and processing equipment, as well as the cost in time to prepare some practices.

This thesis presents the developments and analysis that have been made with the objective of replacing the platform that is currently being used in the laboratories of the Automatic Control department (Departament d'Enginyeria de Sistemes, Automàtica i Informàtica Industrial, ESAII) in the School of Industrial Engineering (Escola Tècnica Superior d'Enginyeria Industrial de Barcelona, ETSEIB) from the Technical University of Catalonia (Universitat Politècnica de Catalunya, UPC) by a low cost platform based on Arduino board and MATLAB/Simulink softwares.

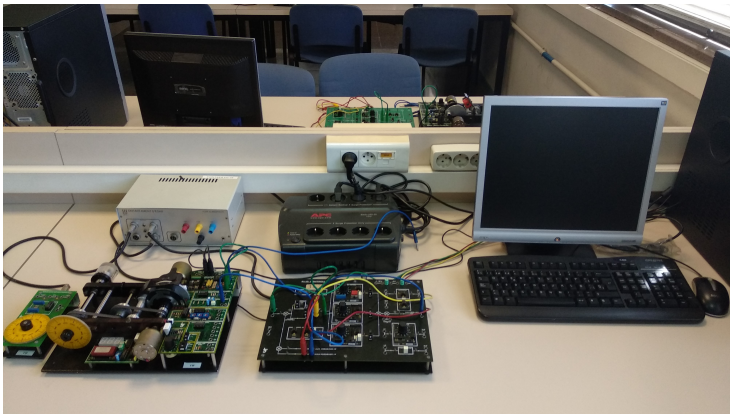
Motivation

The ETSEIB's automatic laboratory is equipped with 12 angular position servosystems (LJ Technical Systems), each of them connected to a PC equipped with AD/DA cards. Supervision and control is carried out through the real-time library for MATLAB (Real-time Widows target). The interface has been developed in the ESAII department (ETSEIB section) and allows the user to specify the sampling period, the parameters for the different controllers and the type of input generated.

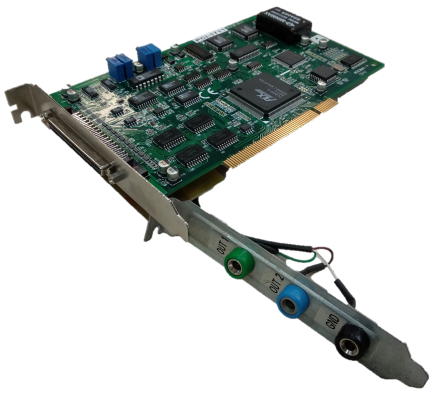
The cost of the input/output cards system is high and it's about devices that use the PCI bus. This is becoming less and less common, what complicates the update of the

equipment. This is one of the main reasons why the project described in this document was started. The main objective is to analyze the viability of replacing systems based on AD/DA cards by systems based on Arduino devices. Figure 1 shows a photo of the ETSEIB's automatic laboratory. One of the workstation and the AD/DA card used before this project began can be analyzed on Figure 2.

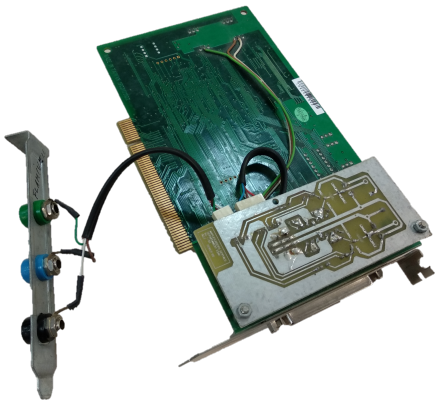
Figure 1: ETSEIB's automatic laboratory.



(a) One of the automatic laboratory's workstation before the start began.



(b) AD/DA card front view.



(c) AD/DA card bottom view.

Figure 2: Setup of the laboratory at the beginning of the project.

Chapter 1

Introduction

1.1 Objectives

Taking into account that experimentation is key in learning and that sometimes teaching centers cannot acquire equipment that allows those experimental practices, the **general objectives** of this project are the following:

- Motivate students to study digital control, using easily accessible hardware.
- Help teaching centers on updating laboratories and educational laboratory projects.
- Provide a starting point for any educator or learner that wants to learn the basics of control engineering through projects.

Focusing those general objectives on our concrete project, we have conclude the **specific objectives** of it are the following:

- Restoration of the devices used in the practices of Automatic Control subject of the ETSEIB by devices based on Arduino hardware with the minimum change of the experiments carried out before starting this research.
- Creation of a web page that introduces the fundamentals of Control Engineering through simulation and walk through videos.
- Implementation of indirect adaptive controllers for velocity and position control for the module used as additional research.

1.2 Methodology and scope of the project

The project has being done following a chronological methodology. First of all, the current set up was analyzed. This meant the LJ Technical System's servo system and the

Arduino Due had to be understood. To do so, their corresponding manual and datasheets had to be read in order to know which their main characteristics are and put ourselves in situation..

Due to incompatibilities in between the Arduino Due board's I/O ranges and the servosystem's I/O ranges, a shield was designed. This meant we had to analyze what transformation the signals had to suffer and, according to that, design a conditioning circuit which at the same time could be embedded on the Arduino as a shield. Once simulating and verifying the correct functioning of our circuit, the components to create the PCB where searched.

In order to validate it, validation tests where done and after that the development of the current practices of the automatic control subject was done, so some students started working on it.

During the semester, while students were using the new setup, a repository and web page were developed, so everyone interested on starting a project could have a starting point. There you can find video tutorials, exercises and the models implemented during the project. Along with the design and development of the web page, we were experimenting with indirect adaptive control.

Finally, a new release of the shield was done, as some new improvements were needed.

In conclusion, this project has correctly covered the initial objectives of the project, having as result a good working shield, one renewed workstation on the automatic control laboratory and a complete webpage with the introduction of the fundamentals of Control Engineering.

Chapter 2

Hardware description

In this section, a brief introduction of the hardware used during the project is done. Figure 2.1 shows all the components and the connection diagram of our complete system.

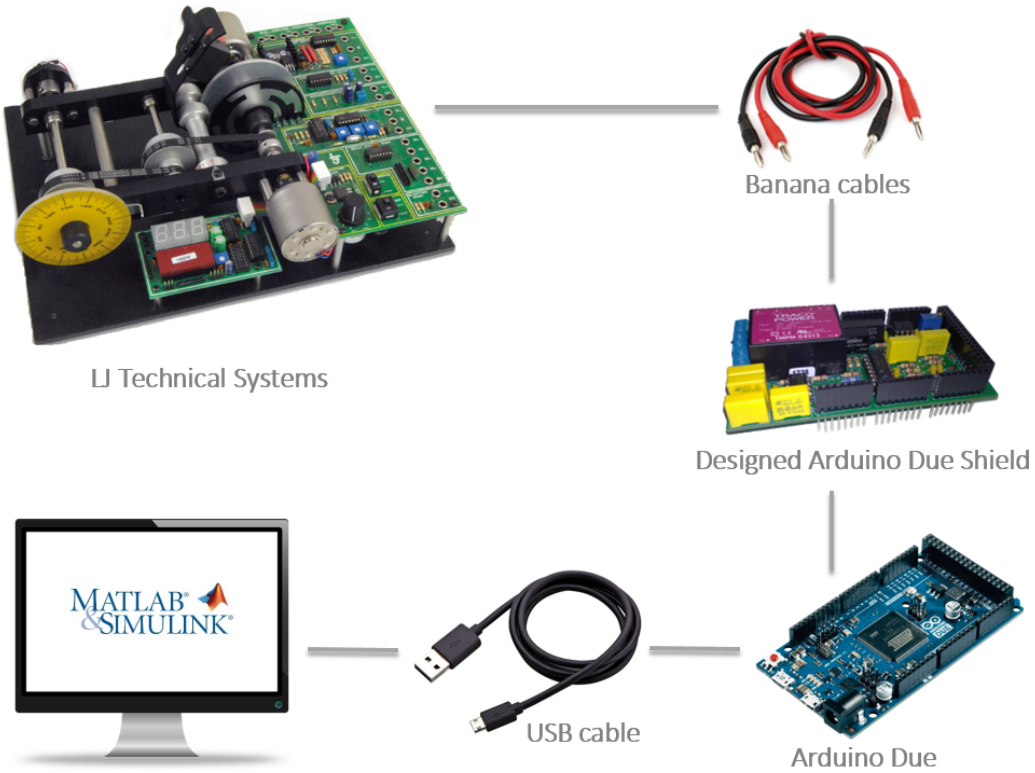


Figure 2.1: Connection diagram of the complete system.

2.1 LJ Technical Systems

The MS15 DC Motor Module enables the user to perform closed-loop, positional or speed control of a d.c. motor. The speed and direction of rotation of the motor can be controlled by either an analog signal or a pulse width modulated (p.w.m) digital signal.

In the other way, speed and position feedback information are available in both analog or digital forms, thus the module can be controlled by either analog or digital system.

The module consists of the following elements: d.c. Motor, tachogenerator, continuous rotation potentiometer, gray-coded disc, slotted disc, digital tachometer, Eddy current brake.

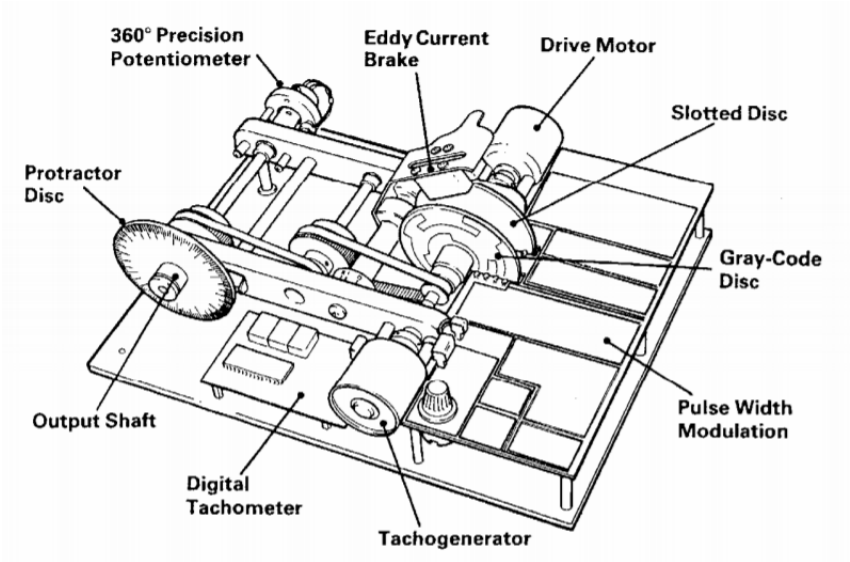


Figure 2.2: MS15 DC Motor Module

In this practice we will only use the tachometer, to close speed control loops and the potentiometer, to close position control loops. In addition to this, we will use the magnetic brake as perturbation.

- **d.c. Motor:** The motor is capable of being driven at speeds of up to 2,500 rpm in either direction. The motor output is geared down by a ratio of 9:1 to drive the output shaft which has a calibrated indicator disc to show angular shaft rotation. Input circuitry is provided to allow the motor to be driven from either the analog input, V_{in} or the digital p.w.m input P_W.

- **Tachogenerator:** A second d.c. motor driven directly by the first motor provides an analog voltage feedback proportional to the speed and direction of rotation. A variable load can be applied to the d.c. motor by switching the **Generator Load** circuit across the tachogenerator output. In this case the tachogenerator output, V_{OUT}, is not available.

- **Continuous Rotation Potentiometer:** This potentiometer is driven by the output shaft and provides an analog output proportional to the angular position of the output shaft. The potentiometer can be disengaged from the output shaft when not in use.

- **Eddy Current Brake:** A 2 position Eddy Current brake is fitted to the module, allowing repeatable mechanism loading to be selected.

More information can be founded in it user's manual [15] or in [12]. Information about the power supply the motor uses is avaliable in [13] and about the banana cables in [14].

2.2 Arduino Due

Arduino is an open-source hardware and software platform. It is based on a board with a microcontroller and a development environment, but it also allows to be programmed from other automatic code generating environment, such as MATLAB/Simulink. Thanks to its design, easy of use and open-source, makes from it a very powerful and versatile board to work on multidisciplinary projects. Along with this, Arduino has a large family of boards, which means you can find different types of boards for the needs of the project you are working on.

For this project, the Arduino Due board has been selected. The Arduino Due is a microcontroller board based on the Atmel SAM3X8E ARM Cortex-M3 CPU. It is the first Arduino board based on a 32-bit ARM core microcontroller. It has 54 digital input/output pins (of which 12 can be used as PWM outputs), 12 analog inputs, 4 UARTs (hardware serial ports), a 84 MHz clock, an USB OTG capable connection, 2 DAC (digital to analog), 2 TWI, a power jack, an SPI header, a JTAG header, a reset button and an erase button. Unlike most Arduino boards, the Arduino Due board runs at 3.3V.

This board was selected due to it high computational performance, due to the fact that it is one of the few boards that has real D/A converters and because it has 12 bits of resolution, most of the Arduino board only have 10 bits of resolution.

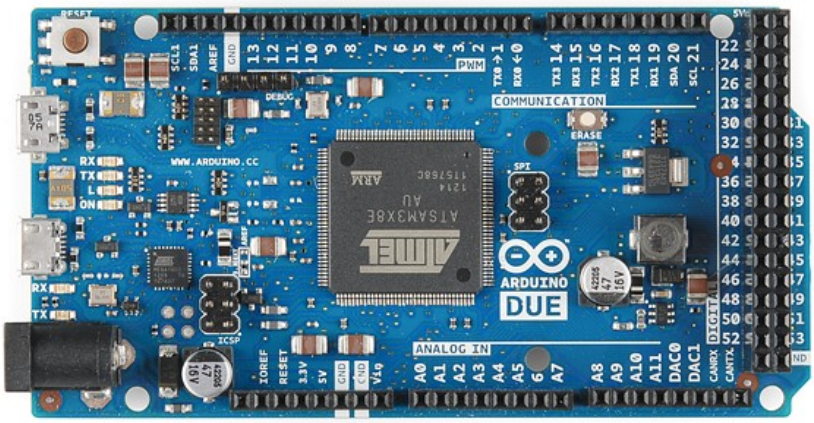


Figure 2.3: Arduino Due board

The technical specifications are shown in Table 2.1.

For more information [2].

Microcontroller	AT91SAM3X8E
Operating Voltage	3.3V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-16V
Digital I/O Pins	54 (of which 12 provide PWM output)
Analog Input Pins	12
Analog Output Pins	2 (DAC)
Total DC Output Current on all I/O lines	130 mA
DC Current for 3.3V Pin	800 mA
DC Current for 5V Pin	800 mA
Flash Memory	512 KB all available for the user applications
SRAM	96 KB (two banks: 64KB and 32KB)
Clock Speed	84 MHz
Length	101.52 mm
Width	53.3 mm
Weight	36 g

Table 2.1: Arduino Due's technical specifications

2.3 Designed Signal Adapter Shield

At the beginning of this project, we realized that the input and output ranges of the Arduino did not coincide with the I/O ranges of our plant (LJ Technical Systems), so we had to design an intermediate signal conditioning stage, which was decided to be connected to the Arduino as a shield. In this section how the shield has been designed will be introduced, for this, the following values must be taken into account:

- I/O ranges of the LJ Technical Systems: $[-5, 5]V$
- Output range of the Arduino Due's DAC: $[0.6, 2.7]V$
- Arduino Due's analog input range: $[0, 3.3]V$

Theoretically, the output range of the DAC of the Arduino Due is $[0, 3.3]V$ but if we analyze it by an oscilloscope, we can see this range does not meet with the reality. As we can see on Figure 2.4, in our case, the output range of the DAC is between $[0.58, 2.83]V$.

Searching this issue, we can find that the range of the Due boards varies between $[0.55, 2.75]V$, but every DAC is singular and this limits can vary a little between different Due boards. In order to design a shield that can be used by every Arduino Due, the selected range for the DAC is $[0.6, 2.7]V$.As we can see, the range has been slightly reduced, so we can make sure every DAC will be able to give those minimum and maximum voltages.

In order to make the change of the ranges, the signals must suffer a change in gain and an offset compensation both in the measurement channels (A/D conversion) and in

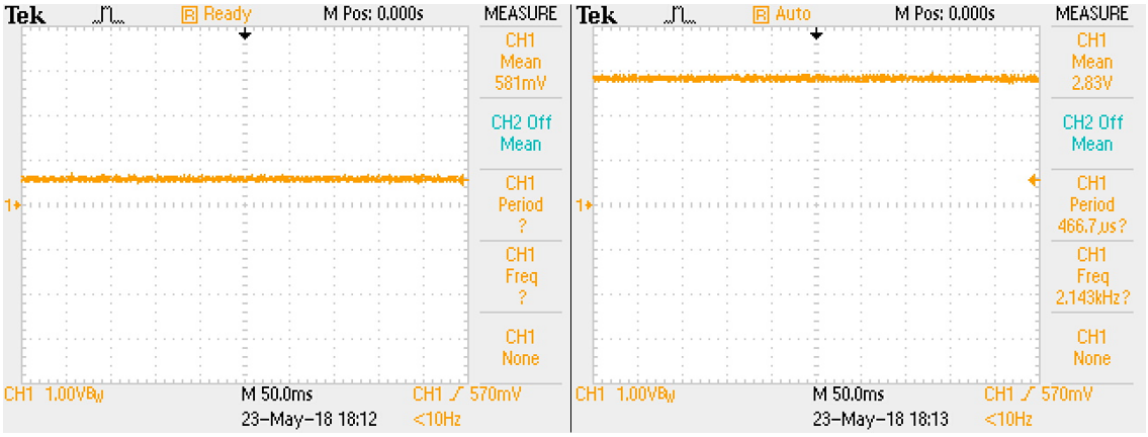


Figure 2.4: Minimun and maximum voltages of the Arduino Due DAC

the control signal channel (D/A conversion). These two actions are performed separately (in cascade) to reduce the coupling between the two actions and facilitate the analysis by the students. The offset has been calculated from the selected DAC's range's mean value, so the same offset can be used at the output signal or at the input one. By that range, the offset is obtained by the following formula:

$$\text{offset} = 0.6 + \frac{2.7 - 0.6}{2} = 1.65V \tag{2.1}$$

In the shield, three different circuits have been designed, one for the conversion of the control signal (output channel), two for the potentiometer and tachometer signals (input channels) and another one for the offset creation. In the following lines, these three circuits will be explained.

2.3.1 Output Channels

The input signal to the plant is bipolar, while the signal of the Arduino Due's D/A converter is unipolar. Therefore, in order to condition it, we first subtract the offset and then amplify its amplitude. To displace the signal through an operational amplifier, a unit gain circuit with a voltage of $1.65V$ at the inverting input is used, colored in blue in Figure 2.5. The amplitude of the signal is changed in the second stage, colored in orange.

As shown in Figure 2.5, in the second stage there are two capacitors that allow low-pass filtering (purple), with a cut-off frequency of $300Hz$, set in the design, this cut-off frequency is calculated as:

$$f_c(Hz) = \frac{1}{2\pi RC} \tag{2.2}$$

In order to reduce the variety of values in the resistances, we have chosen some resistances to $R = 20k\Omega$ and calculate the remaining.

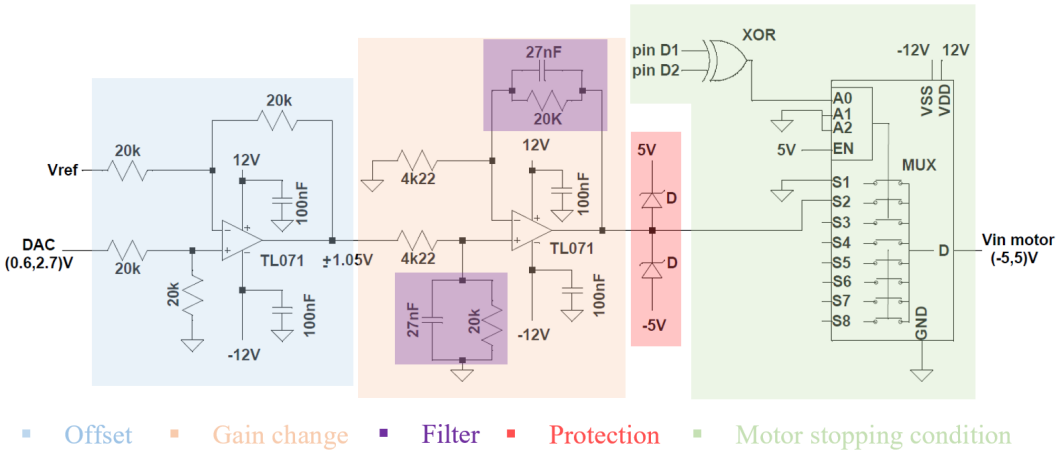


Figure 2.5: Circuit of the signal conditioner for the output of the D/A converter.

The temporal and frequency responses of the circuit on Figure 2.5 simulated on LT Spice, can be seen on Figures 2.6 and 2.7 respectively. This last diagram shows the evolution of the circuit's module and argument at different frequencies on a double scale diagram.

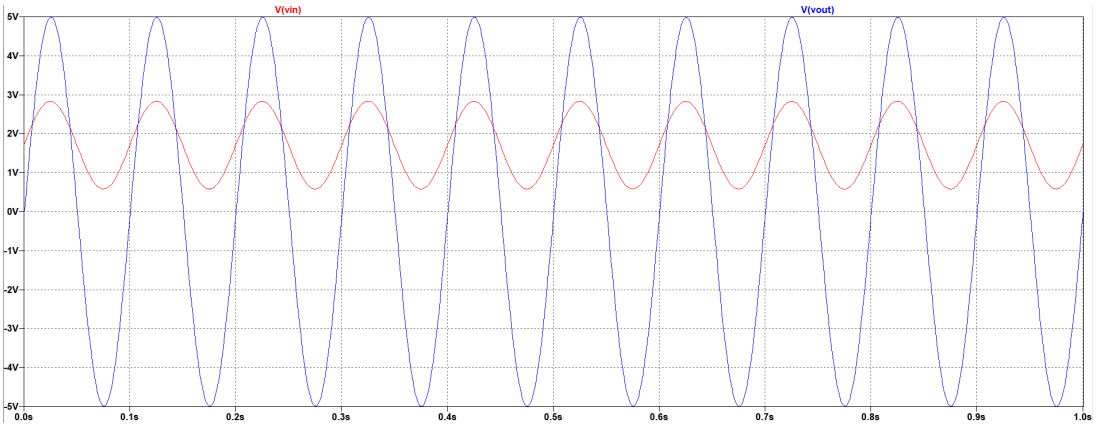


Figure 2.6: Temporal response of the D/A converter circuit simulated on LT Spice. Input signal in red. Output signal in blue

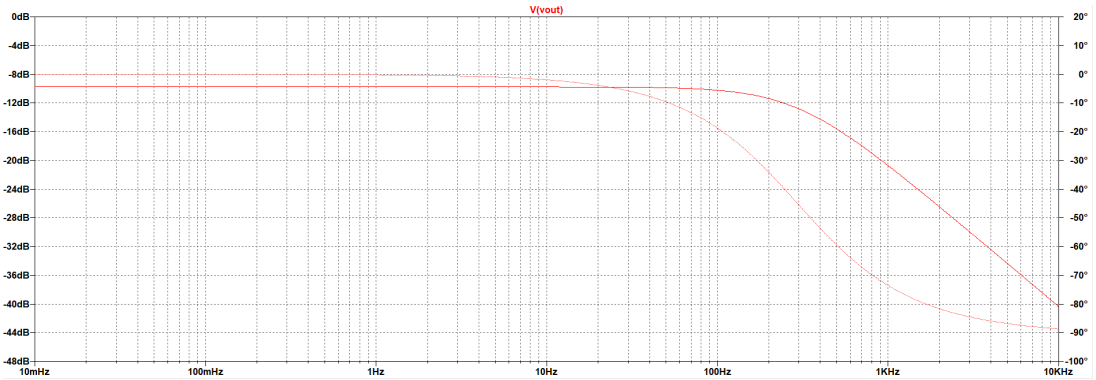


Figure 2.7: Frequency response of the D/A converter circuit simulated on LT Spice on a double scale diagram.

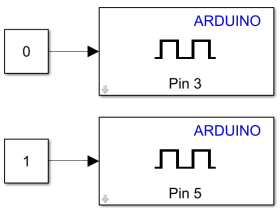
As we can be seen on Figure 2.5, after the output of the the last the operational amplifier, we have an XOR and an analog multiplexer, colored in green. These last two chips are the responsible for keeping the motor stopped. This circuit acts as an automatic switch on the input of the engine. The problem before embedding this circuit was that whenever we where transferring all the code to the Arduino board, we where not able to configure the DAC's output to be in a constant value (in order to have $0V$ in the input of the motor), so the engine kept spinning until the board started executing the code. So, when we tried to control the engine, it did not start from a steady state.

As we were not able to control none of the pins in the Arduino, at first we thought to add an external hardware which controls the motor's input voltage, but after some attempts, we realized the digital pins took random values while executing the code to the board, but those values where the same and at the same time in all digital pins. Taking two digital pins (in this case digital pins 3 and 5) and a two input XOR, we were able to give the input signal to the MUX, which by default is connected to GND. This last step works in the following way: on one hand, if both digital signals are equal, the XOR will give a 0, and the output of the MUX will be connected to the ground, so while the code is being transferred to the Arduino, the motor will be stopped. On the other hand, if both digital signals are different (this condition, which will be shown afterwards, has been codified in Simulink), the output of the XOR will give a 1, and the MUX will switch to the channel connected to the output of the signal conditioning circuit, allowing the control of the motor.

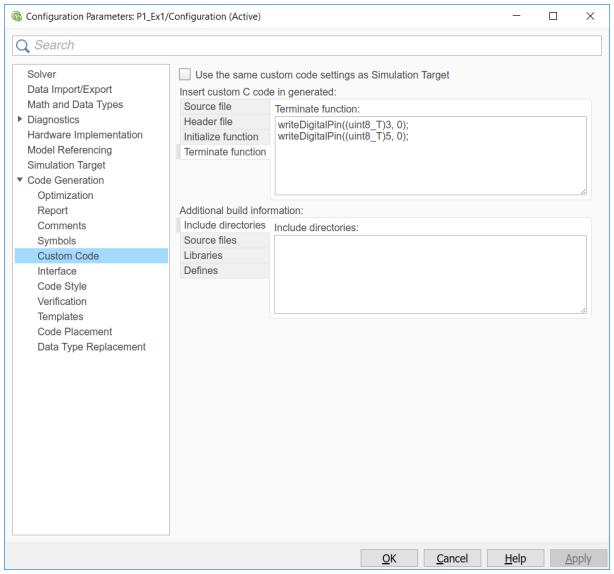
Simulink configuration for the motor stopping condition

In order to configure the Simulink sheet two conditions have to be taken into account. As we have seen, while transferring the code, as the digital outputs take the same output voltage, the XOR will give a 0 and the output of the MUX will be connected to ground, so the motor will be stopped. That's why, we have to configure the condition to switch the MUX on when the code is already running on the board and again to switch it to the ground when the simulation has been finished. As told before, the switching of the MUX is done from the XOR, so while the Simulink model is running we have to output different voltages on the digital output pins connected to the XOR input pins, which is done by the `Digital Output block`, see Figure 2.8a.

Finally, to make the motor stop again when the simulation has been finished, we have to modify the terminate function of the model. For that, we go on `Model Configuration Parameter` > `Code Generation` > `Custom Code` > `Terminate Function`, and we make the digital output pins take the same output voltage, in this case we make both pins take $0V$, see Figure 2.8b.



(a) Condition to move the motor while the Simulink model is running.



(b) Condition to stop the motor when the Simulink model has finished.

Figure 2.8: Simulink configuration for the motor stopping condition.

2.3.2 Input Channels

In the case of the circuit for the the feedback signals (input to the Arduino), the signal changes form bipolar to unipolar, so the first stage makes the gain reduction (colored in orange on Figure 2.9), and the second stage makes the correction of the offset, in blue. As in the previous case, the range of the plant's output signals is $\pm 5V$, but now the analog input pines of the Arduino have a range of 0 to $3.3V$. With this, the values of the resistance have been recalculated so that the input to the second stage of the circuit has a range of $\pm 1.65V$ (to be able to make the same change of offset and obtain the desired range $[0, 3.3]V$ in the analog input pins of the control board). In the second stage, the offset voltage will be added to convert the bipolar signal of amplitude $1.65V$ and centered at $0V$ into an unipolar centered at $1.65V$.

In this case too, the circuit has been equipped with a low pass filter (purple) with the same cut-off frequency as the filter before. As it has been done previously, to avoid resistances with different values, some of them have been fixed to $R = 20k\Omega$ and the others have been calculated.

The simulation of the temporal and the frequency responses of the previous circuit can be seen on Figures 2.10 and 2.11 respectively, obtained in LT Spice software.

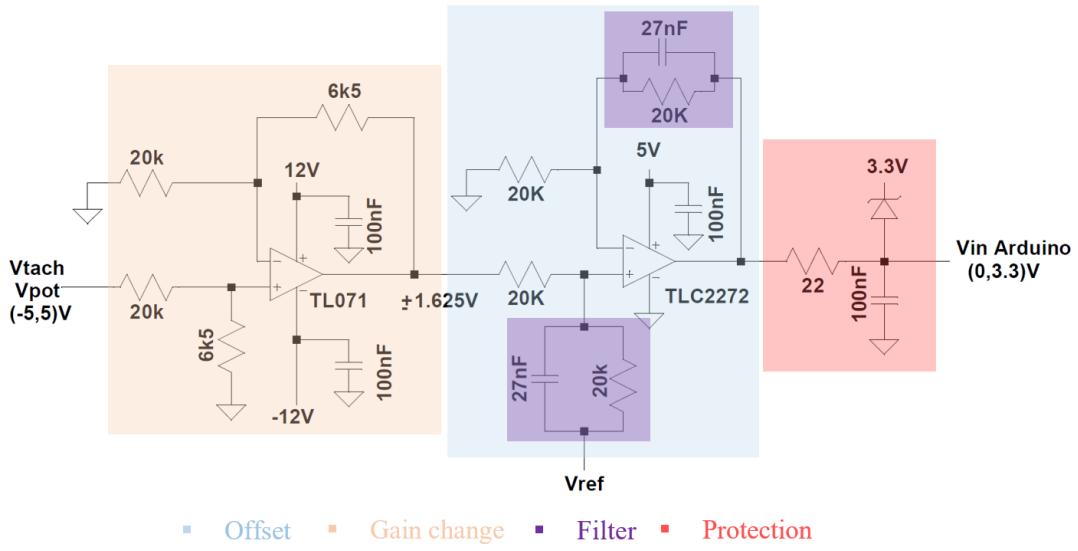


Figure 2.9: Circuit of the signal conditioner for the input of the A/D converter.

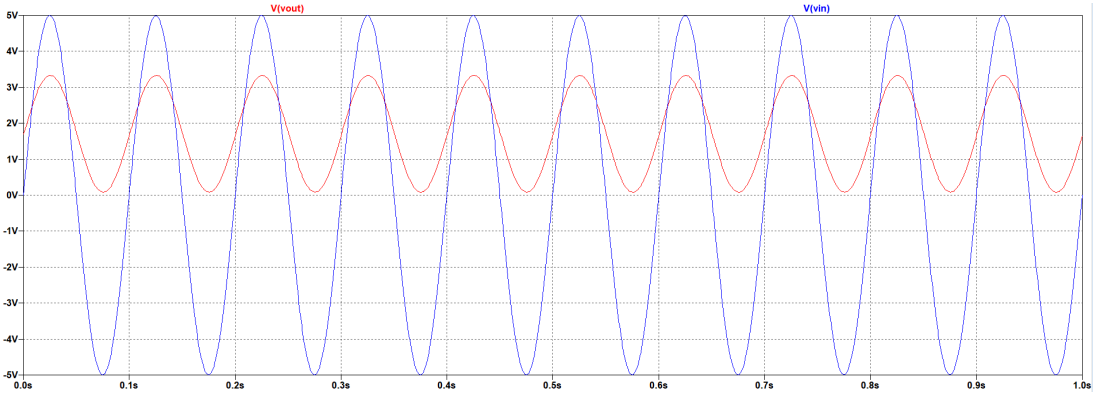


Figure 2.10: Temporal response of the A/D converter circuit simulated on LT Spice. Input signal in blue. Output signal in red

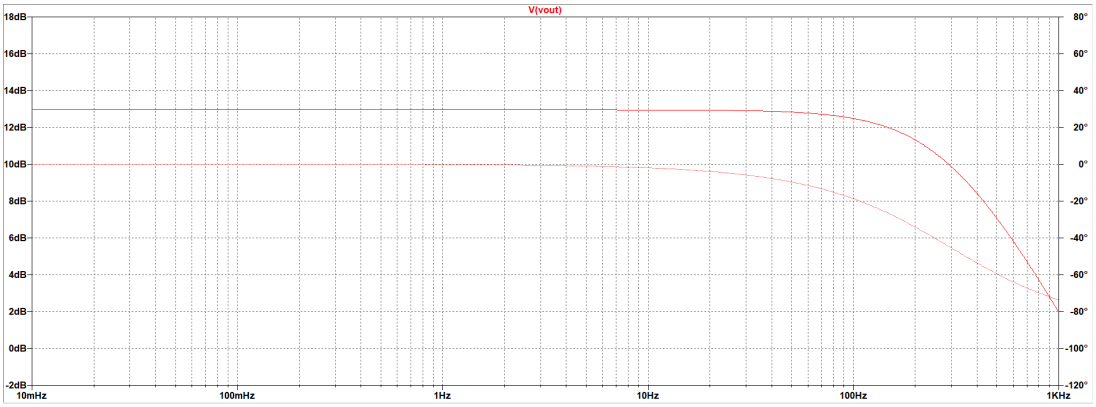


Figure 2.11: Frequency response of the A/D converter circuit simulated on LT Spice on a double scale diagram.

2.3.3 Generation of the reference voltage

The reference voltage used in the stages of offset correction is obtained from the voltage reference LM385Z in series with a voltage follower to minimize the effects of load on the reference device, see Figure 2.12.

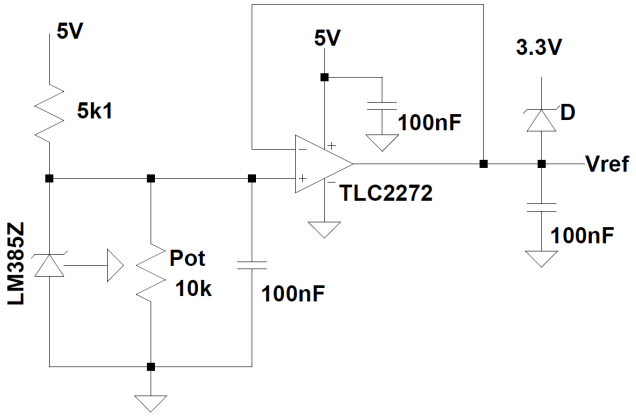


Figure 2.12: Circuit to obtain the voltage reference.

The final result of the board can be analyzed in Figure 2.13. Where the three circuits have been embedded in it and it fits into the Arduino Due as a shield.

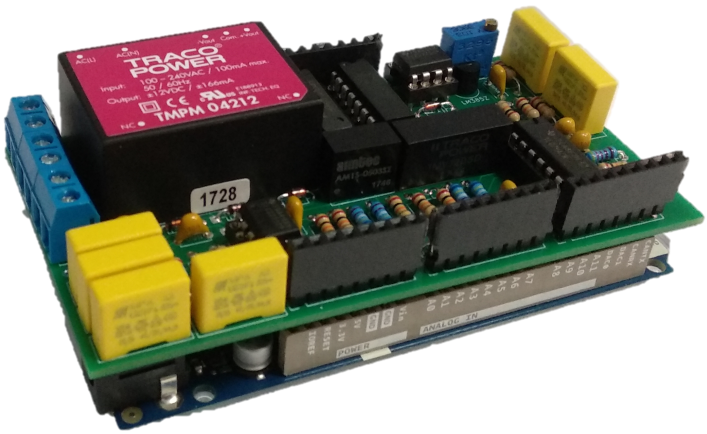
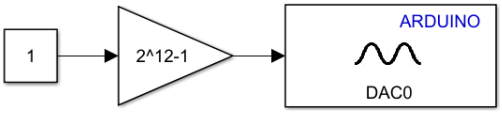


Figure 2.13: Designed signal adapter for the LJ Technical Systems plant embedded in the Arduino Due board.

Once having the shield, we have to adapt the Simulink's analog output and input in order work with voltages on it. For the characterization of the devices, two Simulink models have being designed `OutputConditioning.slx` and `InputConditioning.slx` which block diagram can be seen on 2.14.

In the first case, we apply an input voltage, giving constant values from 0 to 1 and we measure the voltage in the engine's input terminal and obtain the linear function that relates the Simulink constant value and the input voltage to the motor.



(a) Output conditioning model.

(b) Input conditioning model.

Figure 2.14: Input and output block conditioning models.

```
AppliedStep=[0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1];
VinMotor=[-5.241 -4.229 -3.177 -2.164 -1.113 -0.103 0.947 1.995 3 4.056
    5.069];

P1=polyfit(VinMotor,AppliedStep,1)
```

With which we obtain the following equation:

$$bits = 0.0968 * voltage + 0.5085 \tag{2.3}$$

So, if we work with voltage values on the Simulink model, applying this formula, we can know which is the equivalent bit number we have to apply.

In the second case, we apply different voltages in the motor, so it keeps turning in different velocities and we measure the voltage in the terminal of the tachometer. We will obtain again a linear relationship between the voltage in the terminal and the bits in the scope of Simulink. It is worth to comment that just doing this either in the tachometer's terminal or in the potentiometer's terminal is the same, as both signals suffer the same offset and changes.

```
Vtaco=[-4.81 -3.998 -2.947 -1.844 -0.767 0 0.786 1.84 2.945 4.049 4.639];
Bits=[90 430 865 1320 1770 2090 2415 2857 3309 3765 4010];

P2=polyfit(Bits,Vtaco,1)
```

With which we obtain the following equation:

$$Voltage = 0.0024 * bits - 5.0322 \tag{2.4}$$

So, if we apply this formula on the analog input block, we will convert the signal into virtual voltage signals.

Those two equations are coded on the following Simulink A/D and D/A signal conditioner blocks. The first one on the D/A signal conditioner stage and the second one on the A/D signal conditioner stage.

2.3.4 Components

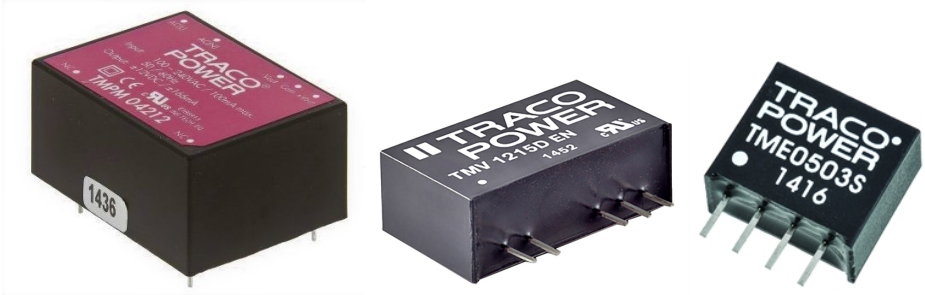
In this section, the component selected for the PCB creation are introduced. These components have great importance when creating the board, as they will be the responsible of the correct functioning of it. For that, it is necessary to check the characteristics of the elements and compare them with the other options available in the market.

Taking into account that these will be handled by the laboratory technicians, in order to be easier to manipulate, it has been decided to choose THT components.

Power supply

As we can see, there are some components on the board that have to be fed and in order to be an autonomous shied, we decided to use some embedded power supplies. We decided to select TRACO power supplies as they offer a large range of fully encapsulated and isolated power modules. For this shield, three different tracos have been chosen: `TMPM04212`, `TMV1205D` and `TME0503S`, see Figure 2.15.

By the first one, we are able to obtain $\pm 12V$ from the plug voltage [30], from the second one, we obtain $5V$ from the $12V$ obtained from the TRACO before [31]. Finally, by the last one, we obtain $3.3V$ giving as input $5V$ [29].



(a) TMPM 04212. (b) TMV 1205D. (c) TME 0503S.

Figure 2.15: TRACO power supplies used to feed the board.

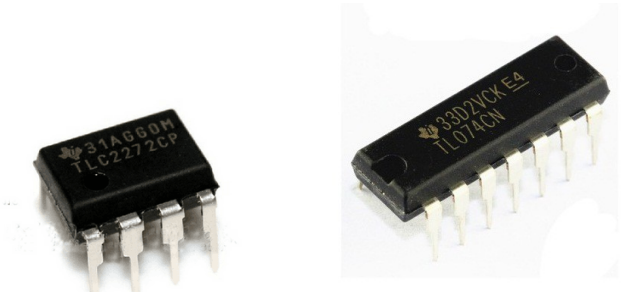
TLC2272 and TL074

The TLC2272 is a dual operational amplifier [28], while TL074 is a quadruple operational amplifier [27], see Figure 2.16.

We have chosen the TL074 as it has the same characteristics as the TL071, but with 4 opamps. As, for the circuit creation we need four TL071, we reduce the number of chips

on the board by selecting the TL074.

The main issue we have to take into account is the power supply, the TL074 is connected to $\pm 12V$, while the TLC2272 is connected to $[0-5]V$. As we can see, this last pin's output is not a bipolar signal, that's why, we don't need to have negative power supply on it.



(a) TLC2272. (b) TL074.

Figure 2.16: Operational amplifiers used to feed the board.

LM385Z

The LM385Z is a 3-terminal adjustable band-gap voltage reference diode, see Figure 2.17. Operating from 1.24 to 5.3V. On-chip trimming is used to provide tight voltage [26]. By the additional voltage reference, good precision and low noise voltage reference is obtained.

Figure 2.17: LM385Z TO92 package used on the board.

Potentiometer

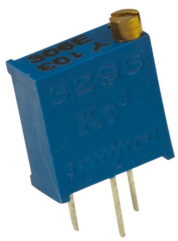


Figure 2.18: T910Y 10k potentiometer.

In addition to the resistors included in the board, there is a potentiometer with which the value of the system's offset can be adjusted. This offset adjustment can be varied by changing the position of the potentiometer's screw, and therefore the value of the resistance. This would be the process to obtain the desired offset of the system, it is better not to modify the position of the potentiometer once the desired offset has been obtained [25]. This potentiometer makes a more versatile board, as changing the offset can be used to adapt different I/O range processes.

Diodes

The 1N4148 switching diodes, are fast switching speed diodes with which we avoid to have greater voltages than $3.3V$ value in the input pin of the A/D converter of uC and so, avoid damaging the board in the circumstances of unexpected operations [4].

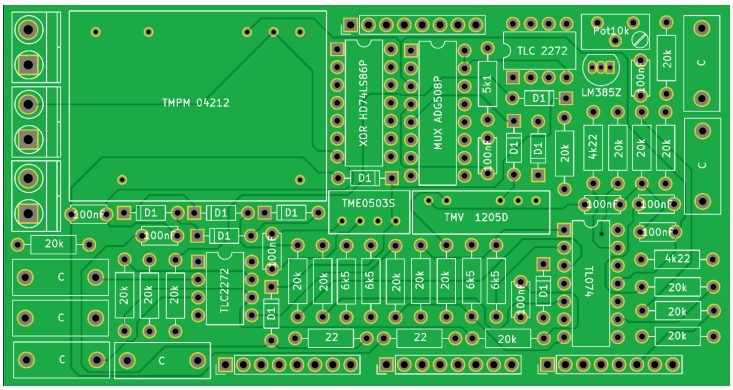
XOR

A four XOR embedded chip has being used, HD74LS86P XOR [24]. One of them is connected to two digital inputs and the output of it is connected to the A0 logic control input of the multiplexer. By means of the output signal of the XOR, we can switch the MUX between the first and second inputs.

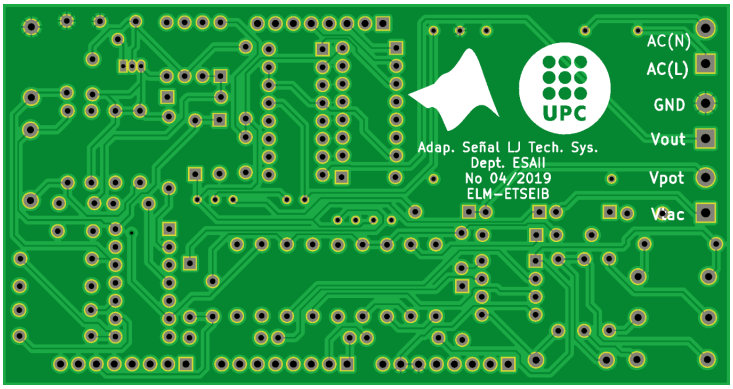
Analog multiplexer

The analog multiplexer switches one of eight inputs to a common output as determined by the 3-bit binary address lines A0, A1, and A2 [1]. In or case, by default the common output will be connected to the first input, which will be connected to ground, and in the case we want the motor to move, we will switch to the second input, which will be connected to the output of the D/A's signal conditioner circuit, see Figure 2.5.

Figure 2.19 shows the designed board's prototype, where all the components commented above are embedded. The design of the board has been done in KiCad software, where the TRACO power supplies had do be designed taking the dimensions from their datasheets.



(a) Front view of the PCB.



(b) Bottom view of the PCB.

Figure 2.19: Front and bottom views of the Signal Adapter board.

Chapter 3

Software description

All the codes and models of this project have been codified in MATLAB/Simulink, because of this reason, this chapter gives the basics of the programs used and how we have to configure them in order to be able to connect and interact with an Arduino board.

3.1 MATLAB

MATLAB platform, is a matrix-based language which integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation [17]. MATLAB includes:

- Maths and computation
- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including Graphical User Interface building

3.2 Simulink

Simulink is a platform that works through a visual programming environment, the functions are represented by blocks, which makes it very easy to use. When executing a model implemented in simulink, a C code is generated that the computer recognizes and executes. Using simulink you can build and simulate models of physical systems and control systems using block diagrams. The behavior of these systems is defined by

transfer functions, mathematical operations, MATLAB elements and predefined signals of all kinds.

Simulink gives the possibility of incorporating MATLAB's own algorithms, MATLAB functions. Simulink offers the possibility of connecting the model with hardware to verify in real time and in a physical way how it works [18].

3.2.1 Configuration

Before start working with Arduino and Simulink, we have to know that some toolboxes have to be installed. This toolboxes are the responsible for enabling the data transference from the computer to the motor and vice-versa. The installed packages are: `MATLAB and Simulink Support Packages for Arduino Hardware` in order to acquire inputs and send outputs on Arduino Boards, see Figure 3.1, and `MATLAB Coder, Simulink Coder and Embedded Coder` for the generation of automatic C and C++ code for embedded systems. This packages can be easily found in MATLAB's home tab, on adds-on section, see Figure 3.2.

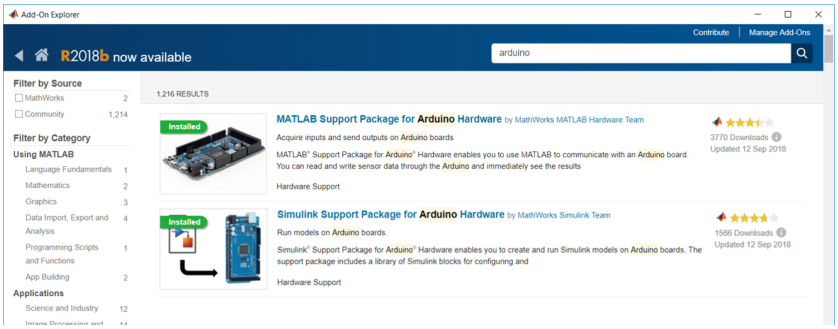


Figure 3.1: MATLAB and Simulink Support Packages for Arduino Hardware.

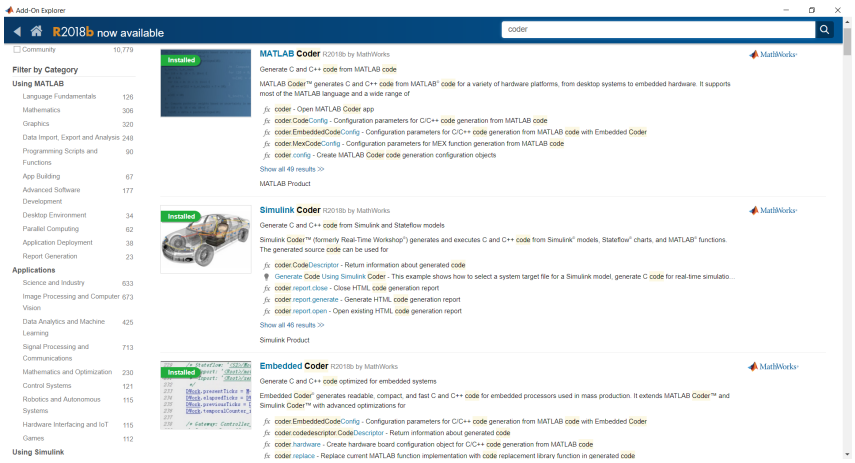


Figure 3.2: MATLAB Coder, Simulink Coder and Embedded Coder packages.

After their installation, we have to differ the two main methods that Simulink has to work for the communication between the board and the PC: `Normal Mode` and `External`

`Mode`. Both methods have their pros and cons, and, depending on the project, one method may prove to be more beneficial than the other.

- Normal Mode with Simulink IO:

In order to connect to an external hardware such as Arduino and download programs onto it, you can use Simulink I/O. It is a fast mode to execute your code and communicate with the I/O peripherials, but the main disadvantage of this connection mode is that data recording is more difficult than with External Mode.

- External Mode:

By the External Mode you have the ability to tune parameters and monitor data in real-time, without having to re-download or re-compile the model each time a change is done. Once External mode is started, it uses the serial port to communicate between the Arduino and the computer. Therefore, to run in External mode, the USB should be connected between them. This ability to adjust and monitor changes in real-time is what makes External Mode such an interesting option for running these experiments. You can benefit from being able to see the immediate effects your changes have on the hardware. However, this ability to make changes in real-time has a cost, as in order to communicate back and forth from the serial port, the port is kept opened and has a limited amount of bandwidth. This limitation affects the rate at which the program can run. In other words, this mode is slower than the Normal mode.

It is worth to comment that there is another option, called `Deploy to Hardware` with which we can export all the code to the micro-controller and so there is no need to be connected to a PC. But, in this case, the external mode will be used as we are going to use the Scope blocks of Simulink as oscilloscopes of the most important signals, such as the control signal or the output of the plant.

As we have seen, the main drawback of this mode is the limited amount of bandwidth. This issue must be calculated in order not to have missing data because of the fastness of the digital control system. With it, we will be able to select the maximum graphing sampling time too.

For that, the Simulink model shown in figure 3.3 has been created where a sinusoidal signal created in Simulink is compared with the same signal after passing it from the DAC pin and getting back through an analog input block.

This model has been computed with different sampling times and the signals and their sampling time variability has been analyzed for each iteration. We selected a sine signal of 10 Hz and started from a sampling time of 0.001 s and it is verified if there is loss of samples in the Scopes. If there are losses, the grafication period is increased until there is no loss of information.

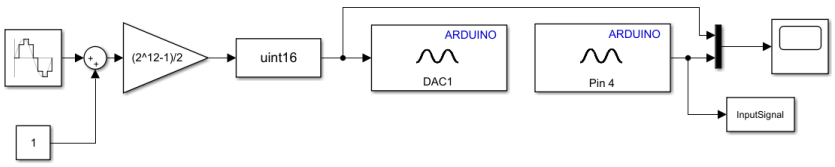


Figure 3.3: Simulink sheet used to analyze the sampling time variability.

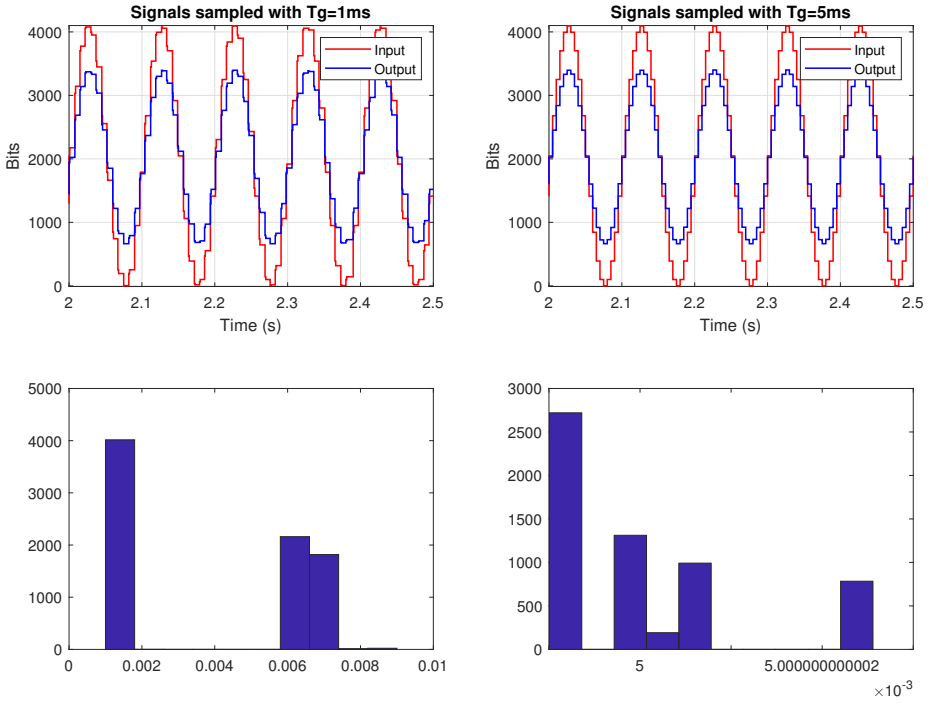


Figure 3.4: Results with different graphing times. On the left, $T_g = 0.001s$, on the right $T_g = 0.005s$.

To be able to make a quantitative analysis of the variability of the grafication period, the following MATLAB code has been used with which we obtain a histogram representing the distribution of periods occurred in a finite experimentation time.

```
h=InputSignal.time;
h=diff(h);
hmean=h-mean(h);
mean(h);
hist(h)
```

Figure 3.4 shows the result of two experiments done when calculating the loss of samples.

As we can see, from sampling periods higher than 5 milliseconds, no data is lost. From this value on, the acquired signal's period variability is very small, as it can be seen on the histogram, all the samples occur at 5ms of sampling time with an small jitter. While

when selecting a sampling time less than 5 milliseconds, in the example 1 milliseconds, half of the samples are centered on 1ms but the other half between 5 and 6 milliseconds.

Internal configuration of the laboratory PC's

As the Simulink models are going to be used by lots of student groups and can happen that they can modify and save unintentionally the models, we have created a student user on the laboratory PCs. Apart from this user, we have the `Administrator` user. The original practices' files are located on that last user, and every time student user is launched, the original files located on `Administrator` are copied. This is done by a batch file. We have named it iniCA.BAT and it command can be seen on Figure 3.5. It is located on the following directory: C:\Users\Control Automatic\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup.

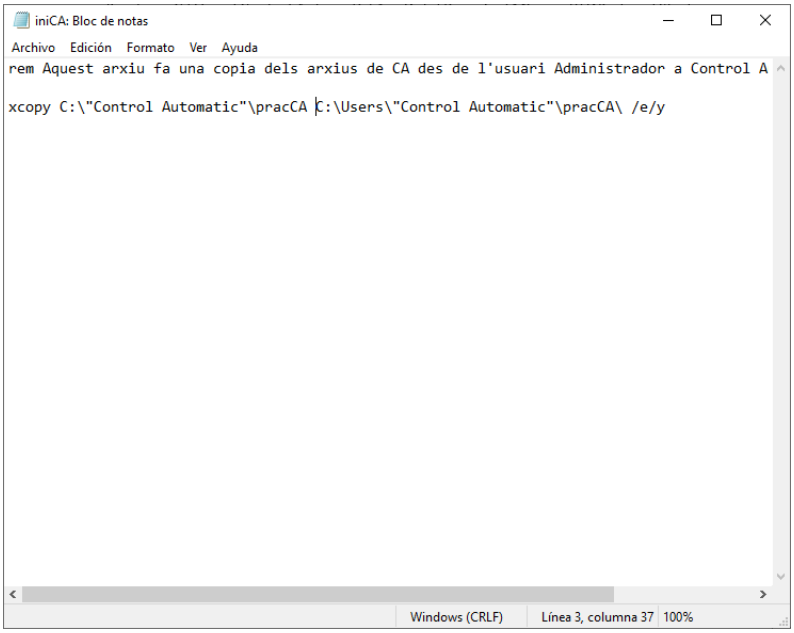


Figure 3.5: iniCA.BAT file.

As we can see, every time the student user is logged, a copy of the original files is done in "Control Automatic" user. The command /e copies any subfolder, even if it is empty and the command /y overwrites existing files without prompting the user. By the command *rem* we add a comment.

Chapter 4

Duino-Based Learning (DBL)

As we know, projects are hard to implement from scratch but it is the best way to put into practice all the knowledge learned theoretically and to learn which are the differences between the theory and the reality. Duino-Based Learning (DBL) provides a set of projects' starting point for any educator or learner. On it you can find build instructions for all setups, MATLAB live scripts with exercises, Simulink models and walk-through videos. All this material is available in three languages; English, Spanish and Catalan. The web page can be found in `https://duinobasedlearning.github.io/`. Figure 4.1 shows the layout of the platform. The web page has been created from GitHub, so we could link with the repository created.

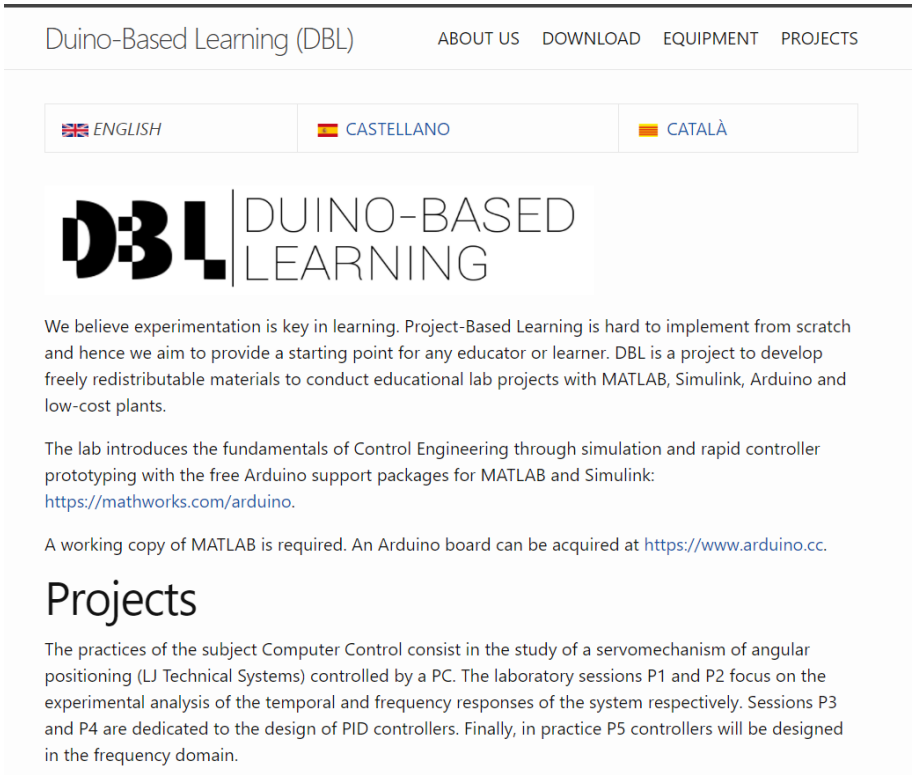


Figure 4.1: Duino-Based Learning's homepage.

As it can be seen, the web page is divided in 5 sections. The first one is the homepage, where all the main information is embedded, the second tab is the section where the authors' roles are introduced. On the following one, `download`, you can download the desired files available in the repository, this includes the gerbers used to create the shield, the instructions of how the shield has been designed, the Simulink models and the MATLAB live scripts. On `equipment` section, it is explained all the instrumentation used to carry out the projects, the reason of why they were chosen and some photos explaining how they are connected to each other. Finally, in the last tab, you can find the six workpackages, which are the ones carried out in the ETSEIB's Automatic Control laboratory [21], [19] with instructions for building low-cost plants, exercises and videos. In every lesson, the first link corresponds to MATLAB scripts along with data, graphs and explanations, while the second one corresponds to the explanatory videos, where theoretical issues are explained and the motor's real behaviour can be seen.

4.1 Workpackage 0: Introduction to Arduino programming using MATLAB/ Simulink

The main objective of this first workpackage is to learn how to configure Simulink in order to enable the connection and data transference between an Arduino board and the software itself. Along with this, the use of the specific Arduino analog and digital input and output blocks is explained. Before going in deep with the exercises, the Simulink sheet has to be configured in such a way that we enable the interaction with an Arduino board. For that, we have to take into account three steps. The first one, would be to choose the model we are using; the second, the running mode, as it has been told before, we will select the external mode, and the last one, the running time.

- Choosing the device:

To specify the hardware board we are using, in Simulink we go on **Simulation** > **Model Configuration Parameters**, or we will click on the button '*Model Configuration Parameters*' button  , which is in the Simulink toolbar.

Once inside, we will select the **Hardware Implementation** tab and in the **Hardware Board** option we will select the model we are using, in this project, as mentioned before, the Arduino Due is the board in use, see Figure 4.2.

- Simulation mode:

As told before, since the hardware to be used is an external hardware and we want to use the Simulink scope blocks as oscilloscopes, the mode in which the Simulink model is executed must be "external". Thanks to this mode, the Simulink block diagram were our application has being coded, will be executed in the target hardware.

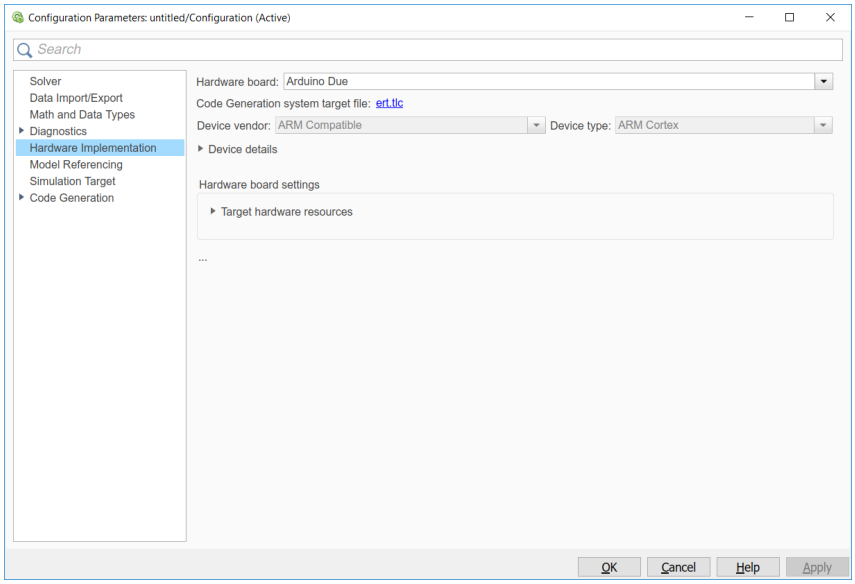


Figure 4.2: Configuration Parameter tab.

- Execution time:

Thanks to this option, we can choose the time we want the model to keep running. If we want to execute it until a certain interruption occurs, it will be delimited by typing *inf*.

Exercises:

As told before, four exercises will be carried out in this first workpackage, with which the concepts of digital and analog inputs and outputs of Arduino will be more easily internalized using Simulink.

Exercise 1: Digital output This first exercise tries to turn on the microcontroller's integrated led. This led is incorporated in almost all Arduino boards and its enumeration varies with the board in use. In the case of the Arduino Due, this led is connected to the digital pin 13, that is why the block diagram shown in Figure 4.3 has been made. Thanks to this block diagram we can turn on a led constantly by a digital high or make it blink by a pulse signal thanks to the manual switch.

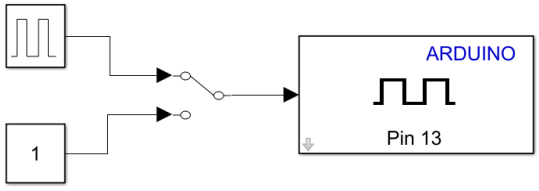


Figure 4.3: Block diagram to control the built in led.

Exercise 2: Digital input This second exercise consists on obtaining a digital signal from a button. When the button is pressed, a digital HIGH, or a 1 will be obtained on

the scope of Simulink. Therefore, when the button is not activated, what we will obtain will be a digital LOW or a 0. To do this, the block diagram an the connections shown in Figure 4.4 have been created.

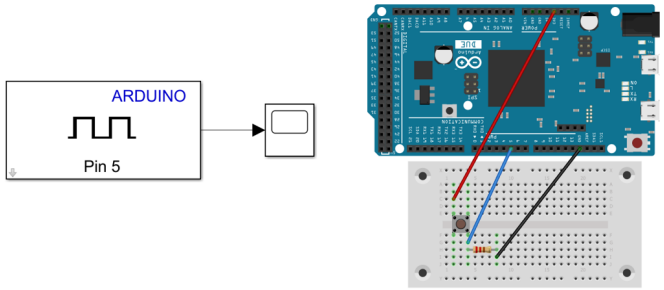


Figure 4.4: Block diagram and connection scheme for the digital input exercise.

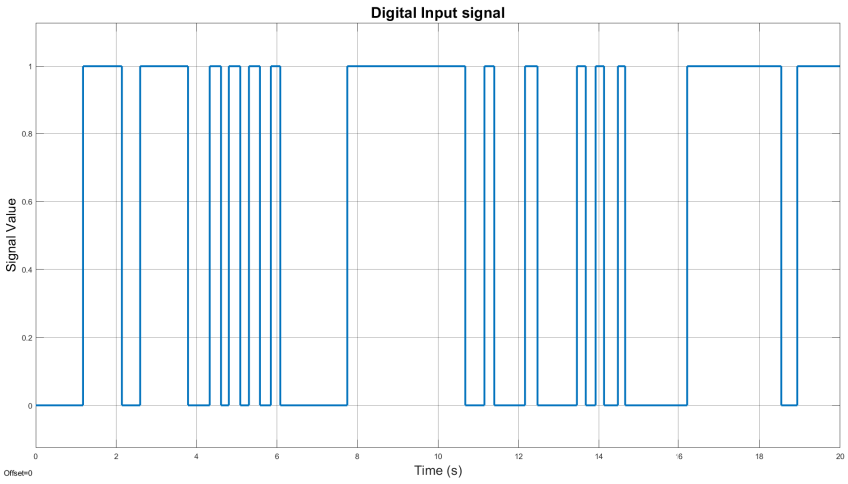


Figure 4.5: Signal obtained from an experiment carried out with the model above.

Exercise 3: Analog output In this exercise, we want to turn on a led by an analog signal. For this, we will use the DAC pin of the Arduino Due. These pins are responsible for converting digital signals into analog. We want to turn on and off a led progressively and at different frequencies, that is why the block diagram an the scheme shown in Figure 4.6 have been designed.

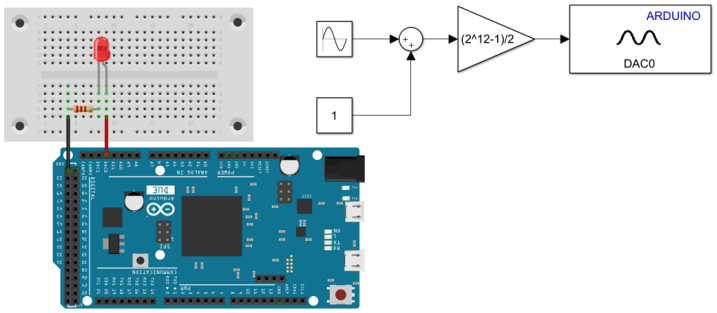


Figure 4.6: Block diagram to switch analogically a led.

Exercise 4: Analog input To conclude the first workpackage, it is desired to obtain an analog signal on the scope. For this, a potentiometer has been used. The circuit and block diagram created can be seen in Figure 4.7.

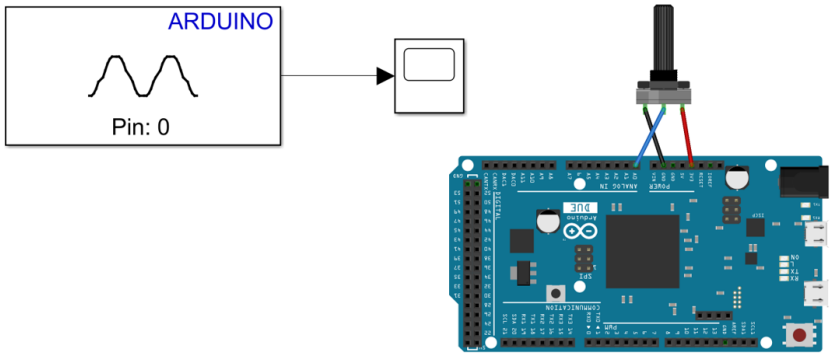


Figure 4.7: Block diagram to see the voltage obtained from a potentiometer.

Figure 4.8 shows a result obtained when simulating the exercise 4. As it can be seen, by turning the shaft of the potentiometer, we change the amount of resistance on either side of the wiper which is connected to the center pin of the potentiometer, which gives us a different analog input. When the shaft is turned all the way in one direction, there are 0 volts going to the pin, and we read 0. When the shaft is turned all the way in the other direction, there are 3.3 volts going to the pin and we read 4095.

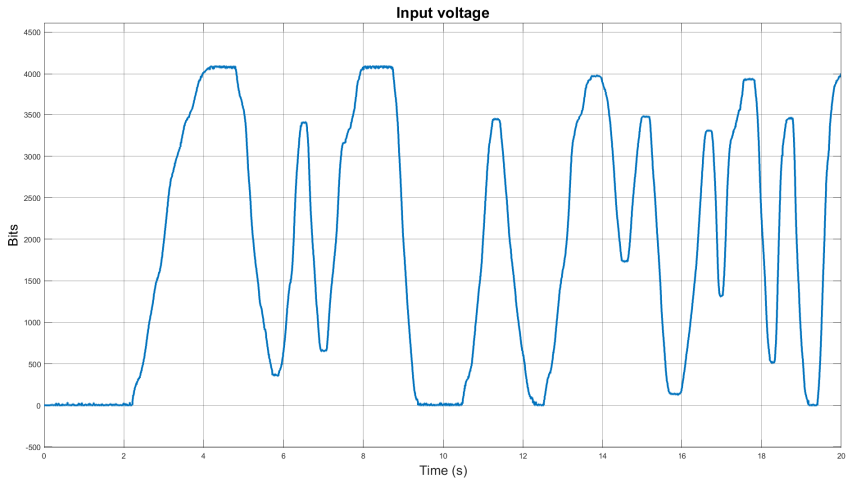


Figure 4.8: Signal obtained from an experiment carried out with a potentiometer.

4.2 Workpackage 1: Analysis of the temporal response of a digital control system

In this second workpackage we are going to analyze the temporal response of the LJ Technical Systems servo system. Firstly, we will obtain a behavior model based on the

study of the open-loop temporal response. We will then close a proportional control loop in speed and position output, and analyze the effect of the controller gain on the accuracy and stability of the closed loop system. We will also analyze the effect of the sampling time in the stability of the system, sampling it with different T_s. The steps to follow are those. First, we will analyze the response of the motor in open loop, taking as output the signal of the tachometer dynamo and with which we will obtain the mathematical model. Next, we will close the loop with a proportional controller and analyze the responses for different values of k_p. Finally, the performance of the plant will be analyzed, sampling it with different sampling times. The same steps will be followed, taking the potentiometer signal as output.

Velocity Control:

Figure 4.9 shows the block diagram for the open loop experiment built in Simulink editor. Depending on the output we take, we will obtain different mathematical models, but the form of the response will be the same. This means, the output of the motor can be the terminal voltage of the tachometer dynamo, the motor speed in rad/s, relating it with the tachometer's constant, or even the speed in $r.p.m$, relating the voltage with the speed by the help of the display on the motor. In this case, the response of the motor will be analyzed, taking the terminal voltage as output.

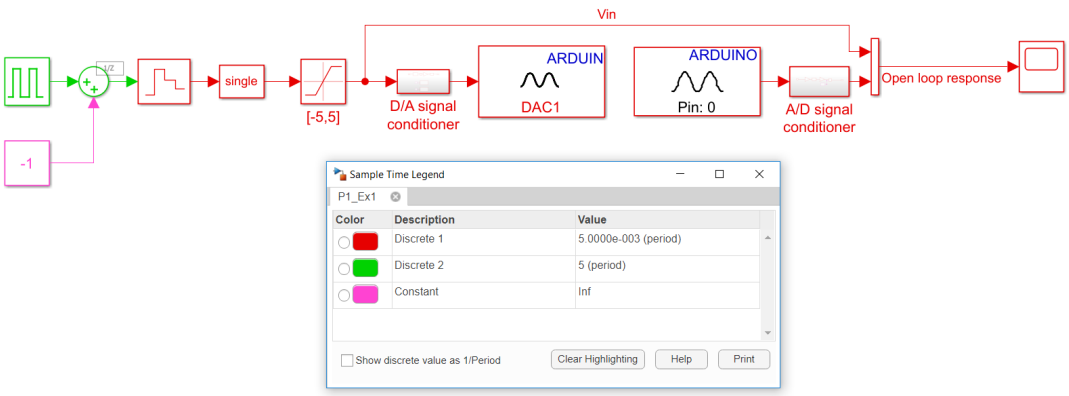


Figure 4.9: Simulink model for the velocity open-loop identification.

As we can see on Figure 4.10, the output resembles the response of a first order system, so in order to obtain its transfer function, from the Figure 4.11 the following mathematical formulas are obtained.

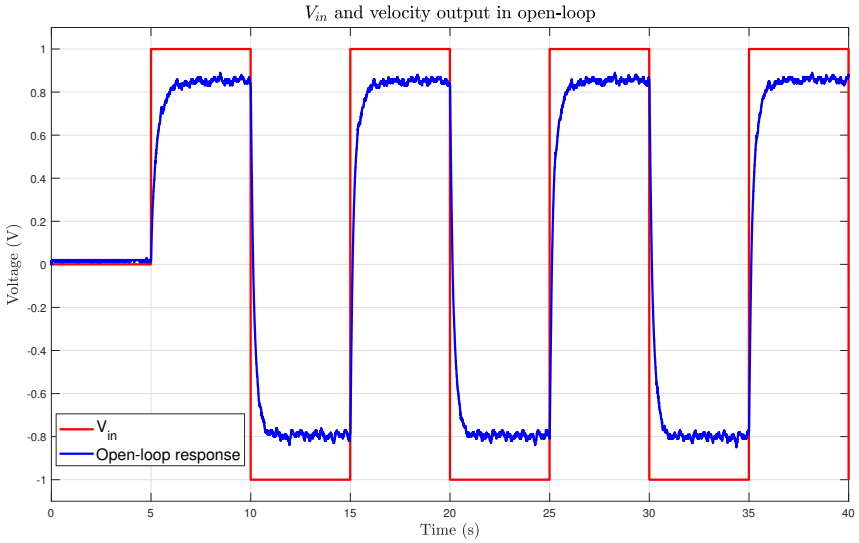


Figure 4.10: Velocity output open-loop response.

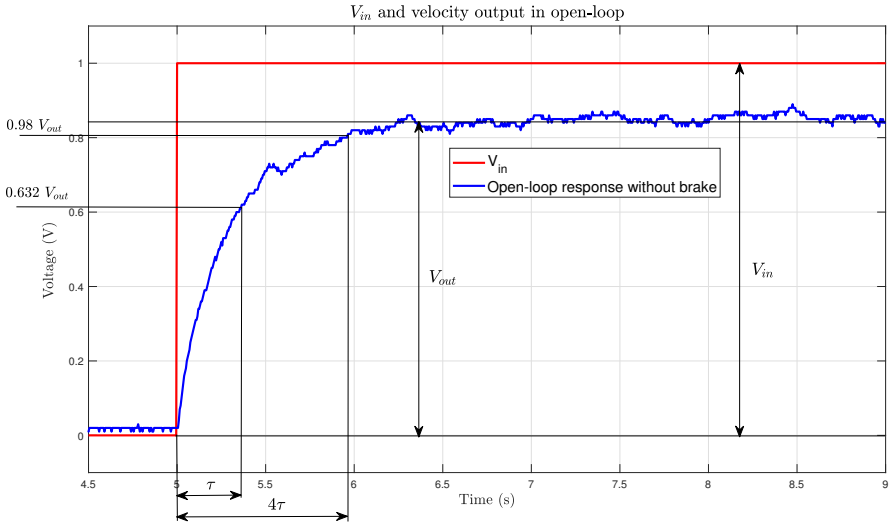


Figure 4.11: Obtaining the parameters from the temporal response graph.

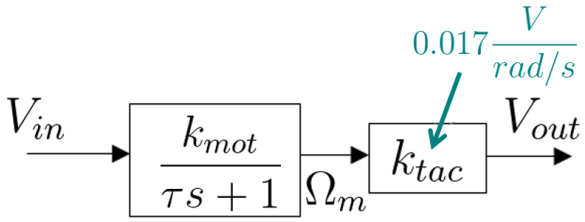


Figure 4.12: Motor circuitry explanation.

From Figure 4.11 we can conclude that:

$$\tau = 0.632 \cdot V_{out} \tag{4.1}$$

And from Figures 4.11and 4.12:

$$k_{mot} = \frac{V_{out}}{V_{in} k_{tac}} \tag{4.2}$$

So the obtained transfer function that models our system stays as:

$$G(s) = \frac{0.82}{0.26s + 1} \tag{4.3}$$

Figure 4.13 shows the comparison between the obtained open-loop response and the model's open-loop response. As it can be seen both responses are practically equal, what means the model can be accepted as the behaviour model.

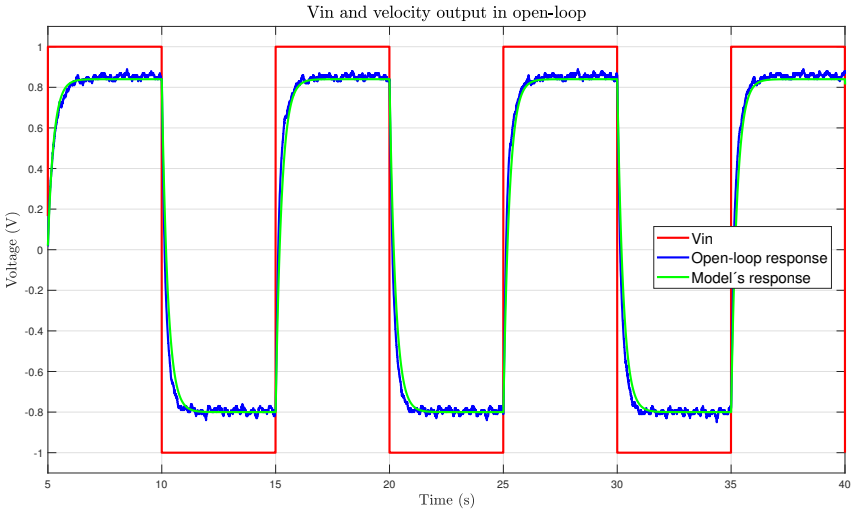


Figure 4.13: Comparison between the real and model responses.

In the case of open loop, it also worth to comment that, this gain varies depending on the brake. As you can see in Figure 4.14, if we apply the brake, the static gain will be reduced.

After the identification, the tachometer signal has been fed back and a proportional controller has been created being its Simulink model the one shown in Figure 4.15.

The model has been implemented with three different gains, $k_p = 1, 2$ and 3. By the Figure 4.16 we can analize how increasing the gain of the controller, the faster the response and the smaller the static error are. It can also be analyzed that the higher the gain, the higher the control signal.

Finally, before going on the position control, the effect of the sampling period is

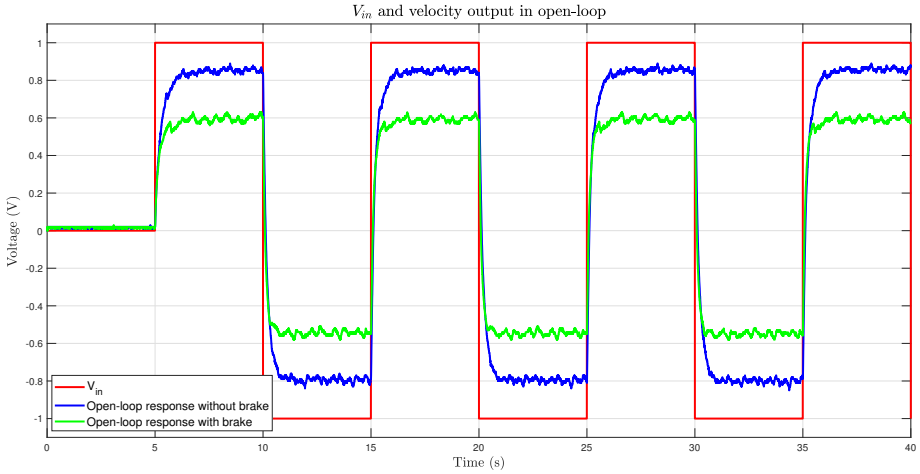


Figure 4.14: Effect of the application of the brake in open-loop.

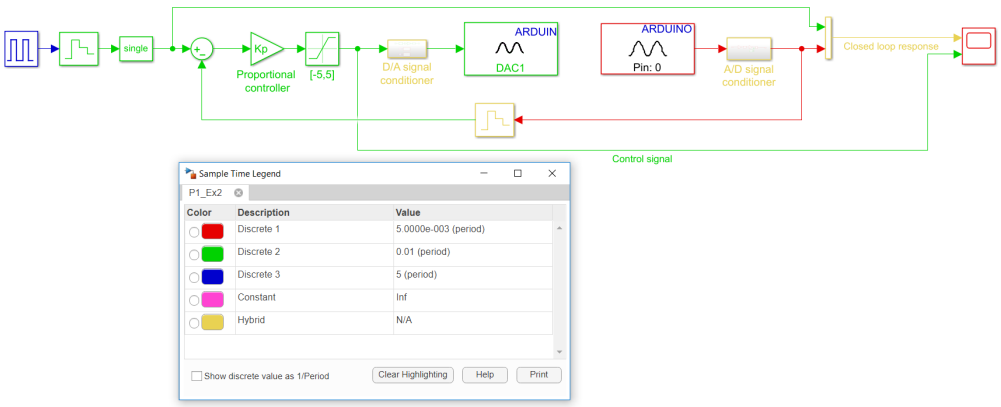


Figure 4.15: Simulink model for exercises 2, 3, 5 and 6.

analyzed. For that the Simulink model used for the previous exercise is used (Figure 4.15), changing the reference so that it varies between $(-2, 2)V$. In this case, instead of changing the value of k_p, the value of T_s (sampling period) is changed, using in all cases a value of $k_p = 1$ for the proportional controller.

As we can see, the response of the system with a sampling period of 0.2 seconds, the response of magenta color, has an overshoot and with greater sampling times, the output is even more oscillating. Theoretically, this critical sampling time can be obtained by obtaining the digital transfer function of the closed loop system, as it is shown in the following lines.

For that, the open-loop digital transfer function will be calculated using a zero-order holder.

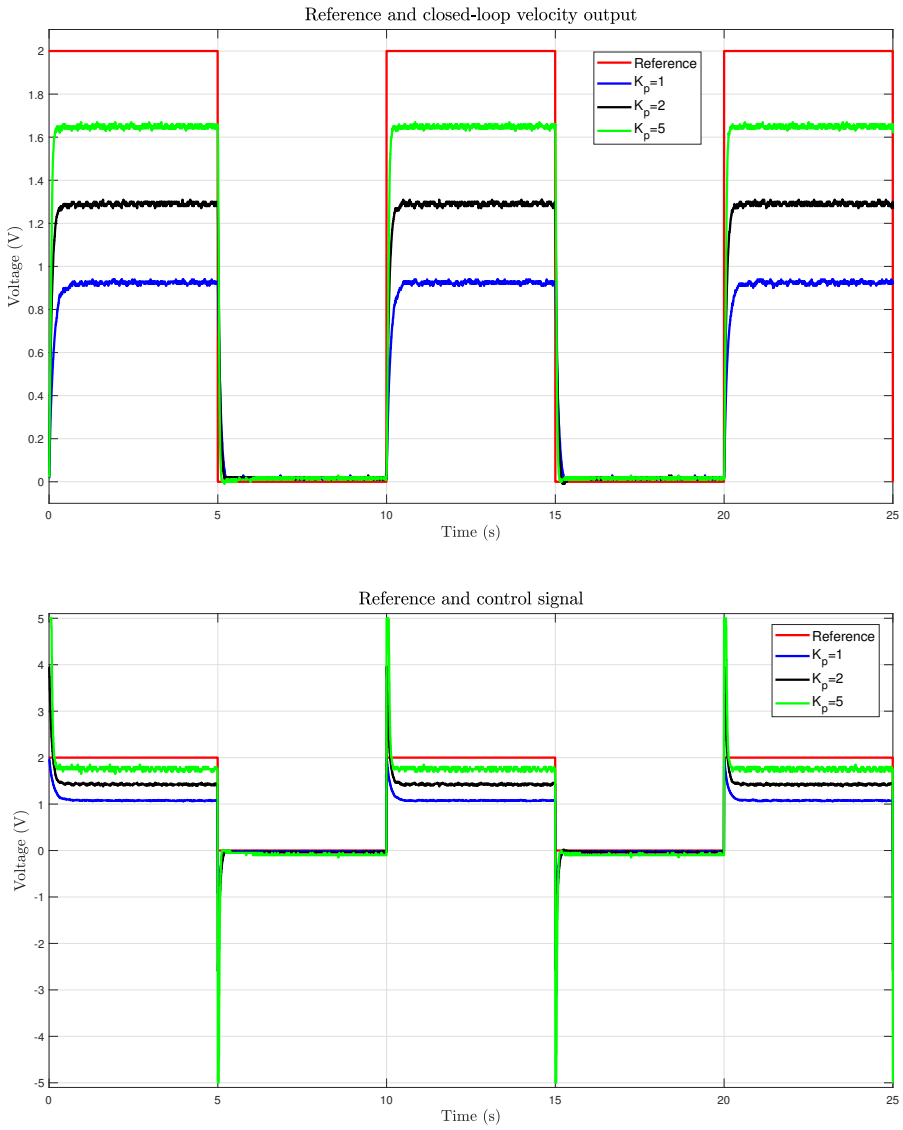


Figure 4.16: Results obtained for exercise 2 of the workpackage 1.

$$\begin{aligned} L(z) &= \mathcal{Z}\left\{k_p \cdot G_{mot}(s) \cdot G_{zoh}(s)\right\} = \mathcal{Z}\left\{k_p \cdot \frac{k_{mot}}{\tau_{mot}s+1} \cdot \frac{1-e^{-T_s \cdot s}}{s}\right\} = \\ &= k_p k_{mot} \frac{1-\alpha}{z-\alpha} \end{aligned} \tag{4.4}$$

After that, the digital closed-loop transfer function is computed. Knowing we have a unitary feedback, the transfer function will stand as follows:

$$G(z) = \frac{L(z)}{1+L(z)} = \frac{k_p k_{mot}(1-\alpha)}{z-\alpha+k_p k_{mot}(1-\alpha)} \quad \text{where} \quad \alpha = e^{\frac{-T_s}{\tau_{mot}}} \tag{4.5}$$

From which evaluating the denominator, we can obtain the expression of the closed-

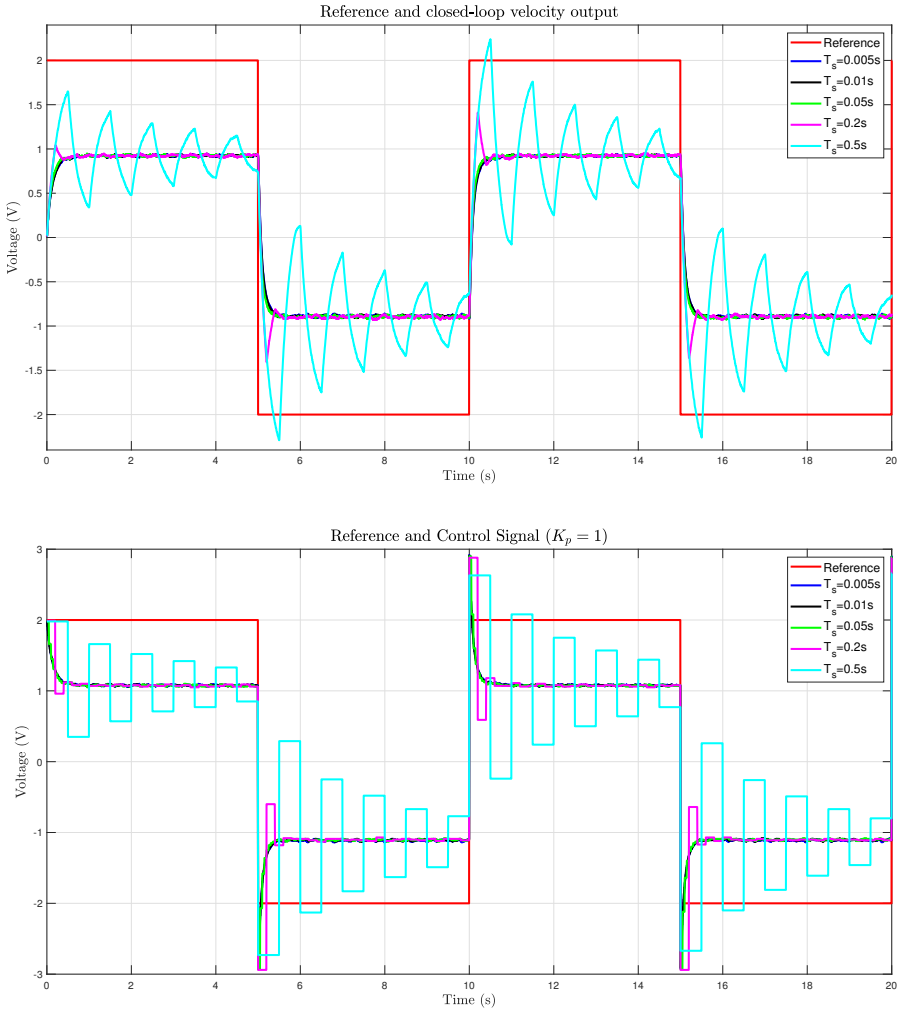


Figure 4.17: Results obtained for exercise 3 of the workpackage 1.

loop pole.

$$p = -k_p k_{mot} - (1 + k_p k_{mot})\, e^{\frac{-T_s}{\tau_{mot}}} \tag{4.6}$$

And evaluating it to zero, which will be the point where the system will start to oscillate, the expression for the sampling period is:

$$T_s = \tau_{mot} \cdot \ln \left| \frac{1 + k_p k_{mot}}{k_p k_{mot}} \right| \tag{4.7}$$

Which computing it with our parameters, we obtain a sampling period of $T_s = 0.2s$.

Position Control:

For the position control experiments the same experiments will be carried out, but taking as output the potentiometer's voltage signal. To do this, we have obtained a new

mathematical model that represents the position open-loop dynamics. This has been achieved from the mathematical model obtained in the previous experiment, where a reducer, an integrator and the potentiometer's constant have been added (see Figure 4.18), in order to be able to relate the output voltage with the input voltage.

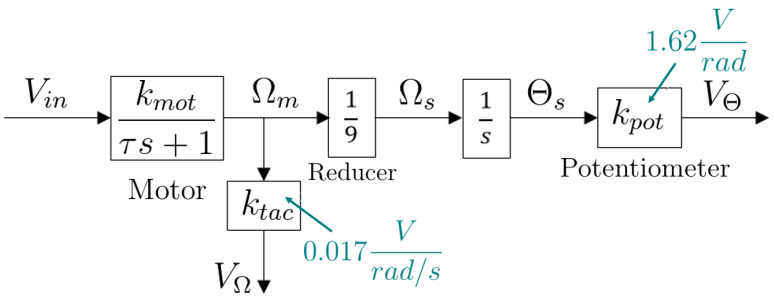


Figure 4.18: Motor circuitry explanation for position output.

$$G_{pos}(s) = \frac{8.68}{s(0.26s+1)} \tag{4.8}$$

If we analyze Figure 4.19 we can see, how the response of the potentiometer is a periodical signal with saw wave form. The saw wave form varies symmetrically depending on the turning direction. On one hand, this comes from the fact that the plant is of type 1, with which, a ramp type signal will be obtained in front of a step type input. On the other hand, the periodicity occurs for each spin of the potentiometer's axis.

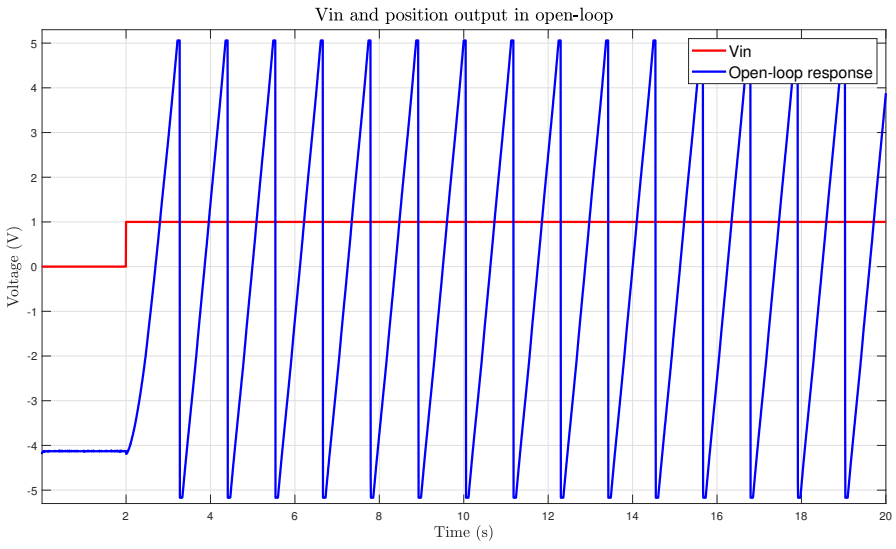


Figure 4.19: Open-loop position response.

As has been done in the case of the speed control, a proportional controller has been designed. The Simulink model for the closed loop system is the one implemented before (Figure 4.15), but taking into account that we have to acquire the potentiometer's signal.

As in the position case, the system is of type 1, in all cases, the stationary error is null, but what varies are the overshoot and the settling time. As it can be seen on Figure 4.20 the higher the k_p, the greater the overshoot and and the longer the settling time are.

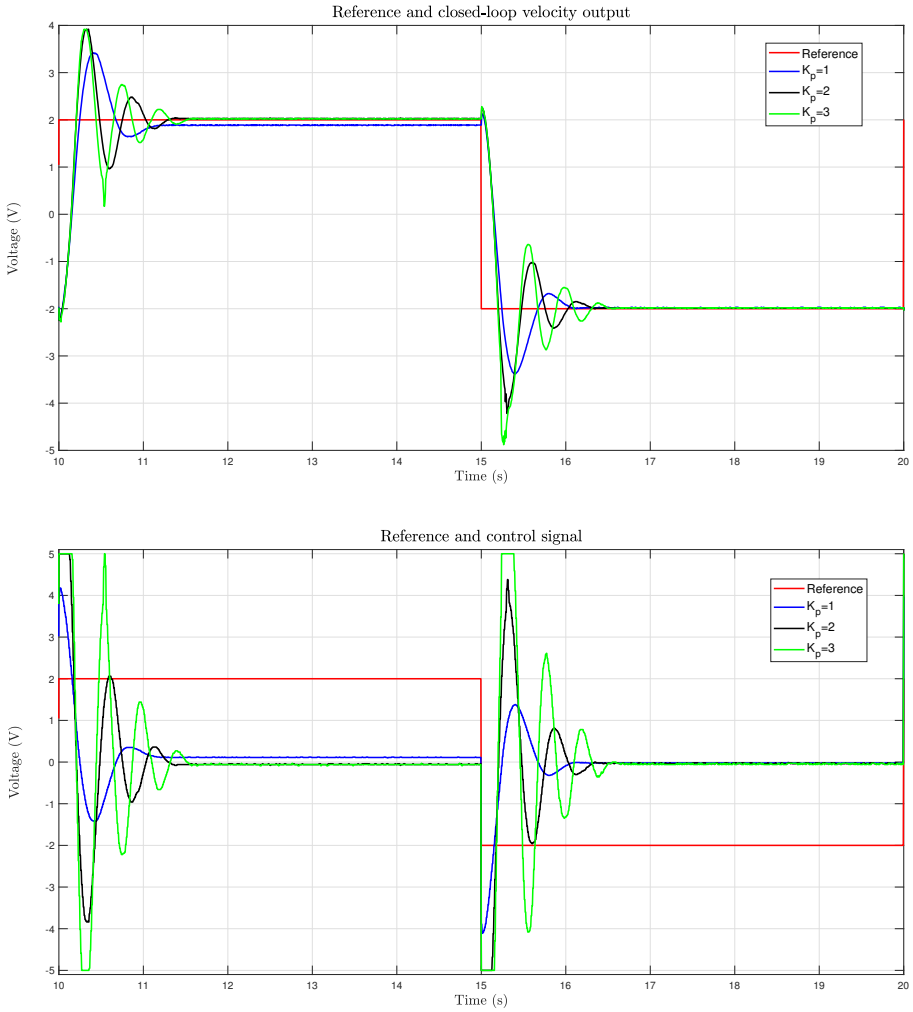


Figure 4.20: Results obtained for exercise 5 of the workpackage 1.

Finally, the accuracy and stability of the system have been analyzed for different sampling periods. In this case also, to analyze from what value the system will become unstable, as we have done previously, the plant has been digitized with help of a zero order holder and the closed-loop digital model has been obtained. In this case, we will evaluate the stability by applying Jury's stability criterion to the denominator.

The digital open-loop transfer function is the following:

$$\begin{aligned} L(z) &= \mathcal{Z}\left\{k_p \cdot G_{pos}(s) \cdot G_{zoh}(s)\right\} = \mathcal{Z}\left\{k_p \cdot \frac{k_{mot}}{\tau_{mot}s+1}\frac{1}{s}\frac{1}{N}k_{pot} \cdot \frac{1-e^{-T_s \cdot s}}{s}\right\} = \\ &= k_p k_{tot}\tau_{mot}\frac{\left(\frac{T_s}{\tau_{mot}}-1+\alpha\right)z+\left(1-\alpha-\frac{T_s}{\tau_{mot}}\alpha\right)}{z^2-(1+\alpha)z+\alpha} \end{aligned} \tag{4.9}$$

By the transfer function above, we obtain the following digital close-loop representation:

$$G(z) = \frac{L(z)}{1+L(z)} = \frac{k_p k_{tot}\tau_{mot}\left[\left(\frac{T_s}{\tau_{mot}} - 1 + \alpha\right)z + \left(1 - \alpha - \frac{T_s}{\tau_{mot}}\alpha\right)\right]}{z^2 - (1+\alpha)z + \alpha + k_p k_{tot}\tau_{mot}\left[\left(\frac{T_s}{\tau_{mot}} - 1 + \alpha\right)z + \left(1 - \alpha - \frac{T_s}{\tau_{mot}}\alpha\right)\right]} \tag{4.10}$$

From which the characteristic equation is:

$$Q(z) = z^2 + \left[-1 - \alpha + k_p k_{tot} T_s + k_p k_{tot}\tau_{mot}(\alpha - 1)\right]z + \alpha + k_p k_{tot}\tau_{mot}\left(1 - \alpha - \frac{T_s}{\tau_{mot}}\alpha\right) \tag{4.11}$$

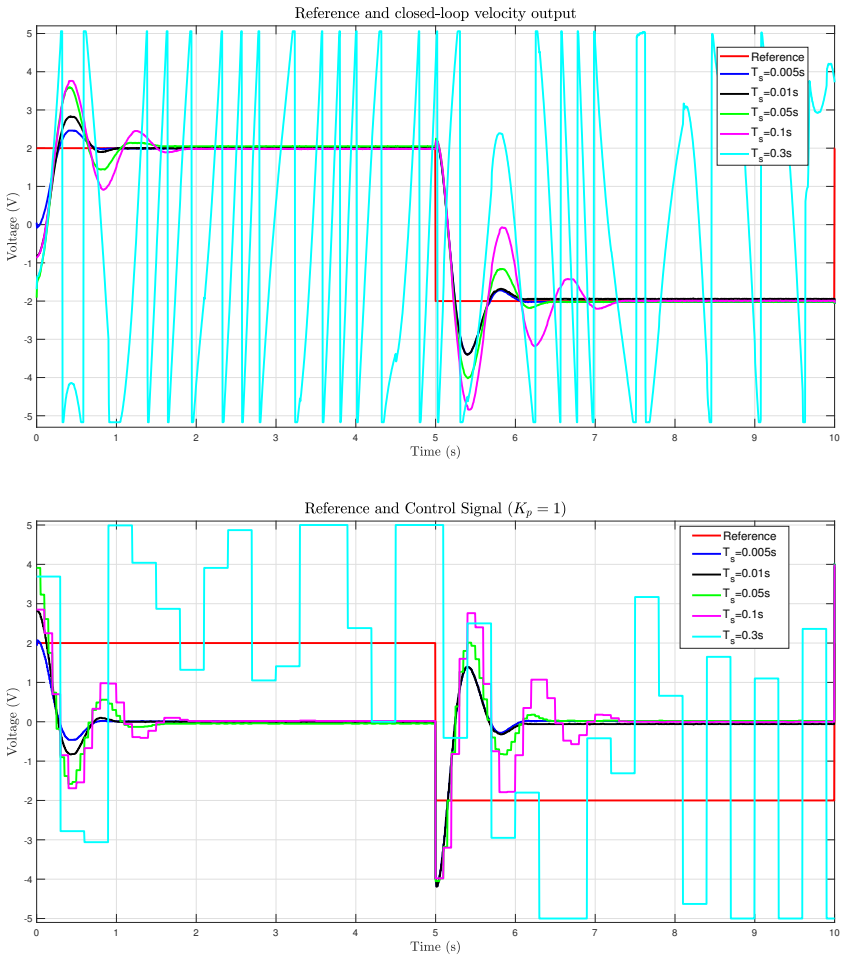


Figure 4.21: Results obtained for exercise 5 of the workpackage 1.

From Jury's criterion, see Appendix A, we obtain that the range for the sampling period, so that the system is stable is $T_s \in [0, 0.279]$ seconds. From this point on, the

system will become unstable. As can be seen in Figure 4.21, at sampling periods of 0.3 seconds the system is already unstable. While, the overshoot and the setting time for the stable cases, are higher when T_s increments.

4.3 Workpackage 2: Analysis of the frequency response of a digital control system

The workpackage 2 analyzes the frequency response of the LJ Technical Systems servo system. The objective of this practice is to obtain the theoretical and experimental frequency responses of the open-loop system. Along with this, the Nyquist criterion will be applied in order to determine the range of values of a proportional controller so that the system is stable. It will be verified experimentally. Figure 4.22 shows the Simulink model implemented for the open-loop frequency response analysis. As it can be seen, a sine input is aplied and the response can be compared with the input on the scope available. This sine, will have 2V of amplitude and the research will be carried out, by analyzing the response of the system with input frequencies of Hz: 0.25, 0.5, 0.8, 1, 2, 4, 5, 10 and 20Hz.

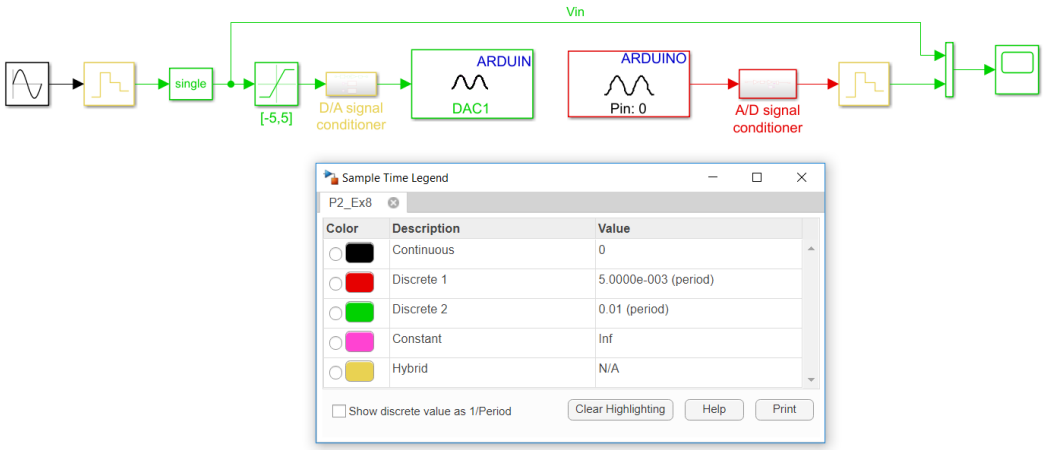


Figure 4.22: Simulink block diagram for the frequency response analysis.

Figure 4.23 shows some experiments done with different sines. Note that the time scale is different in every case.

By obtaining the gain change and phase shift in different frequencies we are able to obtain the experimental Nyquist diagram. Table 4.1 shows the values obtained in different frequencies. If we plot them in a complex graph, we can see how the experimental Nyquist diagram resembles the theoretical one, see Figure 4.24. The small differences come from the measurement errors and non modeled dynamics.

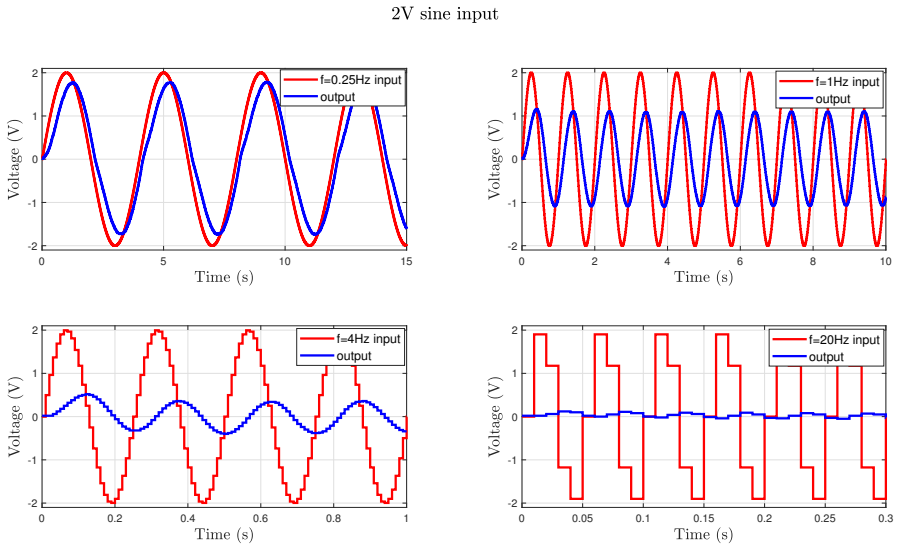


Figure 4.23: Experiments done with different frequencies sine inputs. Up left $f = 0.25Hz$, up right $f = 1Hz$, down left $f = 4Hz$ and down right $f = 20Hz$.

$f(Hz)$	0.25	0.5	0.8	1	2	4	5	10	20
Gain (dB)	0.8	0.7	0.6	0.54	0.34	0.2	0.16	0.085	0.06
Phase (deg)	18.45	32.4	46.08	52	68.4	86.4	108	126	144

Table 4.1: System's gain and phase change for different input frequencies

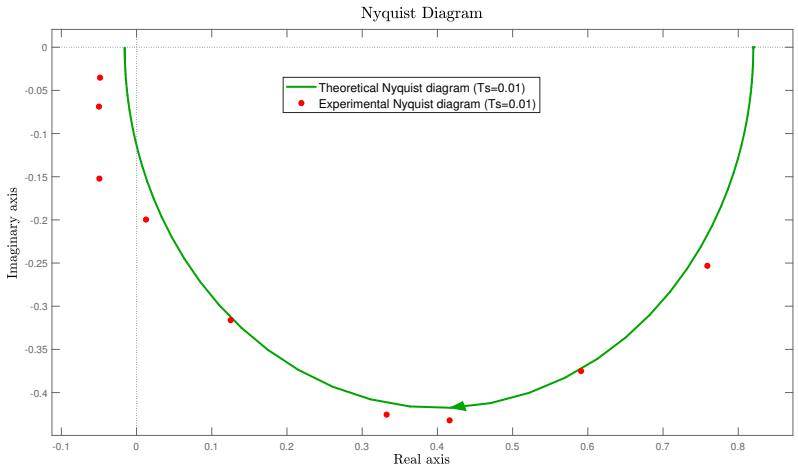


Figure 4.24: Comparison between the ideal and experimental Nyquist.

The MATLAB code for this plot is the following:

```
% Experimental gains vector
G=[0.8 0.7 0.6 0.54 0.34 0.2 0.16 0.085 0.06];
% Experimental phase shifts vector (in radians)
F=[18.45 32.4 46.08 52 68.4 86.4 108 126 144]*(-pi/180);
% Calculation of the complex in Cartesian form
X=real (G.*exp(F*j));
Y=imag(G.*exp(F*j));

% Introduction of plant parameters
K0=0.82;
tau0=0.26;
planta=tf([K0],[tau0,1])

% Discrete time model
Ts=0.01;
plantad=c2d(planta,Ts,'zoh');

% Draw the theoretical and experimental Nyquist diagram
figure(1)
nyquist(plantad,'g')
hold on
plot(X,Y,'r*')
title('Nyquist Diagram');
xlabel('Real axis');
ylabel('Imaginary axis');
legend('Theoretical Nyquist diagram (Ts=0.01)','Experimental Nyquist
    diagram (Ts=0.01)')
```

Many times, to calculate the stability of a control system like the one presented in Figure 4.25, Nyquist criterion is used. For that, the formula 4.13 is implemented, where Z is the number of unstable closed-loop poles, P is the number of unstable open-loop poles and N is the number of encirclement to the critic point.

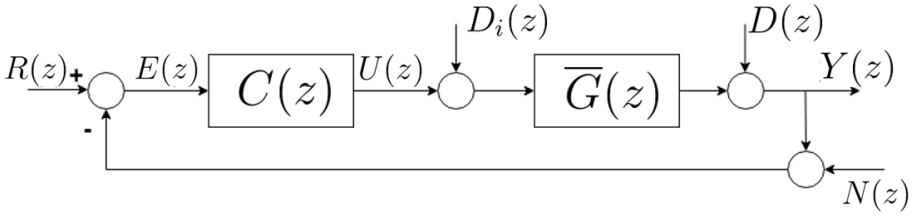


Figure 4.25: Nyquist criterion.

$$Z = N + P \tag{4.12}$$

$\cdot Z$ is the number of zeros of $1 + C(z)\overline{G}(z)$ (Number of unstable closed-loop poles)

$\cdot P$ is the number of poles of $C(z)\overline{G}(z)$ (Number of unstable open-loop poles)

$\cdot N$ is the number of encirclements of the point $-1 + 0j$ by $C(j\omega)\overline{G}(j\omega)$

Switching to our case, we can know whether the system is stable or not by the formula above. As we have seen on the Nyquist plot, in our case, as there is no encirclement in the critical point, the system will become unstable when the equation $1 + k_p\overline{G}(z) = 0$ meets.

$$-1 = 0.9623 - 0.0309k_p \rightarrow k_p = 63.42 \tag{4.13}$$

In order to experimentally validate these results, the Simulink diagram shown in Figure 4.15 has been used, where the output of the system has been feedback and a proportional controller has been created. So, we will analyze the responses in the stable and unstable ranges.

We have taken $k_p = 5$ and $k_p = 65$ in order to see the behavior in a stable and in an unestable range.

Figure 4.26 show the simulation results for $k_p = 5$. As expected, the system behavior is stable.

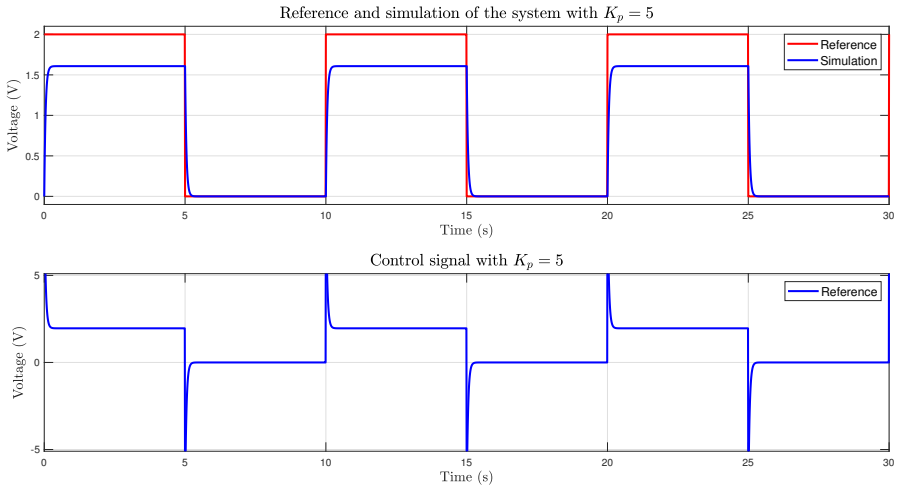


Figure 4.26: Simulation with $k_p = 5$.

As we can see the experimental results (Figure 4.27) is qualitatively similar to the one obtained in the simulation (Figure 4.26). But when we apply a k_p value greater

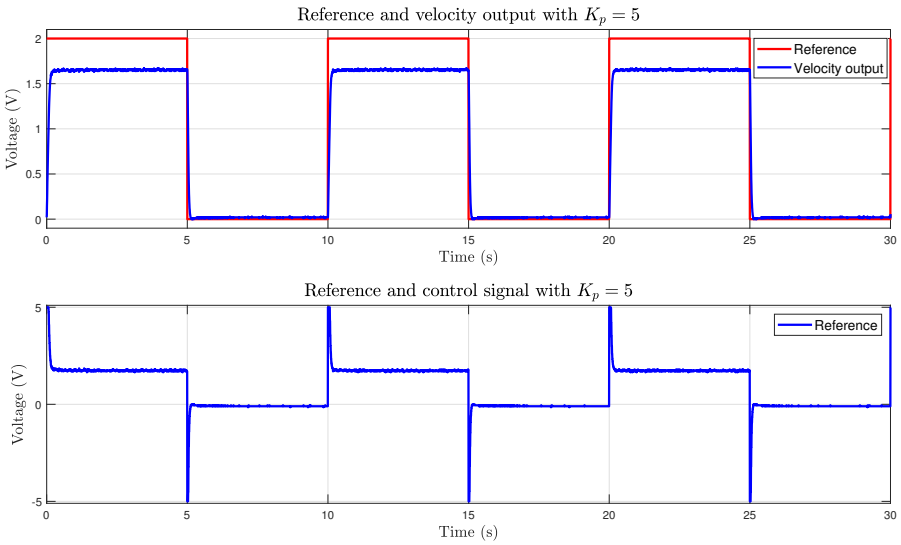


Figure 4.27: Velocity ouput with $k_p = 5$.

than 63.42, in this case 65, we can see that the simulation of closed-loop system behavior becomes unstable (see Figure 4.28).

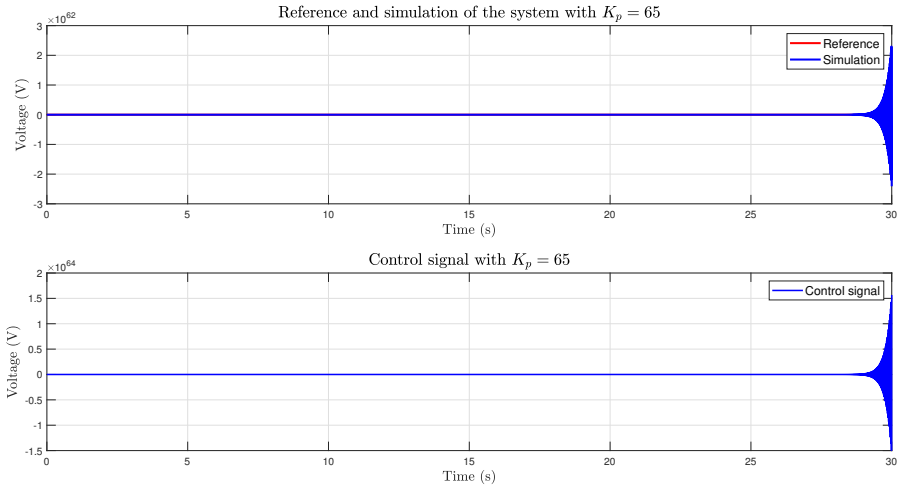


Figure 4.28: Simulation with $k_p = 65$.

In Figure 4.29 it is shown the experimental result obtained for the same case. As it can be seen the results are qualitatively different. In this case, the output is following the reference, and apparently the system behavior is stable. The main difference with the simulation model is that in the experimental setup the control action is saturated an can only take values between -5 and 5 Volts. Consequently the closed-loop behavior can not be explained in terms of a linear system.

For the position analysis, the theoretical Nyquist has been obtained using MATLAB, see Figure 4.30 as, in this case, it is harder to extract the gain change and phase shift experimentally. The transfer function of the system sampled with $T_s = 0.01$ is express by

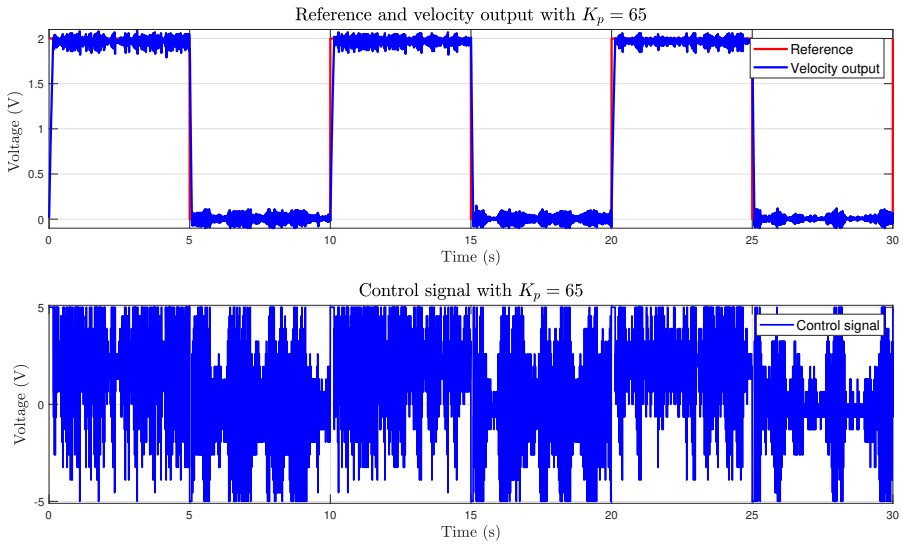


Figure 4.29: Velocity ouput with $k_p = 65$.

the following formula which has been also obtained with MATLAB.

$$\overline{G}(z) = \frac{0.001648z + 0.001627}{z^2 - 1.962z + 0.9623} \tag{4.14}$$

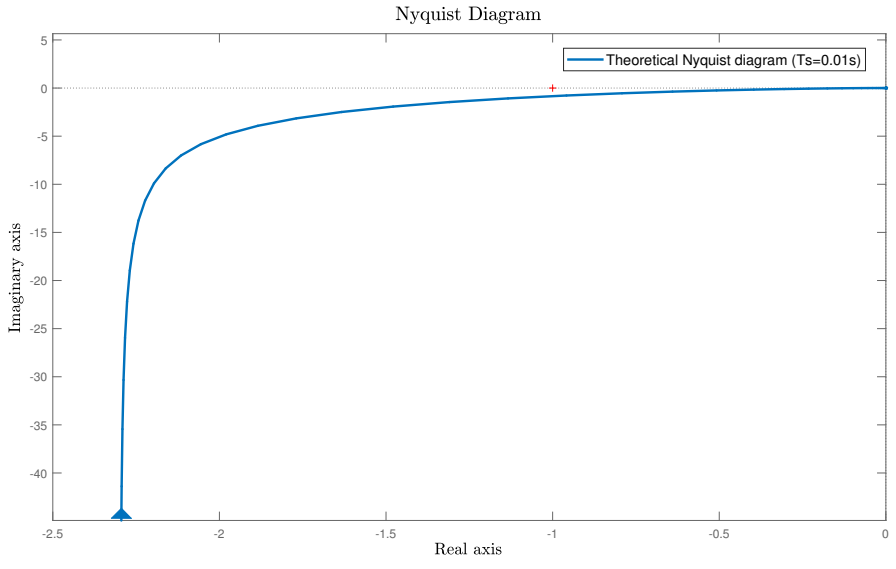


Figure 4.30: Theoretical Nyquist diagram for the position case.

This diagram is qualitatively different from the velocity one. The curve begins at infinite due to the existence of an integrator in the transfer function. To determine if the closed-loop system will be stable or unstable we need to know the relative position between the -1 point and the curve. It can be seen that the curve crosses the real axis

at the right of the -1 point which means no encirclement is produced and the closed-loop system will be stable.

As in the case before, when closing the loop with a proportional controller and increasing the gain of it, the system will get closer to the instability. In this case, the limit for the controller is 23.11. So, we will analyze the responses in the stable ($k_p = 2$) and unstable ($k_p = 25$) ranges using for that, the model shown on Figure 4.15 again.

Figures 4.31 show the simulation results for $k_p = 2$. As expected, because the system is of type 1, after some time, the system will follow the reference.

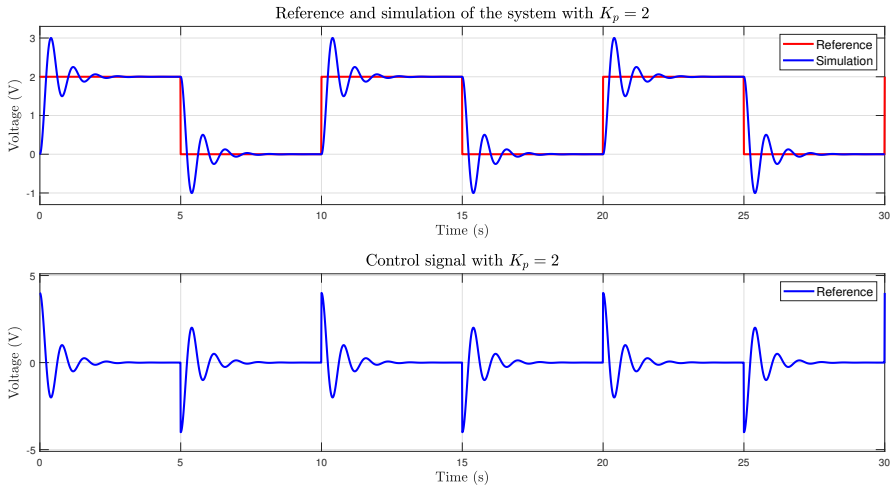


Figure 4.31: Simulation with $k_p = 2$.

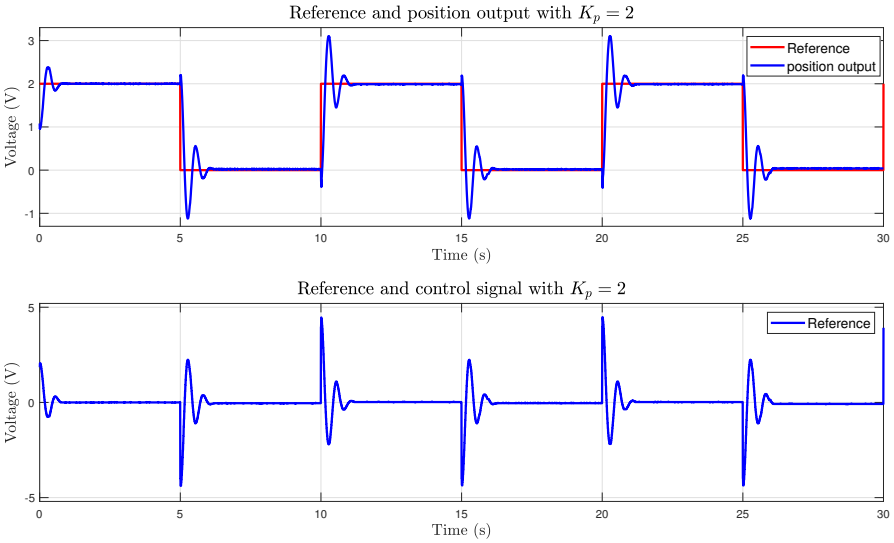


Figure 4.32: Position ouput with $k_p = 2$.

As we can see in the experimental result (Figure 4.32), a stable closed-loop system is obtained. The little differences of the graphs come from the non modeled dynamics.

When we apply a k_p value greater than 23, in this case 25, we can see that the closed-loop system behavior becomes unstable.

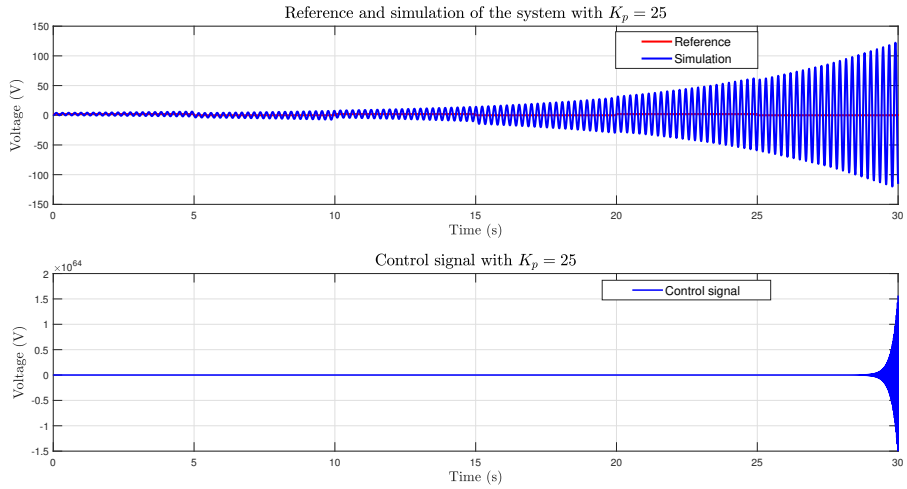


Figure 4.33: Simulation with $k_p = 25$.

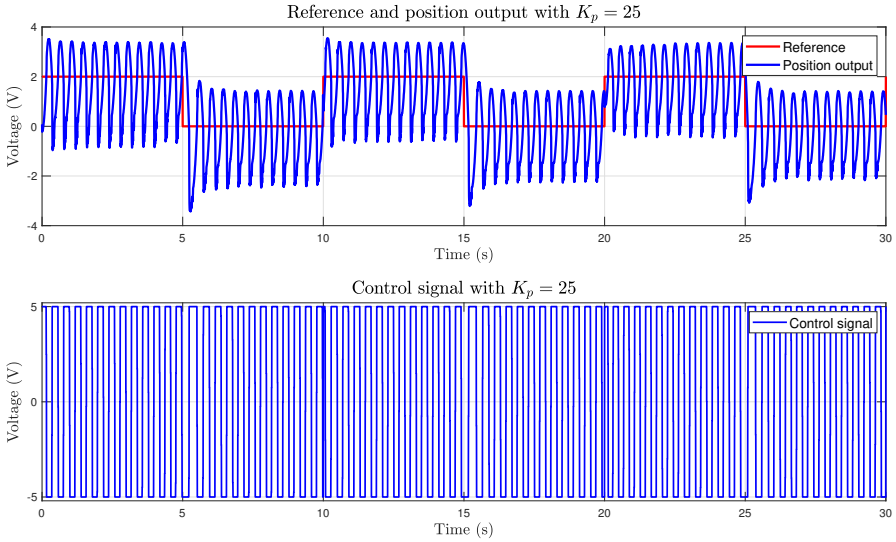


Figure 4.34: Position ouput with $k_p = 25$.

In Figure 4.34 is shown the experimental results obtained for the same case. As it can be seen the results are qualitatively different. In this case, the output is following the reference in an oscillating way and apparently the system behavior is stable. The main difference with the simulation model is again that in the experimental setup the control action is saturated an can only take values between -5 and 5 Volts. Consequently, as in the velocity control case, the closed-loop behavior can not be explained in terms of the linear system.

4.4 Workpackage 3: Design and implementation of PID controllers

This third workpackage explains how to design some PID controllers, which will also be implemented in the LJ Technical Systems servo system. First of all, a digital PI controller for the velocity output will be designed and afterwards, a digital PID for the position output shaft. Both are digital controllers, which have been discretized by the trapezoidal approach and designed by the pole placement method.

As MATLAB, does not offer any direct way to perform the design of a controller by pole placement, we have to obtain the digital transfer function of the PID and close the loop with the system we want to control.

The transfer function for the digital PID controller obtained by the trapezoidal approach is the one shown in the following lines:

$$PID(z) = k_p + k_i \frac{T_s(z+1)}{2(z-1)} + k_d \frac{z-1}{T_s \cdot z} \tag{4.15}$$

Where T_s is the sampling time and k_p, k_i and k_d are the controller's parameters. In order to simplify the tuning process the following variable change is suggested:

$$k_i^* \triangleq \frac{k_i \cdot T_s}{2}, k_d^* \triangleq \frac{k_d}{T_s} \tag{4.16}$$

So, the discrete PID takes the following form:

$$PID(z) = k_p + k_i^* \frac{z+1}{z-1} + k_d^* \frac{z-1}{z} \tag{4.17}$$

As all the controllers implemented in this document have integral part, we have decided to include an anti-windup filter in all the Simulink models. After some analysis, we realized the anti-windup must have the following structure (you can find the complete analysis on Appendix B):

$$AW(z) = \frac{k_{aw}}{z+0.5} \quad \text{and select } k_{aw} = \frac{1}{k_i} \tag{4.18}$$

In this way, the two poles of the anti-windup closed-loop are located on 0.

Design and implementation of a PI controller by pole placement.

When designing a controller, the first step is to choose the controller that best suits to our system. In this case, as we have a first order system, the best option is to choose a PI

controller, so we will have a closed-loop system with two poles to fix by two parameters (k_p and k_i). Once knowing the controller we will use, the following step is to decide some control objectives. In this case, we want the closed-loop system to reach the steady-state in 0.8 seconds with no overshoot.

Finally, in order to obtain the tuning equations, the closed-loop transfer function must be obtained. Following the structure in Figure 4.35.

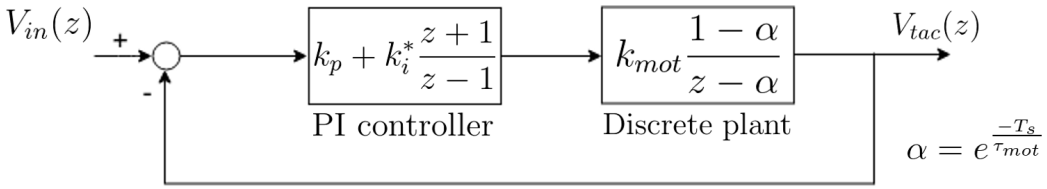


Figure 4.35: Block diagram of the PI control.

The closed-loop transfer function is the following:

$$CL(z) = \frac{\mathrm{PI(z)}\,\overline{G}(z)}{1 + \mathrm{PI(z)}\,\overline{G}(z)} = \frac{k_{mot}\left(k_p + k_i^*\right)\left(z + \frac{k_i^* - k_p}{k_i^* + k_p}\right)(1-\alpha)}{(z-1)(z-\alpha) + k_{mot}\left(k_p + k_i^*\right)\left(z + \frac{k_i^* - k_p}{k_i^* + k_p}\right)(1-\alpha)} \tag{4.19}$$

The controller parameters are obtained by pole placement technique. In this technique, the closed-loop poles are placed in pre-determined locations in the s plane, so that we can control the response of the system. The closed-loop dynamics can be adjusted by selecting the correct gains in the three actions of the controller, for that, a desired characteristic equation is defined and then, the gains are tuned.

So, the procedure is the following. First a desired characteristic equation is obtained, where the specifications told before are embedded. Then the closed-loop characteristic polynomial is obtained, and finally, by matching the coefficient of the denominator of the closed-loop system with the coefficient of the desired characteristic equation we will obtain a set of equations to be solved and obtain the tuning equations.

In this case, the tuning equations are the following:

$$\begin{aligned} k_p &= \frac{-\left(-2e^{T_s\xi\omega_n}\cos\left(\omega_n T_s\sqrt{1-\xi^2}\right) + e^{-2T_s\xi\omega_n} + 1\right)}{T_s k_{mot}(\alpha - 1)} \\ k_i &= \frac{-2e^{T_s\xi\omega_n}\cos\left(\omega_n T_s\sqrt{1-\xi^2}\right) + e^{-2T_s\xi\omega_n} + 1}{2k_{\mathrm{mot}}(\alpha-1)} - \frac{\alpha - 2e^{-T_s\xi\omega_n}\cos\left(\omega_n T_s\sqrt{1-\xi^2}\right) + 1}{k_{\mathrm{mot}}(\alpha-1)} \end{aligned} \tag{4.20}$$

Figure 4.37, shows the responses of a simulation done before implementing the con-

troller in the real system, in green, and the real response, in blue. As can be analyzed, the two responses are practically the same, but if we analyze the desired specifications, we can see that these are not met in none of the cases. This comes from the fact that the PI controller together with a first-order plant have as result a system with two poles and a zero. This zero, has a negative effect on the dynamics of the system, usually making faster responses, here, an overshoot is introduced in the step response.

For the PI control implementation, the block diagram shown in Figure 4.36 has been designed.

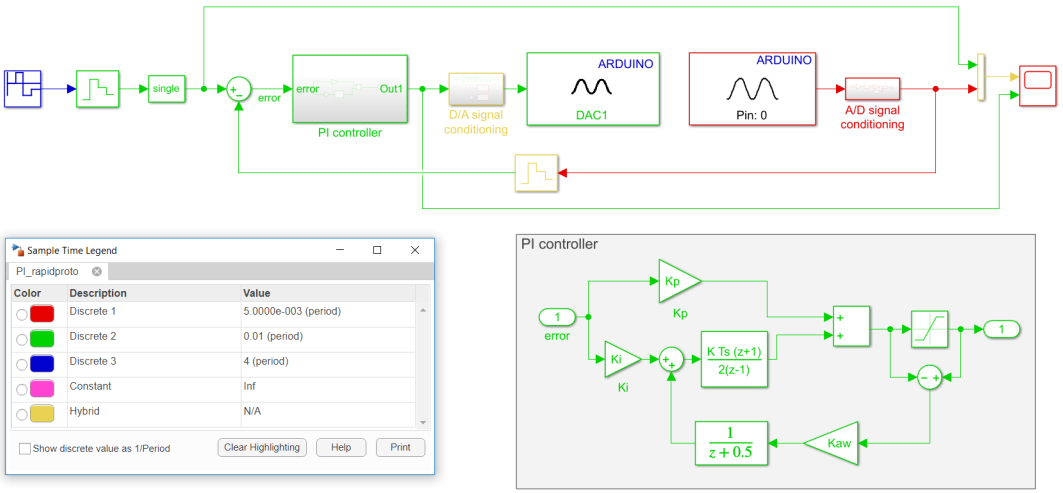


Figure 4.36: Simulink block diagram for the PI control.

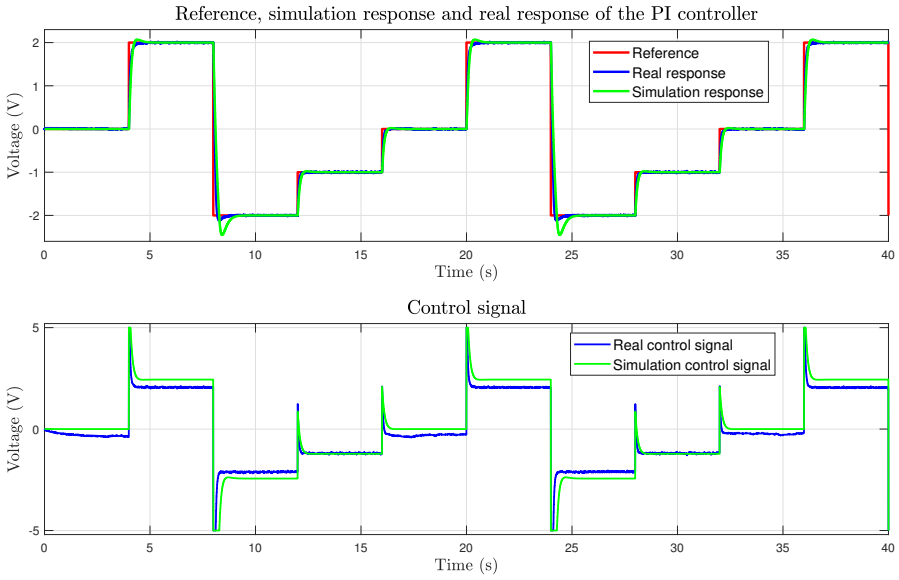


Figure 4.37: Simulation and real response of the PI control system.

Design and implementation of a PID controller by pole placement.

The PID controller will be designed to control the position shaft. As we have seen on workpackage 1, the system in open-loop resembles a second order system, so the controller that best suits to it, is a PID controller. In this part, we want the closed-loop system to have an oscillation frequency of $0,5Hz$ and an 80 % of overshoot.

The block diagram that corresponds to this exercise is shown in Figure 4.38.

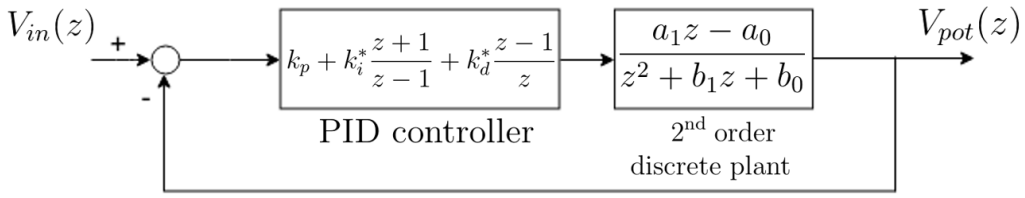


Figure 4.38: Block diagram of the PID control.

In this way, the closed-loop transfer function stands as follows:

$$CL(z) = \frac{\text{PID}(z)\overline{G}(z)}{1 + \text{PID}(z)\overline{G}(z)} = \frac{\text{Num}(z)}{\text{Den}(z)} \tag{4.21}$$

Where:

$$\begin{aligned}
\text{Num}(z) &= a_1\left(k_p + k_i^* + k_d^*\right) z^3 + \left[a_1\left(k_i^* - k_p - 2k_d^*\right) + a_0\left(k_p + k_i^* + k_d^*\right)\right] z^2 \\
&\quad + \left[a_1 k_d^* + a_0\left(k_i^* - k_p - 2k_d^*\right)\right] z + a_0 k_d^* \\
\text{Den}(z) &= z^4 + \left(-1 + k_i^* a_1 + b_1 + k_d^* a_1 + k_p a_1\right) z^3 \\
&\quad + \left(-b_1 - k_p a_1 + k_p a_0 + b_0 - 2k_d^* a_1 + k_i^* a_0 + k_i^* a_1 + k_d^* a_0\right) z^2 \\
&\quad + \left(k_i^* a_0 - b_0 - 2k_d^* a_0 - k_p a_0 + k_d^* a_1\right) z + k_d^* a_0
\end{aligned} \tag{4.22}$$

As we can see, the closed-loop system is a fourth order system, which means the desired characteristic equation will have 4 poles. As we only have three parameters to fix four poles, one of the them will be fixed freely (p_4), so we will have to check whether this pole is dominant or not. The approach will be accepted if this freely fixed pole is stable and not dominant.

The characteristic equation for the this case is the following

$$\begin{aligned}\mathrm{Den}(z) &= (z-p_1)(z-p_2)(z-p_3)(z-p_4) = z^4 + (-p_1-p_2-p_3-p_4)\,z^3 \\ &+ (p_1p_2 - (-p_1-p_2)\,p_3 - (-p_1-p_2-p_3)\,p_4)\,z^2 \\ &+ (-p_1p_2p_3 - (-p_1p_2 - (-p_1-p_2)\,p_3)\,p_4)\,z + p_1p_2p_3p_4\end{aligned} \tag{4.23}$$

As in the case of the velocity control, the coefficient of the denominator of the closed-loop system has been matched by a desired characteristic equation, where the desired specifications are embedded. This linear equation represented in matrix form obtained when matching the two equations has been solved using MATLAB, from which the tuning equations are obtained.

$$\begin{bmatrix} -1 & -a_1 & -a_1 & -a_1 \\ p_1+p_2+p_3 & a_1-a_0 & -a_1-a_0 & -a_0-2a_1 \\ -p_1p_2-p_3p_1-p_3p_2 & a_0 & -a_0 & 2a_0-a_1 \\ p_1p_2p_3 & 0 & 0 & -a_0 \end{bmatrix} \begin{bmatrix} p_4 \\ k_p \\ k_i^* \\ k_d^* \end{bmatrix} = \begin{bmatrix} p_1+p_2+p_3-1+b_1 \\ -p_1p_2-p_3p_1-p_3p_2+b_0-b_1 \\ p_1p_2p_3-b_0 \\ 0 \end{bmatrix}$$

For the PID implementation, the model shown in Figure 4.39 has been designed.

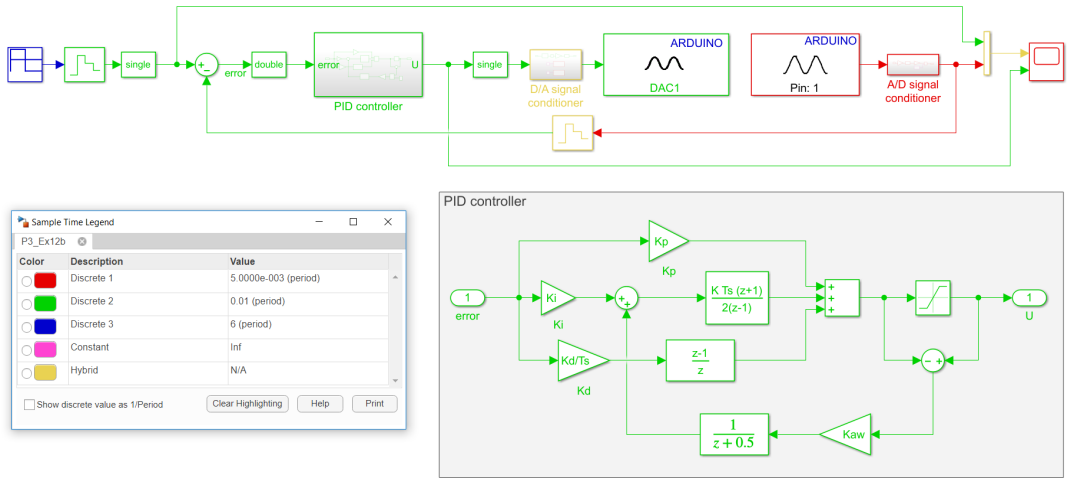


Figure 4.39: Simulink block diagram for the PID control.

Figure 4.40 shows the responses of a simulation done before implementing the controller in the real system, in green, and the real response, in blue. On one hand, as it can be analyzed, the simulation with the PID controller doesn't follow exactly the desired dynamics, which comes from the fact that the closed loop system presents three zeros which have negative effects on the dynamics and because a pole has been fixed freely, which is quite dominant. On the other hand, the real response doesn't follow the simulated response, this is because there are non modelled dynamics and so the obtained transfer function doesn't exactly represent the whole real system. In the next practice we

will analyze how to delete the effects of the zeros and the free pole and so, improve the system's response.

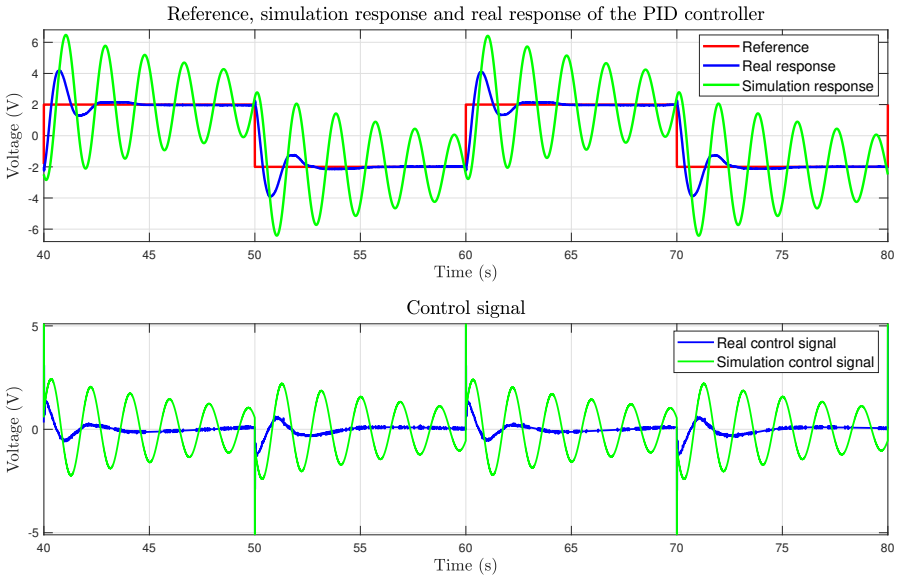


Figure 4.40: Simulation and real response of the PID control system.

4.5 Workpackage 4: Improvement of the PID controllers implemented in practice 3

In this forth workpackage, we are going to see how we can rearrange the PID controllers implemented in the third practice so we can obtain the desired results deleting the effects of having zeros or free poles on the closed-loop system. For that, a different structure to introduce the same control actions will be proposed and the results will be analyzed.

Design and implementation of a I-P controller by pole placement.

As we saw on practice 3, the obtained closed-loop transfer function of the PI velocity control had 2 poles and a zero. By means of the controller's parameters we were able to place the poles and design the stability and the temporal response of the system. But placing the poles by the controller's parameters in this architecture has a drawback. Going back to the closed-loop transfer function obtained when closing the control loop, see formula 4.19, the zero is also automatically placed when placing the poles, so we obtained an undesired overshoot as we could see on Figure 4.37. We designed the PI controller such that we stabilize the plant in 0.8 seconds with no overshoot, but the obtained system didn't fulfill the specifications.

The zero of the PI control loop is located on:

$$z = \frac{-k_i^* + k_p}{k_i^* + k_p} \tag{4.24}$$

As we can see, its locations depends on the controller's gains.

The following figure shows another alternative manner to introduce the proportional and integral actions. This control structure takes the name of I-P controller. As we can analyze, the denominator of the obtained transfer function is the same as in the case of the PI controller, what it has changed is the numerator of this new scheme.

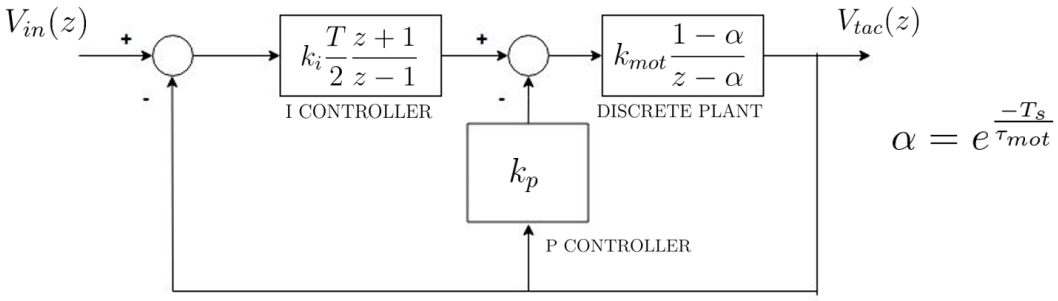


Figure 4.41: Block diagram of the I-P control.

As we can see, form the obtained closed-loop transfer function, see equation 4.25 now we obtain a zero in $z = -1$, which does not depend on the controller gains. It is worth to comment that the proportional action of the I-P controller doesn't see the reference, so the response of the closed-loop system to a step input of a I-P controller is smoother than the one with the PI.

$$CL_{IP}(z) = \frac{k_{mot} k_i^* (z+1)(1-\alpha)}{(z-1)(z-\alpha) + k_{mot}\left(k_p + k_i^*\right)\left(z + \frac{k_i^* - k_p}{k_i^* + k_p}\right)(1-\alpha)} \tag{4.25}$$

On the way, the denominator of the closed-loop of the I-P control is the same as the one obtained in the practice 3 with the PI control, so, the equations used to tune the controller parameters are exactly the same, than those used for the PI controller, see equations 4.21.

On Figure 4.42 we can analyze the responses of both simulations, the blue response corresponds to the PI controller, while the green one corresponds to the I-P controller. As it can be seen, I-P controller allows to obtain the desired dynamics without the additional overshoot.

Finally, Figure 4.43 shows the Simulink block diagram designed for the I-P control system and Figure 4.44 shows the responses of a simulation done before implementing the I-P controller in the real system, in blue, and the real response, in green. As can be

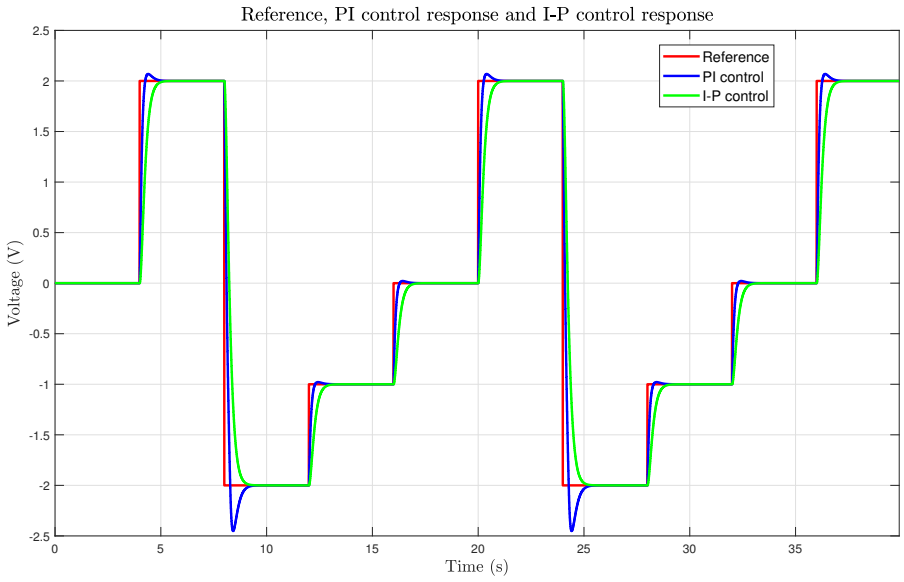


Figure 4.42: Comparison of the PI and I-P control systems.

observed, the two responses are practically the same, and so we can say that specifications are met.

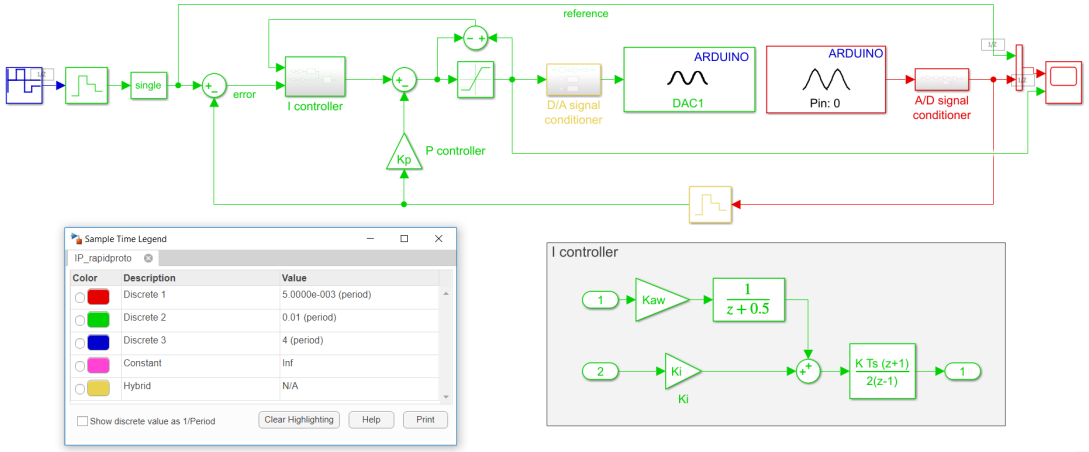


Figure 4.43: Simulink block diagram for the I-P control.

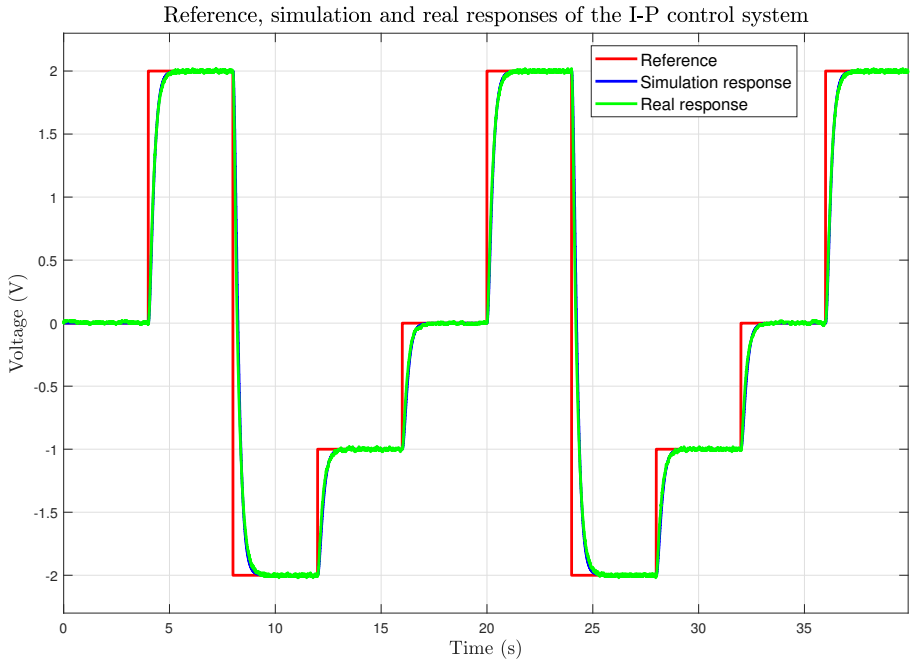


Figure 4.44: Simulation and real response of the I-P control system.

Design and implementation of a I-PD controller by pole placement.

In the case of the position control, we obtained a closed-loop system with 4 poles and 3 zeros. Two main problems where identified when closing the closed loop system. On one hand, we have 4 poles to be fixed by three parameters, this means, one of the poles was freely fixed and we had to check whether this pole was stable and non dominant in order to validate the control problem. On the other hand, we obtained 3 zeros, which affect the dynamics of the closed-loop system. In order to obtain the desired dynamics, those two problems have to be debugged.

If we analyze the 3 zeros obtained in the PID control problem, we can see that one zero is the plant's zero and the other two are imposed by the controller. As in the case of the velocity control, when placing the closed-loop poles, those zeros are automatically fixed, as they depend on the controller's gains too, see equations 4.26. So, we will modify the structure of the controller to delete the effect of the zeros.

$$\begin{aligned} z_{1,2} &= \frac{-\left(k_i^* - k_p - 2k_d^*\right) \pm \sqrt{-4k_d^*\left(k_p + k_i^* + k_d^*\right) + \left(k_i^* - k_p - 2k_d^*\right)^2}}{2 \cdot \left(k_p + k_i^* + k_d^*\right)} \\ z_3 &= \frac{a_0}{a_1} \end{aligned} \tag{4.26}$$

The proposed structure is shown in Figure 4.45. In this new structure, the three actions of the controller are embedded. As we have done before, the input of the integral action is the error, while the input of the proportional and derivative actions is the output

of the system. This control structure takes the name of I-PD controller.

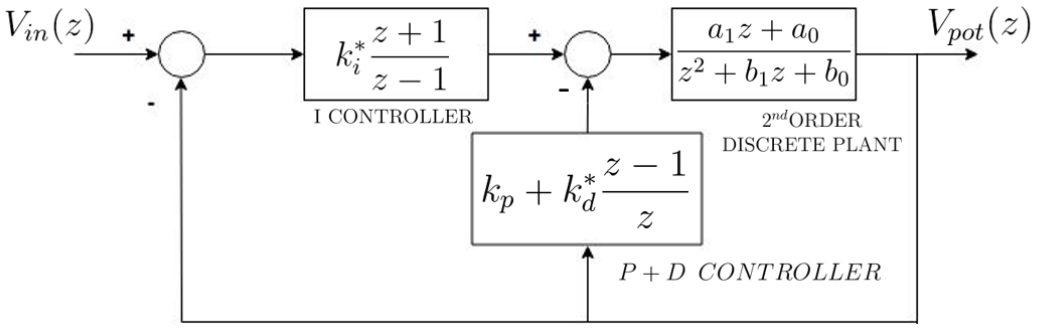


Figure 4.45: Block diagram of the I-PD control.

In this way, the closed-loop transfer function stands as follows:

$$CL(z) = \frac{\text{I-PD}(z)\overline{G}(z)}{1 + \text{I-PD}(z)\overline{G}(z)} = \frac{\text{Num}(z)}{\text{Den}(z)} \tag{4.27}$$

Where:

$$\begin{aligned} \text{Num}(z) &= k_i^* (z+1)(a_1 z + a_0) z \\ \text{Den}(z) &= z^4 + (-1 + k_i^* a_1 + b_1 + k_d^* a_1 + k_p a_1) z^3 \\ &\quad + (-b_1 - k_p a_1 + k_p a_0 + b_0 - 2k_d^* a_1 + k_i^* a_0 + k_i^* a_1 + k_d^* a_0) z^2 \\ &\quad + (k_i^* a_0 - b_0 - 2k_d^* a_0 - k_p a_0 + k_d^* a_1) z + k_d^* a_0 \end{aligned} \tag{4.28}$$

As we can see, the denominator is the same as the one obtained in equation 4.23. What has changed is the numerator, now, we can see the zeros are not dependent on the controller's gains.

$$\begin{aligned} z_1 &= -1 \\ z_2 &= 0 \\ z_3 &= \frac{a_0}{a_1} \end{aligned} \tag{4.29}$$

Figure 4.46 shows the comparison of the simulation of the system with the controllers mentioned above. As we can see, the response is smoother in the I-PD case. As it happened in the case before, that comes from the fact that the proportional action doesn't see the reference.

But if we analyze the specifications, we can see those desired specifications are not fulfilled. As said at the begging of this part, that comes from the fact that a pole has

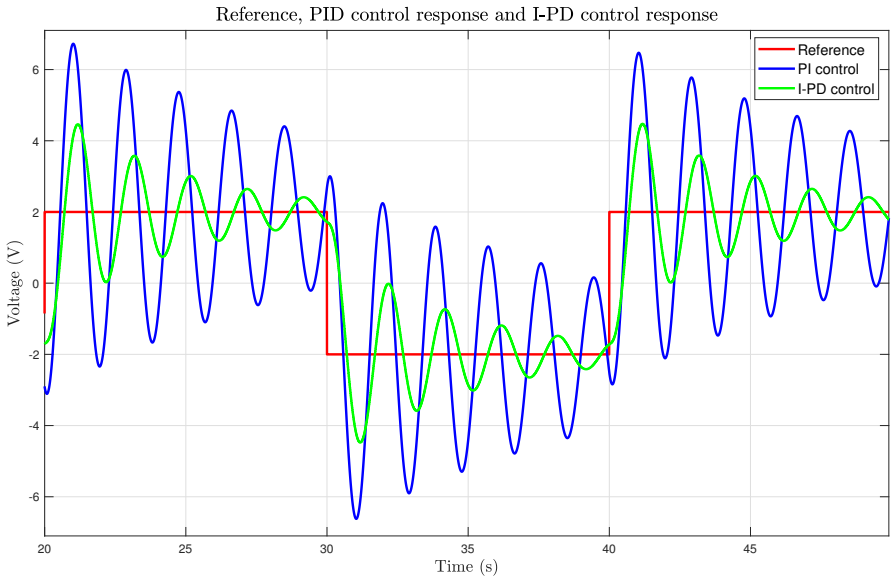


Figure 4.46: Comparison of the PID and I-PD control systems.

been freely fixed. Although the control problem has been validated because the free pole was stable and non dominant, we can conclude it is quite dominant and the closed-loop dynamics remains affected.

Figure 4.47 shows the response of the I-PD controller in the LJ Technical System's servo system.

Figure 4.47: Simulation and real response of the I-PD control systems.

For its implementation the Simulink model shown in Figure 4.48 has been designed.

In order to delete the effect of the fourth pole, an additional parameter is introduced, α, so we will be able to fix the four poles by the four parameters. To this new structure

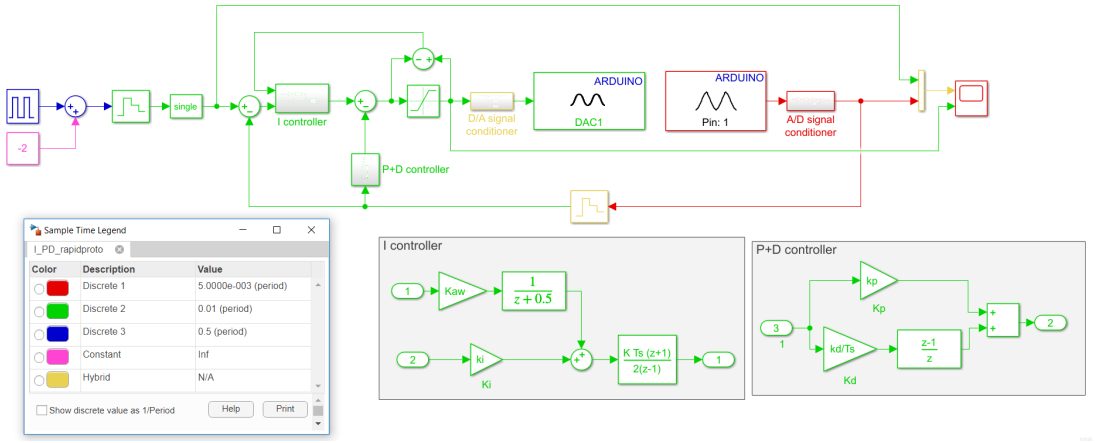


Figure 4.48: Simulink block diagram for the I-PD control.

we will name it I-PDα controller.

Design and implementation of a I-PDα controller by pole placement.

As aforementioned, although we have debugged the problem of the zeros, we still have three parameters to fix four poles, so, a new parameter is added, α. This parameter is added on the derivative part as it is represented on Figure 4.49. As told before, from now on, in order to refer to this new structure, we will name it I-PDα controller.

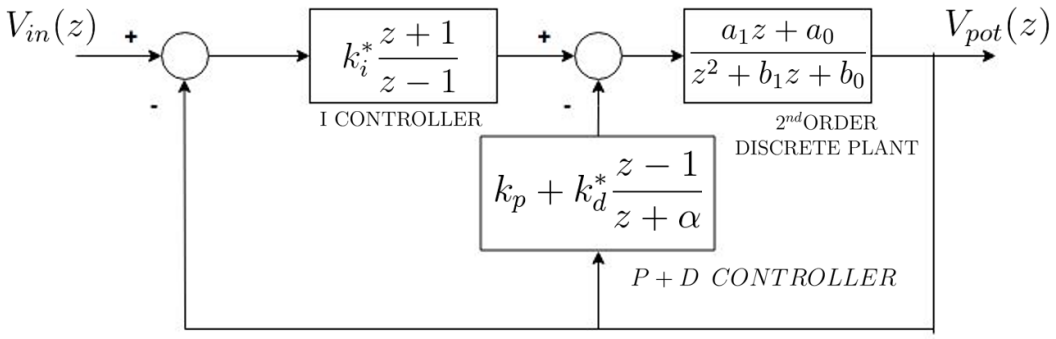


Figure 4.49: Block diagram of the I-PDα control.

First of all, lets analyze the closed-loop transfer function, so we can see where the zeros are located.

$$CL(z) = \frac{\text{I-PD}\alpha(z)\overline{G}(z)}{1 + \text{I-PD}\alpha(z)\overline{G}(z)} = \frac{\text{Num}(z)}{\text{Den}(z)} \tag{4.30}$$

Where:

$$\begin{aligned}
\text{Num}(z) &= k_i^* (z+1)(a_1 z + a_0)(z+\alpha) \\
\text{Den}(z) &= z^4 + (b_1 - 1 + \alpha + k_p a_1 + k_i^* a_1 + k_d^* a_1) z^3 \\
&+ (b_0 - b_1 + b_1\alpha - \alpha + k_p a_0 + k_p a_1(\alpha - 1) + k_i^* a_0 + k_i^* a_1(\alpha + 1) + k_d^* a_0 - 2k_d^* a_1) z^2 \\
&+ (-b_0 + b_0\alpha - b_1\alpha + k_p a_0(\alpha - 1) - k_p a_1\alpha + k_i^* a_0(\alpha + 1) + k_i^* a_1\alpha - 2k_d^* a_0 + k_d^* a_1) z \\
&+ k_d^* a_0 + k_i^* a_0\alpha - k_p a_0\alpha - b_0\alpha
\end{aligned} \tag{4.31}$$

As we can see, we still have three zeros, but they are not dependent on the controller's gains.

$$\begin{aligned}
z_1 &= -1 \\
z_2 &= -\alpha \\
z_3 &= \frac{a_0}{a_1}
\end{aligned} \tag{4.32}$$

Figure 4.50 shows the simulation of the PID, I-PD and I-PDα control systems.

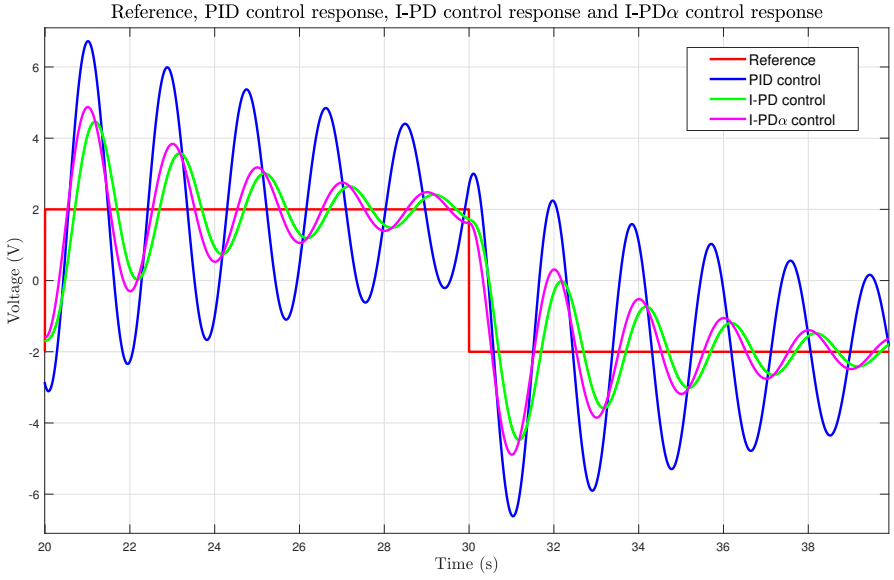


Figure 4.50: Comparison of the PID, I-PD and I-PDα control systems.

In this last case, the specifications are met. The main drawback of this last configuration is that we have to be careful with the parameter α, as if this becomes positive, the filter attached to the derivative gain will became a high-pass filter, so the system will became unstable. In this case, for this configuration, if we place the poles p_3 and p_4 in a very fast place (0.01 for example) the α becomes positive and so the system in closed-loop unstable. In order to avoid this effect the poles p_3 and p_4 have to be slowed down (in our

case 0.8 both poles). Figure 4.51 shows the difference between the ideal I-PDα and the one implemented in the system with slower poles.

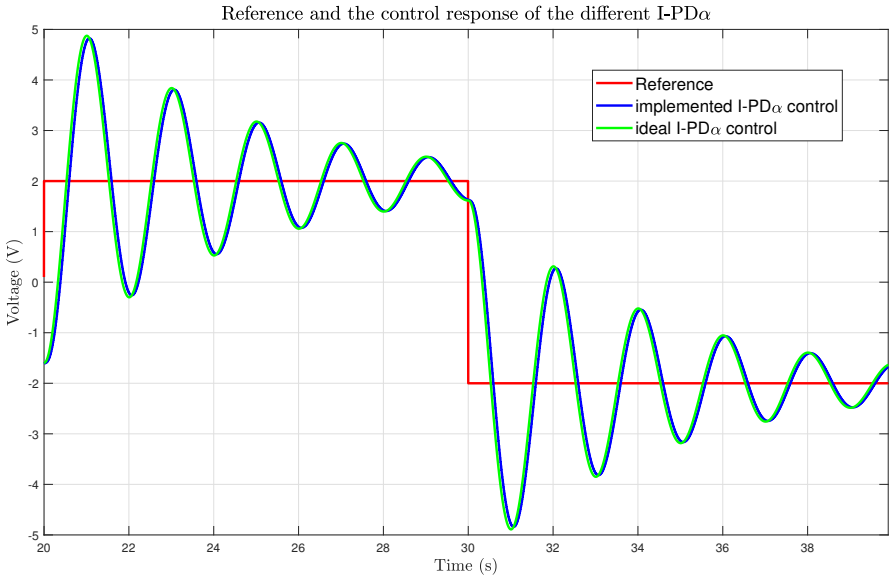


Figure 4.51: Comparison of the ideal I-PDα and the implemented I-PDα control systems.

As it can bee seen the implemented one its a bit slower, but the results are qualitatively equal. In order to conclude the workpackage 4, Figure 4.52 shows the differences between the three experiments carried out with the controllers mentioned.

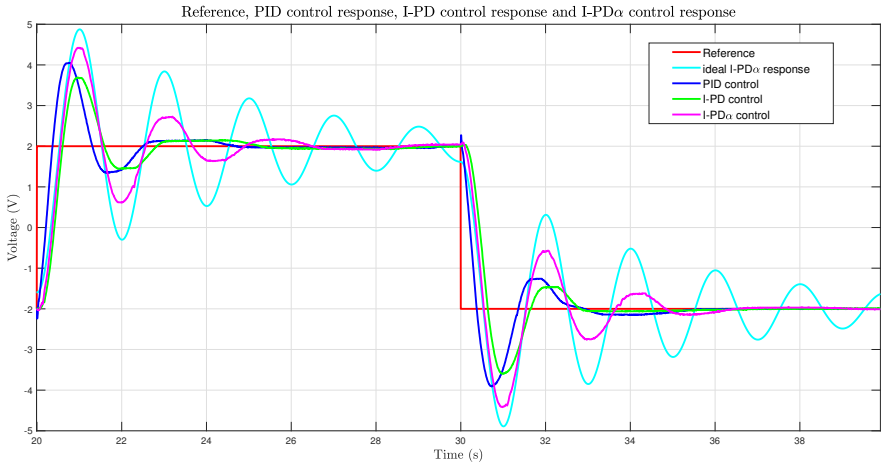


Figure 4.52: Experimental comparison of the PID, I-PD and I-PDα control systems.

The response of cyan color is the response that met the specifications perfectly (simulation), the one with the fast poles. We can see that the one that best suits this response is the one obtained with the I-PDα controller. As in the cases before, there are non modelled dynamics that haven't been taken into account.

Figure 4.53 shows the Simulink block diagram designed for the I-PDα control system.

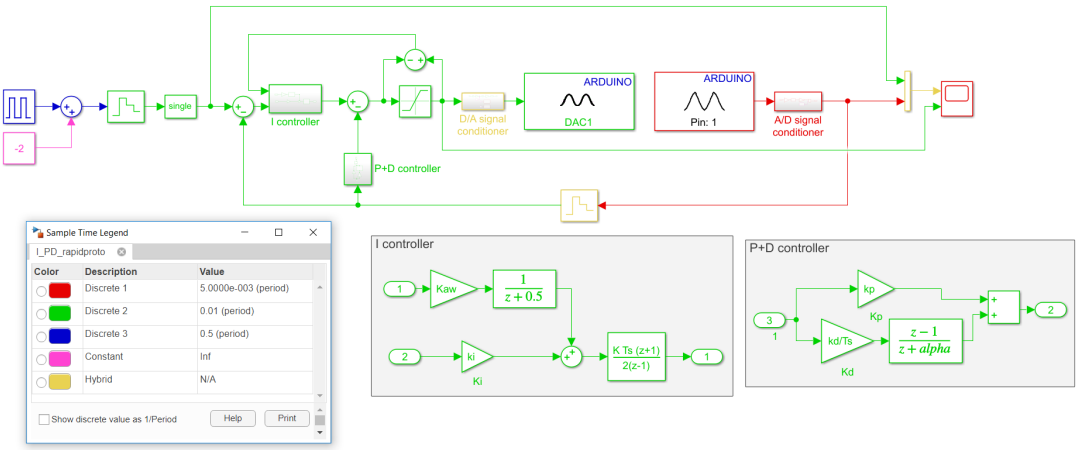


Figure 4.53: Simulink block diagram for the I-PDα control.

4.6 Workpackage 5: Design of controllers in the frequency domain

In this fifth workpackage, we are going to see how to design a controller in the frequency domain, so we can modify the behavior of the system in closed-loop. The design of controllers in the frequency domain is one of the most used in practice because it allows to get designs in a simple way. In many occasions, the design in the frequency field allows to reach the specifications using low order controllers.

There are three kind of compensations that can be implemented on the systems. If we want to modify the transient behavior, we will design an advance compensation while the improvement in the accuracy of the steady state is done with a delay compensation. Both controllers raises the system order in one. By combining both compensations, we can improve both characteristics at the same time, but the main drawback of this last is that it raises the system's order in two, which give as a result a more complex control system.

The chart in Figure 4.54 shows the procedure that has to be followed in order to design a controller in the frequency domain. As it can been, first, having the digital transfer function of the plant, the bilinear transformation is applied, so we obtain the plant transfer function in terms of ω. It is here were the desired specifications are introduced, so we can design a controller to meet those specifications. Finally, the inverse bilinear transformation is applied so we can obtain the digital representation of the controller designed in the frequency domain [22].

In this practice, the phase advance and phase delay controllers are presented. These controllers are set to be in series with the plant to be controlled and aim to modify the behavior of the system in closed-loop. The control system must satisfy certain specifications defined in the frequency domain by means of a gain margin or a phase margin.

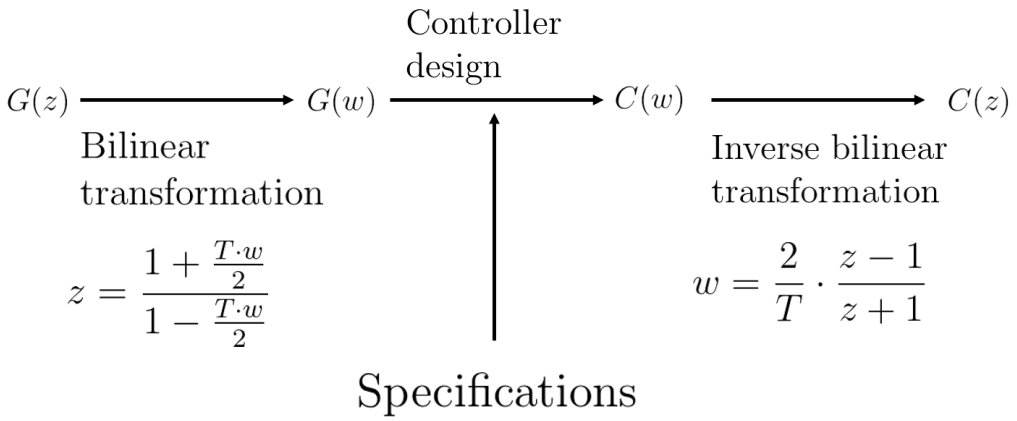


Figure 4.54: Steps to follow when implementing a controller in the frequency domain.

For the design of controllers in the frequency domain, as seeing in the Figure before, the discrete time systems are transformed into similar systems of continuous time and are then is antitransformed to have a digital controller. As it can bee seen in Figure 4.33, the generic transfer function in s of an advance or phase delay controller is given by Gc.

$$G_c(s) = K \frac{\frac{s}{\omega_z} + 1}{\frac{s}{\omega_p} + 1} \tag{4.33}$$

Where ω_z is the zero's frequency and ω_p is the pole's frequency.

Phase advance controllers.

In order to have a positive phase advance, the pole's frequency must be higher than the zero's frequency. This makes the controller to be a phase advance controller, so in a certain range of frequencies, they will increase, or advance, the phase of the controlled system.

The design of the controller is done by solving a system of equations, but as usually this is quite hard to implement the design is done by looking for an approximate solution of that system and finally verifying that the closed loop system satisfies the specifications. It is a method of solution by trial and error. As it can bee seen on equation 4.34, since it is an approximate solution, an additional margin of security is introduced.

$$\begin{aligned} \theta_c &= \Phi_{md} + \Delta - \Phi_{GH} \\ \alpha &= \tfrac{1+\sin\theta_c}{1-\sin\theta_c} \\ |GH\,(j\omega_c)| &= \tfrac{1}{\sqrt{\alpha}} \end{aligned} \tag{4.34}$$

Where Φ_{GH} is the phase margin of the open-loop system, Δ is the additional security margin, θ_c is the maximum phase advance provided by the controller, α is the length of

the controller and ω_c is the central frequency.

Phase delay controllers.

In order to have a negative phase advance, the pole's frequency must be smaller than the zero's frequency. This makes the controller to be a phase delay controller, so in a certain frequency interval, the controller reduces, or delays, the phase of the controlled system. In this case too, the controller is done by looking for an approximate solution of a system of equations and finally verifying that the closed loop system satisfies the specifications.

$$\begin{aligned} \theta_c &= \gamma_d + \Delta - 180 \\ \arg\left(GH\left(j\omega_c\right)\right) &= \theta_c \\ \alpha &= \left|GH\left(j\omega_c\right)\right| \\ \omega_z &= \frac{\omega_c}{10} \\ \omega_p &= \frac{\omega_z}{\alpha} \end{aligned} \tag{4.35}$$

In this practice, a phase advance controller is designed so that it makes the closed-loop system, for the control of the shaft position of the laboratory plant, with a sampling period of 0.01 seconds, have a phase margin of 45 degrees and a static error of the 10% in front of a ramp input. Considering for that a safety phase margin of 15°. For that, first, the maximum phase advance provided by the controller and the length of it is calculated as it can be seen on the image. After that, the controller is constructed and whether the obtained new phase margin is correct is checked. The block diagram implemented for this exercise is shown in Figure 4.55.

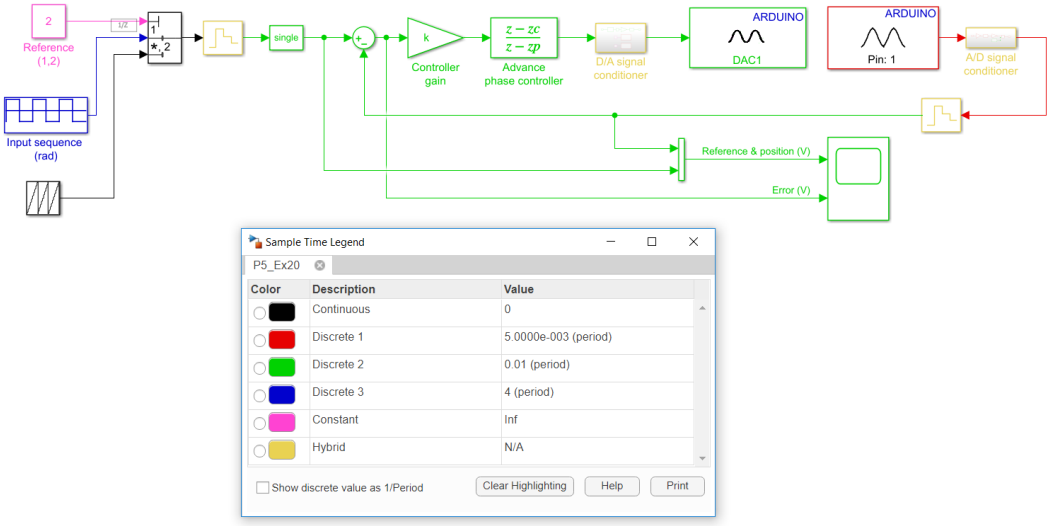


Figure 4.55: Implemented Simulink model for the phase advance controller.

From this point on, we will see, how the advance controller has being designed.

```
% Plants parameters
Kpot=1.62;
N=9;
K0=0.82/0.017;
tau0=0.26;
planta=tf([Kpot*K0/N],[tau0,1,0])
% Discrete time model
Ts=0.01;
plantad=c2d(planta,Ts,'zoh');
% Bilinear transformation
plantaw=d2c(plantad,'tustin') ;
% Velocity error coefficient
errordes=0.1;
Kv=(1/errordes);
% Calculation of the needed gain increase (Controller gain)
Kvplanta=Kpot*K0/N;
Kcontrolador=Kv/Kvplanta
% Phase margin calculation
[Gm,Pm,Wcg,Wcp]=margin(Kcontrolador*plantaw)
% Calculation of the phase that the controller must add
gamma=45;
delta=15;
theta=(gamma+delta-Pm)* pi /180;
% Obtaining alpha parameter
alpha=(1+sin(theta))/(1-sin(theta));
% Search of point where the system has the gain = 1/sqrt(alpha)
[aux1,aux2,aux3,wc]=margin(Kcontrolador*plantaw*sqrt(alpha));
wc
% Pole and zero placement
wz=wc/sqrt(alpha);
wp=wz*alpha ;
% Controller construction
Gcw=tf([1/wz,1],[1/wp,1]);
% Bilinear inverse of the controller
Gcz=c2d(Gcw,Ts,'tustin')
% Closed-loop system
closed=feedback(Gczplantad,1);
err=1-closed;
```

By plotting the the temporary response to a ramp type input, Figure 4.56, we can check that the system represent an error of 0.1.

Finally, we can check the responses of the system to a pulse type input, see Figure 4.57 and to a saw wave type input, see Figure 4.58. The little errors came from the non modelled dynamics.

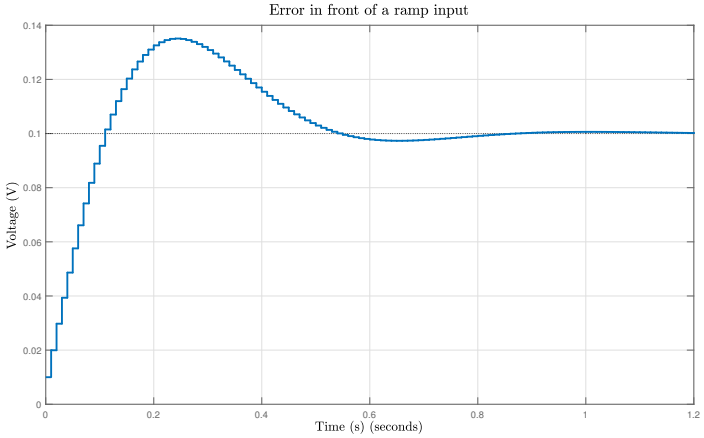


Figure 4.56: Steps to follow when implementing a controller in the frequency domain.

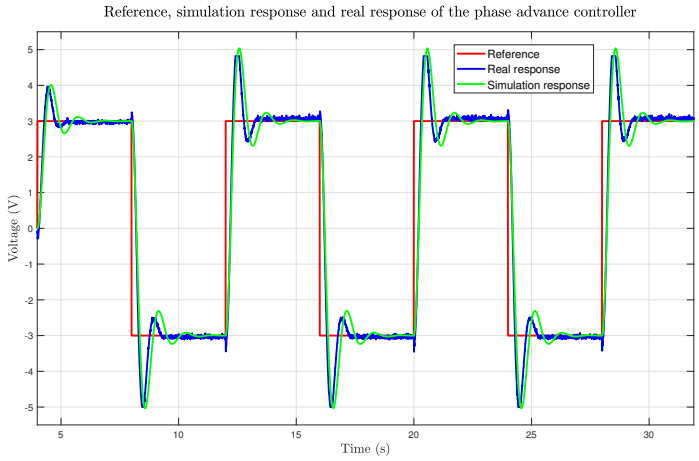


Figure 4.57: System's response to a pulse type input with phase advance controller.

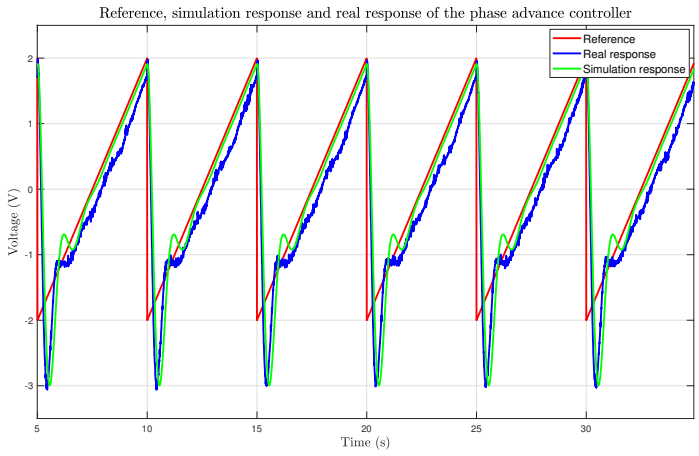
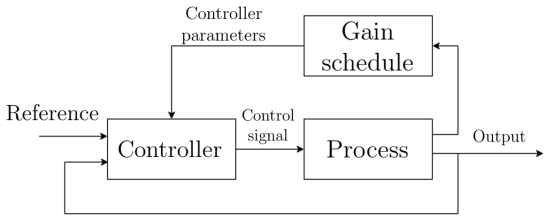


Figure 4.58: System's response to a saw wave type input with phase advance controller.

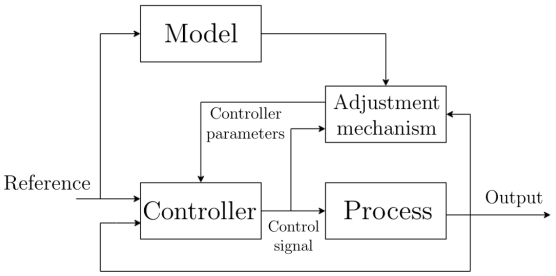
4.7 Workpackage 6: Indirect Adaptive Control

An adaptive controller can be described as a controller that can modify its response behavior when changing the dynamics of the process. So, an adaptive controller can be stated as a controller with adjustable parameters and a mechanism for adjusting the parameters [11].

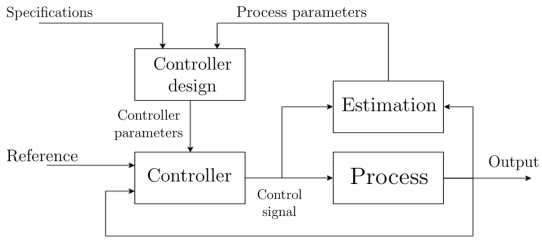
There exist lot of adaptive systems, such as, gain scheduling, model-reference adaptive control, self-tuning regulators and dual control. Figure 4.59 shows their corresponding block diagram schemes.



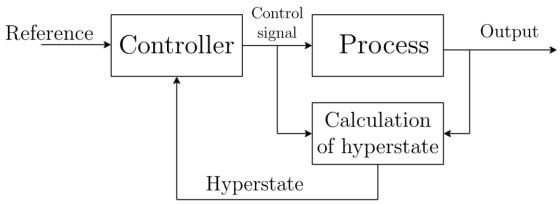
(a) Block diagram of a system with gain scheduling



(b) Block diagram of a model-reference adaptive system (MRAS)



(c) Block diagram of a self-tuning regulator (STR)



(d) Block diagram of a dual controller

Figure 4.59: Block diagram of the different adaptive systems.

Each of them has their pros and cons and depending on the control problem, one can suit better than another. In this project, we will develop a self-tuning regulator.

In addition to this, every one has a design procedure to satisfy some specifications. Here, we can also split the problems as direct or indirect adaptive controllers. In direct adaptive control, the control parameters are directly changed without the need of estimate some other intermediate variables. In indirect adaptive method instead, first, the process variables are estimated and in terms of this variables the control variables are updated.

So, in this sixth workpackage, we will design an indirect self-tuning regulator for velocity and position control. The process variables will be estimated by Recursive Least Square (RLS) algorithm.

The algorithm is the following:

$$\begin{aligned} \theta_{est(k)} &= \theta_{\text{ est }(k-1)} + K_{(k)} \cdot \varepsilon_{(k)} \\ \varepsilon_{(k)} &= y_{(k)} - \varphi^T_{(k)} \cdot \theta_{\text{ est }(k-1)} \\ K_{(k)} &= P_{(k-1)} \cdot \varphi_{(k)} \cdot \left[\lambda \cdot Id + \varphi^T_{(k)} \cdot P_{(k-1)} \cdot \varphi_{(k)}\right]^{-1} \\ P_{(k)} &= \left[Id - K_{(k)} \cdot \varphi^T_{(k)}\right] \cdot P_{(k-1)}/\lambda \end{aligned} \tag{4.36}$$

Where y is the output, θ_{est} is the vector of the estimated process' parameters, φ is the regression vector, λ is the forgetting factor, ε is the estimation error and $K_{(k)}$ and $P_{(k)}$ are the correction gain matrices.

Influence of λ

λ or the forgetting factor, has a very high influence when estimating the parameters. The higher the parameter, the smother the parameter estimation, since K goes to zero. When $\lambda < 1$, the estimator gain K will not go to zero and the estimates will fluctuate. The fluctuations increase when decreasing λ. The estimator's memory is calculated as:

$$N = \frac{2}{1-\lambda} \tag{4.37}$$

For $\lambda = 0.99$ the estimates are based on approximately the last 200 steps. That's why, in all the models, we have chosen $\lambda = 1$. Figure 4.60 shows the parameter estimation with different λ.

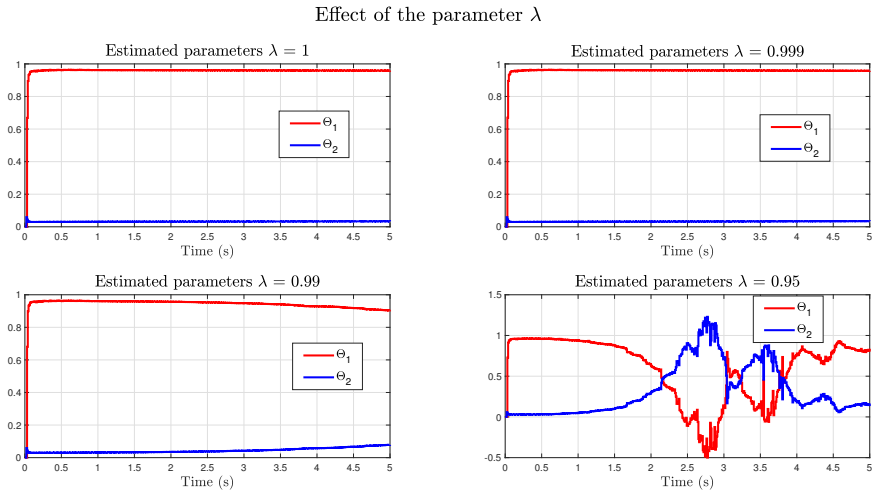


Figure 4.60: Effect of the parameter λ on the system's velocity control parameter estimation.

Influence of P

The matrix P is defined only when the matrix $\Phi^T_{(k)}\Phi_{(k)}$ is non singular. Since

$$\Phi_{(k)}^T \Phi_{(k)} = \sum_{i=1}^{k} \varphi_{(i)} \varphi_{(i)}^T \tag{4.38}$$

and in the recursive method P is initialized by some initial condition the equation for it stand as

$$P_{(K)} = \left(P_0^{-1} + \Phi_{(k)}^T \Phi_{(k)} \right)^{-1} \tag{4.39}$$

Where $P_{(K)}$ can be made arbitrarily close to $\Phi_{(k)}^T \Phi_{(k)}$ by choosing P_0 sufficiently large. That's why, we will initialize it to $P_0 = 1e^4 \cdot I$ where I is the identity matrix of order n, where n is the number of parameters to estimate. Figure 4.61 shows the output of the system having different P_0 matrix.

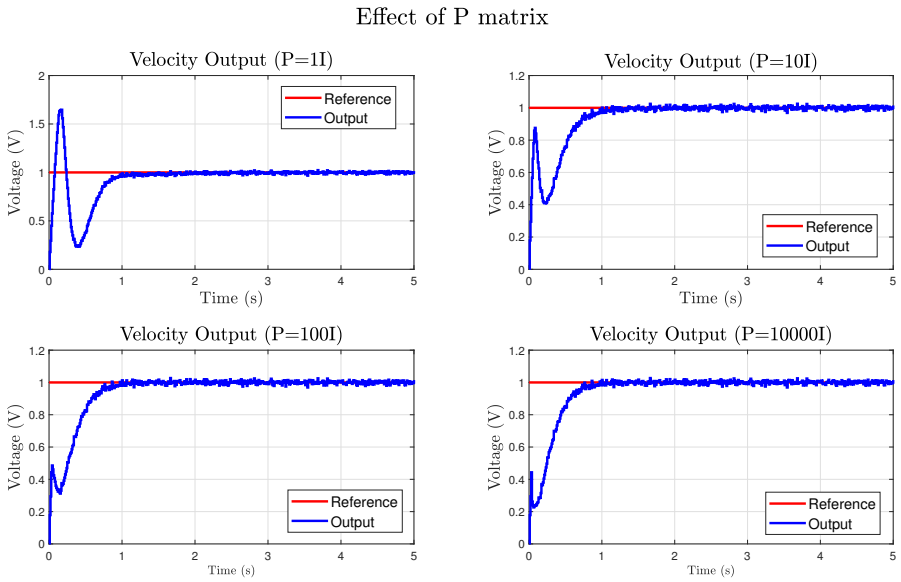


Figure 4.61: Effect of the P_0 matrix on the velocity output of the system.

Indirect Adaptive Velocity Control

For the realization of the adaptive velocity control the Simulink model shown in Figure 4.62 has been designed. As it can be analyzed, the adaptive controller is composed of two loops. The inner loop, or the systemâĂŹs control loop, and the outer loop where the parameters of the regulator are updated by the estimated parameters.

As it can be seen, two main subsystem block are the responsible of estimating the parameters of the plant and to update the controller's gains. Those two subsystems have been coded by block as it can be seen on Figure 4.63.

The RLS algorithm is the one explained on 4.36 and the parameter is updated following the expressions obtained on 4.21.

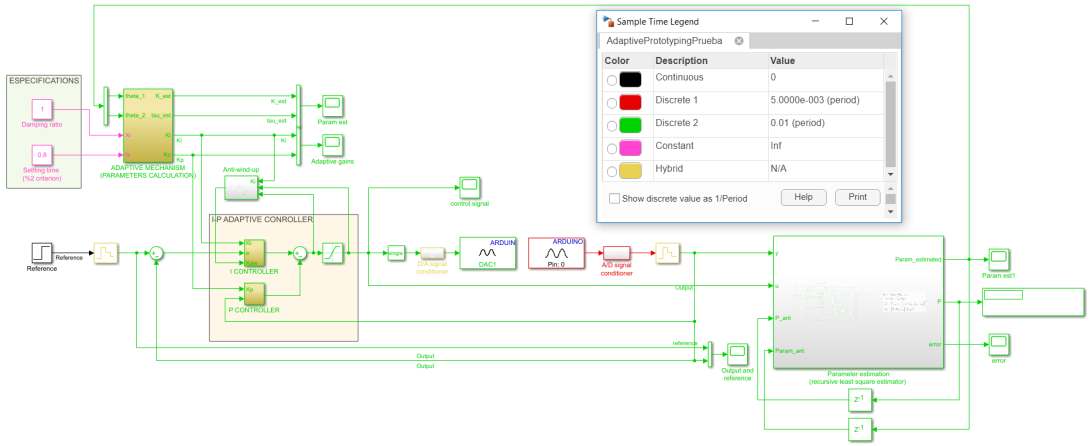
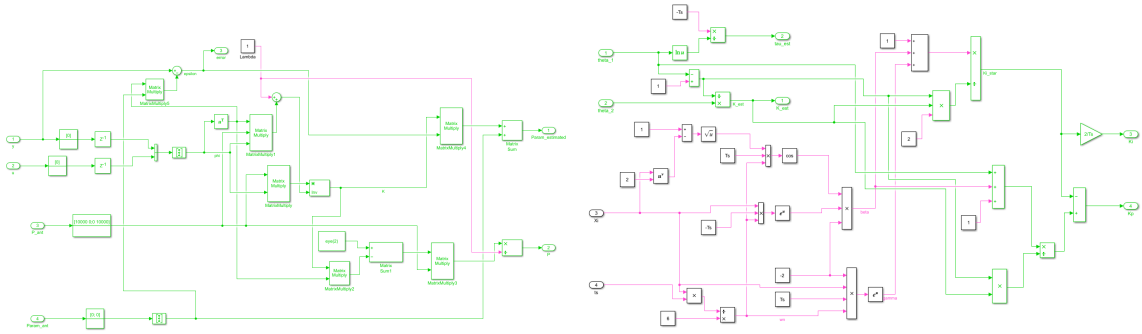


Figure 4.62: Simulink model designed for the indirect adaptive velocity control.



(a) RLS algorithm for the velocity control parameter estimation

(b) I-P controller gain estimation

Figure 4.63: Block diagram for the parameter estimation and controller update.

Figure 4.64, shows the first 5 seconds of an experiment carried out in the model above. There we can see how the parameters are estimated, the gains updated and the output of the system controlled.

Furthermore, an experiment of 30 second will be shown in the following lines. In this experiment the magnetic brake has been applied around the time 10 and disengaged around the time 20.

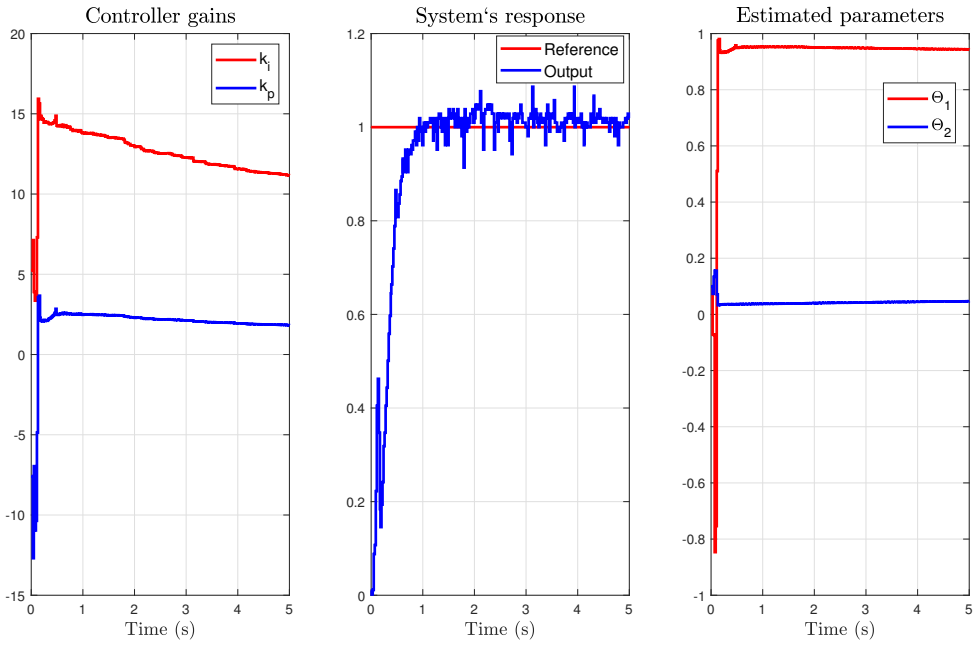


Figure 4.64: Parameter estimation process.

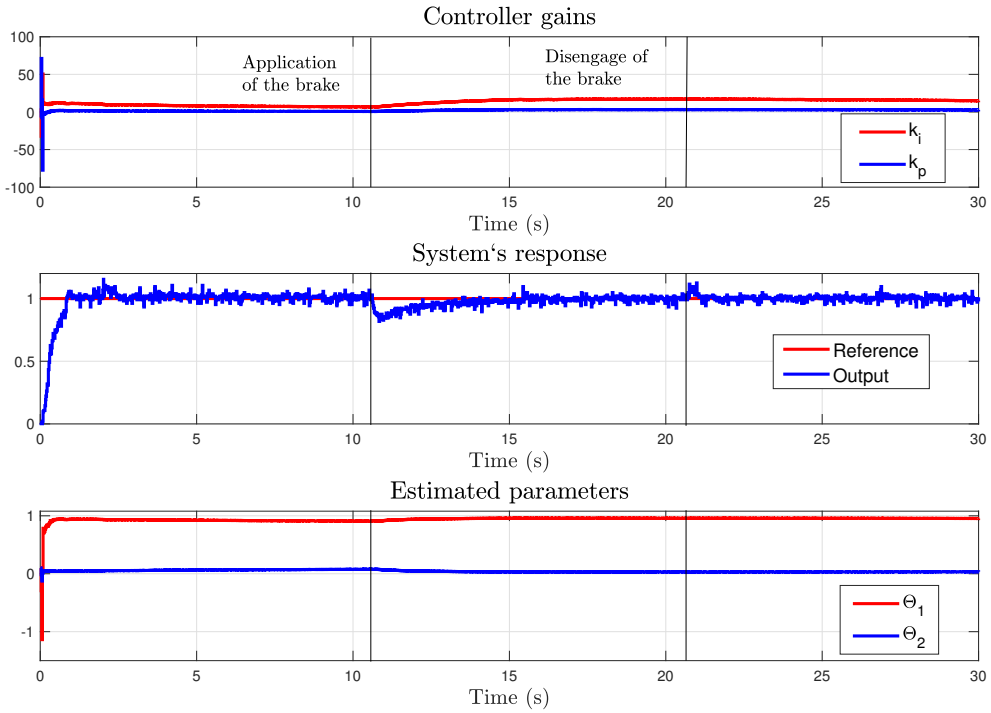


Figure 4.65: Velocity experiment with application and disengage of the magnetic brake.

As we can also analyze on Figure 4.66 the control signal varies depending on the brake position, and the estimation error keeps always 0.

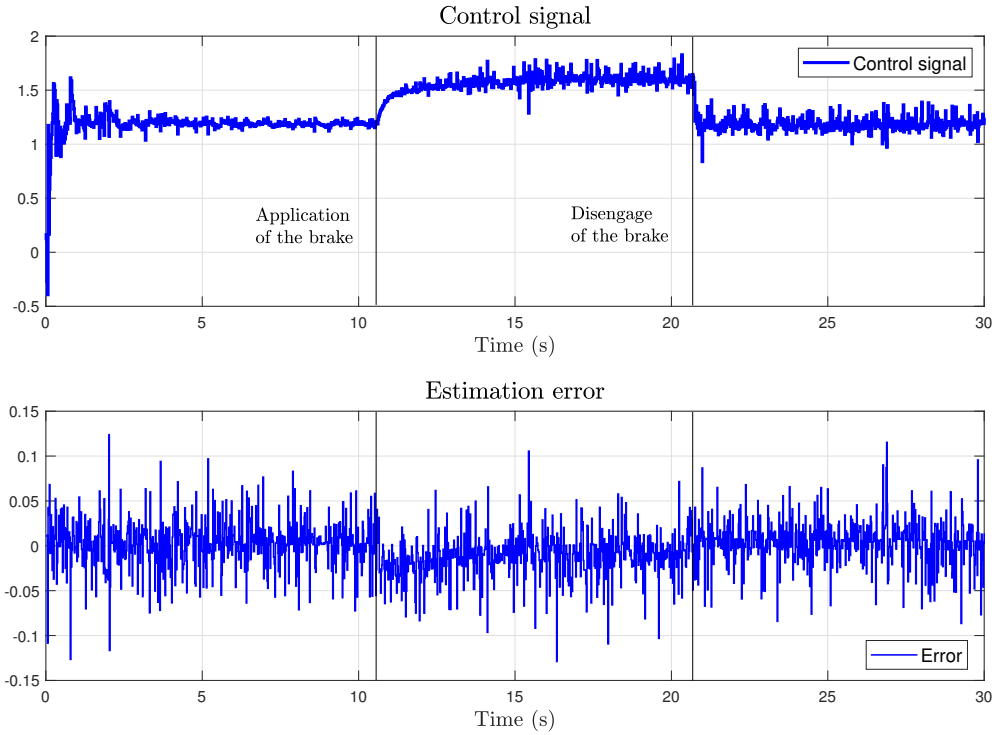


Figure 4.66: Control signal and estimation error of the experiment carried out above.

Indirect Adaptive Position Control

As in workpackage 4, here we will also implement the I-PD and I-PDα controllers and compare their output. Here, as the plant's parameters are estimated online, the specifications will be changed, and a more robust controller will be designed. For that, we will reduce the overshoot in both controllers to 40%, and we will leave the 0.5Hz natural frequency. By this change, the phase and gain margins will increase, leading to a more robust controller, so although the parameter changes may be quite abrupt at first, the controller will be able to control the output.

Adaptive I-PD controller

For the implementation of the I-PD controller, the Simulink model shown in Figure 4.67 has been designed. As we can see, apart from the feedback loop, the parameter estimation and the controller's gains update feedback is implemented. As before, in the first block, the parameters are estimated, while in the second one, the control actions are calculated. in this case, the second block, has been codified on a MATLAB function, see Apendix C.0.2.

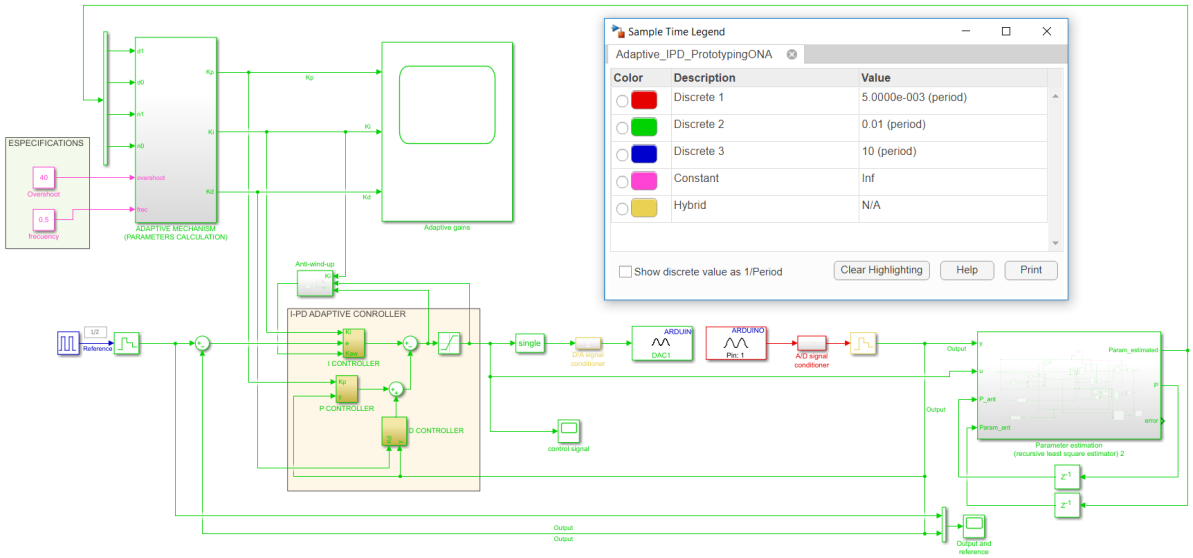


Figure 4.67: Simulink block diagram for the adaptive I-PD control.

After an experiment of 150 seconds, the following results have been obtained. As it can be seen on Figure 4.68, at first the output's dynamics is quite different from the final ones. That comes from the fact, that at first, the estimated parameters are quite far from their final value. On the other hand, the output's overshoot is higher than 40% this effect comes again from the parameter estimation, the higher the system's order, the more difficult is to converge the parameter to their real value, in this case, the parameters converge to a value, but it is quite far from its nominal value, this makes the output not to fulfill the specifications.

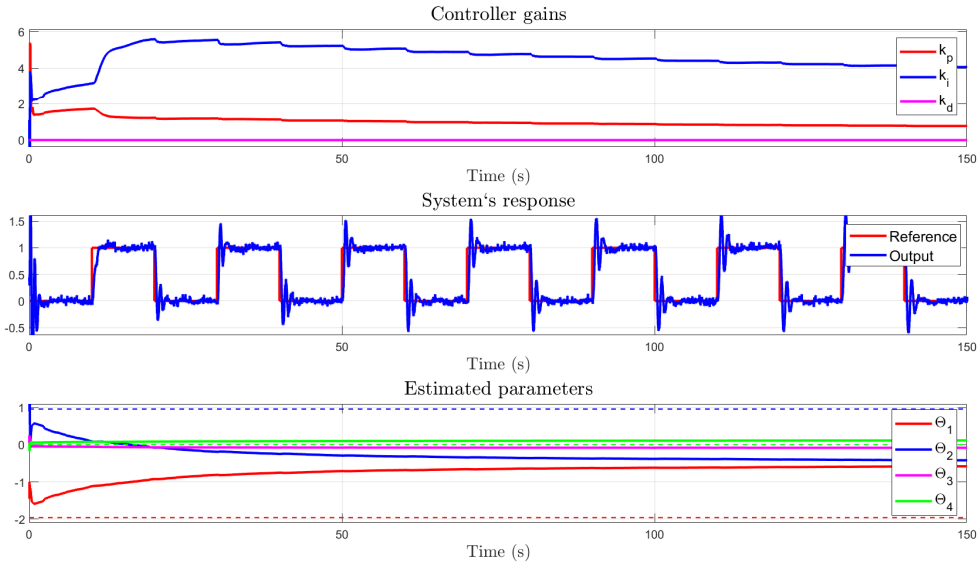
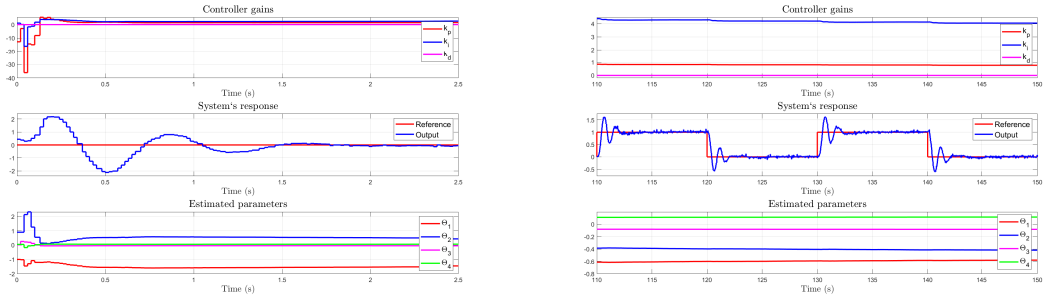


Figure 4.68: I-PD controller experiment.

Figure 4.69 shows the first 2.5 seconds and the last 40 seconds of the experiment

carried out above.



(a) First 2.5 seconds of the experiment. (b) Last 40 seconds of the experiment

Figure 4.69: Initial and final data of the adaptive I-PD.

But if analyze the Figure 4.70 we can see, the system estimates the parameters such that the estimation error tends to 0.

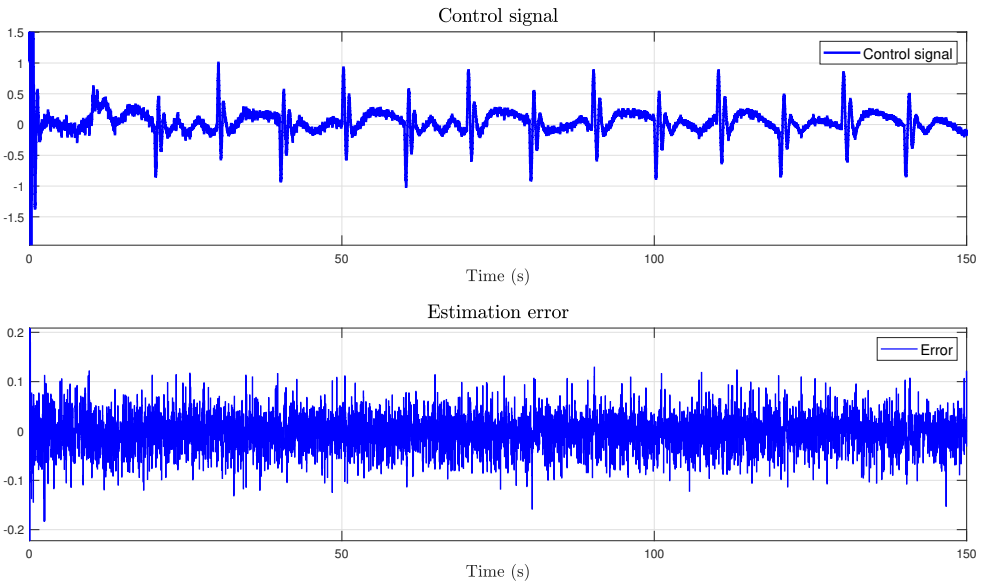


Figure 4.70: Control signal and estimation error of the adaptive I-PD experiment.

Adaptive I-PDα controller

For the implementation of the I-PDα, the Simulink model shown in Figure 4.71 has been designed. In this case too, the control action update block, has been codified on a MATLAB function, which implementation can be found in Appendix C.0.3.

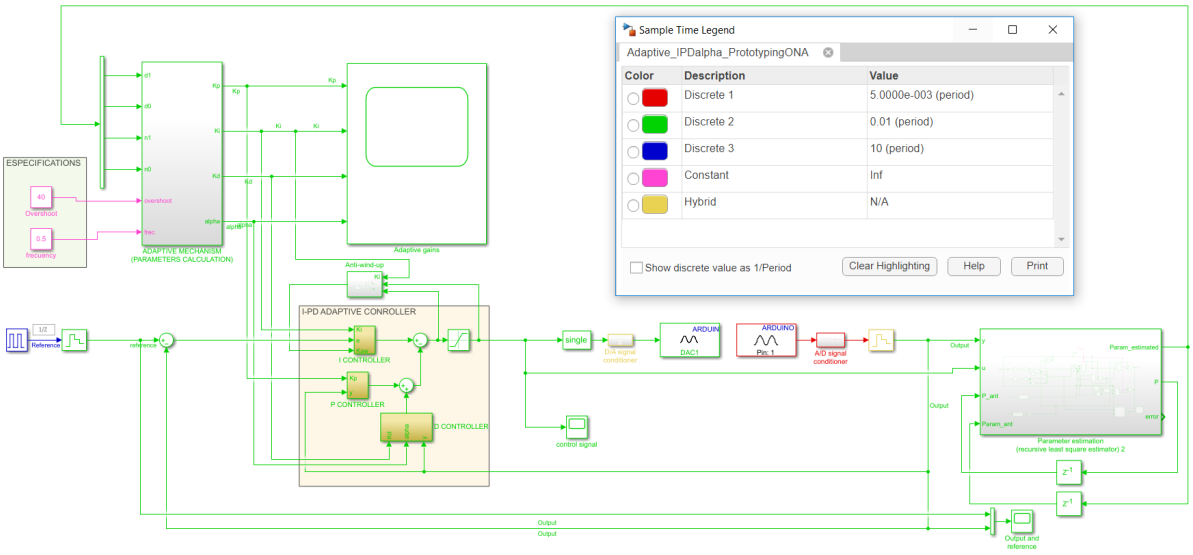


Figure 4.71: Simulink block diagram for the adaptive I-PDα control.

After an experiment of 150 seconds, the following results have been obtained. As it can be seen on Figure 4.72, at first the output's dynamics is quite different from the final ones. That comes from the fact, that at first, the estimated parameters are quite far from their final value. On the other hand, the output's overshoot is higher than 40% this effect comes again from the parameter estimation, the higher the system's order, the more difficult is to converge the parameter to their real value, in this case, the parameters converge to a value, but it is quite far from its nominal value, this makes the output not to fulfill the specifications.

Figure 4.73 shows the first 2.5 seconds and the last 40 seconds of the experiment carried out above.

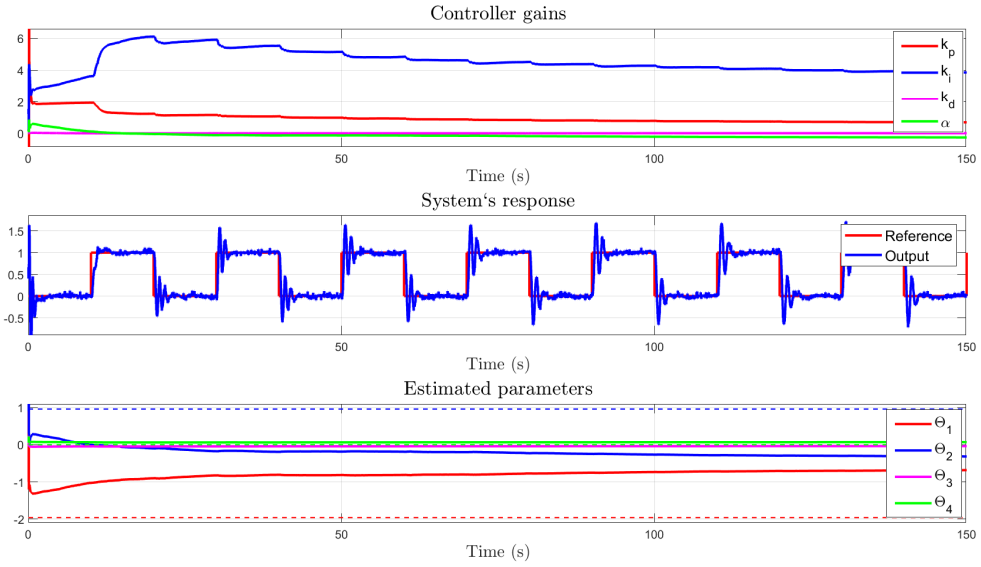
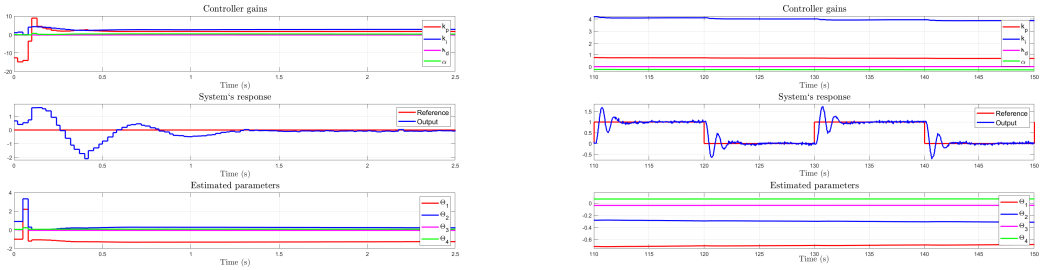


Figure 4.72: Adaptive I-PDα controller experiment.



(a) First 2.5 seconds of the experiment. (b) Last 40 seconds of the experiment

Figure 4.73: Initial and final data of the adaptive I-PD.

But if analyze the Figure 4.74 we can see, the system estimates the parameters such that the estimation error tends to 0.

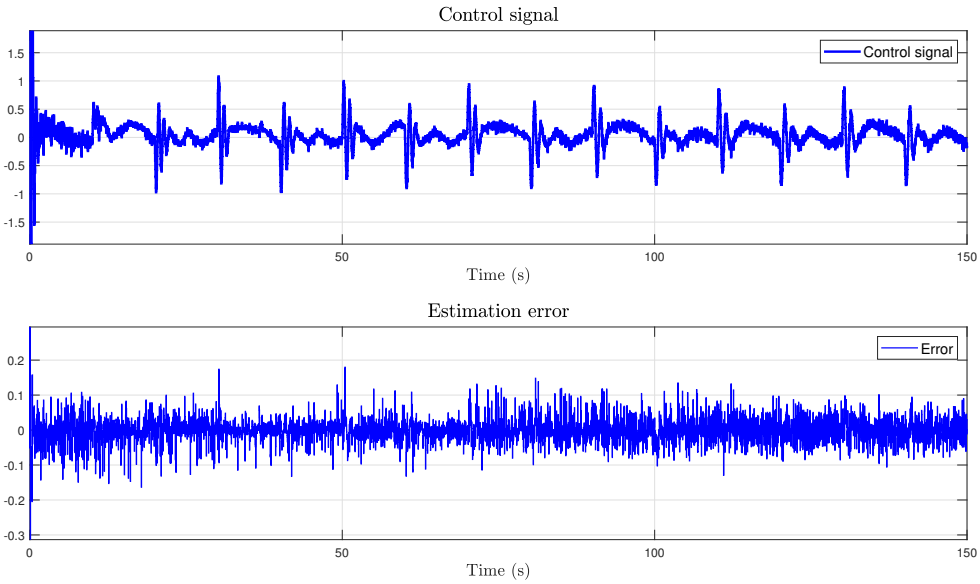


Figure 4.74: Control signal and estimation error of the adaptive I-PD experiment.

Finally, to conclude this last workpackage, a comparison of the adaptive I-PD and adaptive I-PDα controllers have been done. As we can see on Figure 4.75, both controllers have as a result quite similar outputs. As aforementioned, the output does not fulfill the specifications, as the RLS does not estimate the correct parameters. This comes form the fact that in the recursive least squares algorithm, the cost function is to minimize the error between the real output and the estimated output (ε), which does not guarantee the parameters to converge to their real values. Nevertheless, the controller is able to control the output trivially.

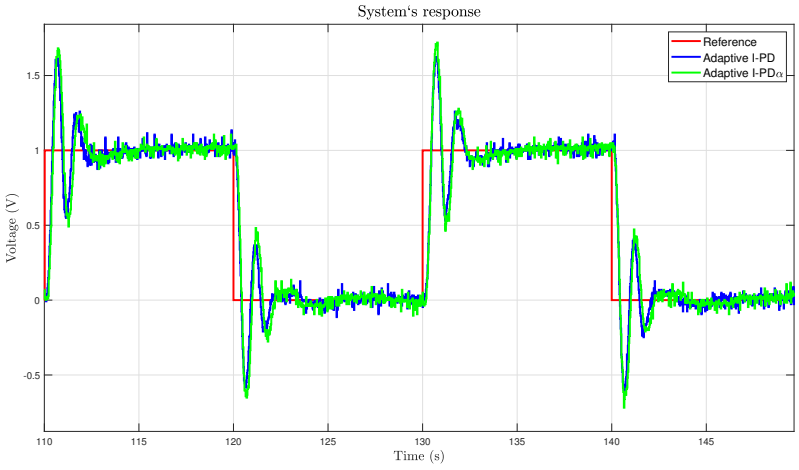


Figure 4.75: Comparison of the two adaptive position controllers implemented, I-PD and I-PDα controllers.

Chapter 5

Satisfaction questionnaire

After the implementation of the new equipment, (an image of the final result of a workstation can be seen on Figure 5.1) a sort of student and teachers were in charge of proving the correct functioning and acceptance of the changes produced. After a semester, the two teachers answered some questions.

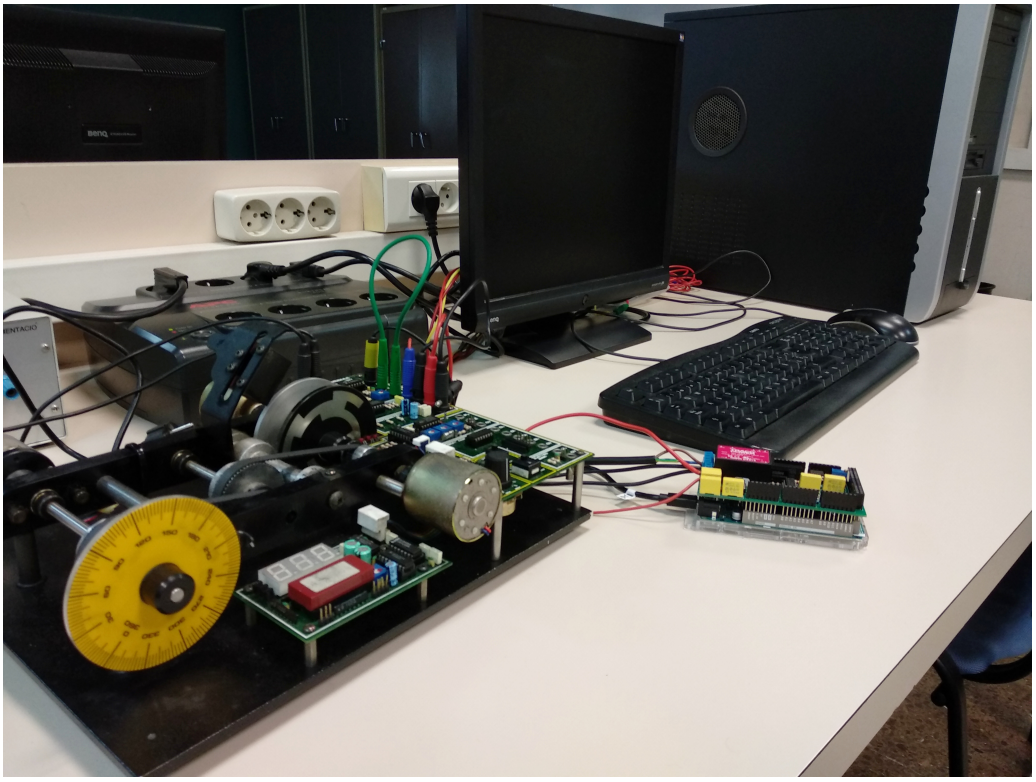


Figure 5.1: One of the workstations of the automatic control laboratory.

To help ensure a successful product launch, or even to notice about some leaks it is important to get feedback from potential users early in the planning process. So, after the test a satisfaction questionnaire has been administered with several questions that must be answered in a scale from 0 to 6 (by the following options: strongly agree, agree, neutral, disagree, strongly disagree or not applicable) corresponding to the degree of the

satisfaction felt. By this questionnaire we will have an input on what people think of our offering, as well as new ideas to implement on it.

This questionnaire it is based on a Product Testing Survey which includes the following questions:

1. You are a:
    - Teacher
    - Student
2. What is your first reaction to the product?
    - Very positive
    - Somewhat positive
    - Neutral
    - Somewhat negative
    - Very negative
3. Ease of installation
    - Strongly agree
    - Agree
    - Neutral
    - Disagree
    - Strongly disagree
4. Ease of use
    - Strongly agree
    - Agree
    - Neutral
    - Disagree
    - Strongly disagree
5. The changes produced in the Simulink models are very high
    - Strongly agree
    - Agree
    - Neutral

- Disagree
- Strongly disagree

6. How much did you like the design of the product?
   - I like it a lot
   - I like it a bit
   - Neither liked or disliked
   - I disliked it a bit
   - I did not liked at all
7. Overall, how satisfied are you with this new equipment?
   - Very satisfied
   - Satisfied
   - Neutral
   - Unsatisfied
   - Very unsatisfied
8. What are the things that you like most about this new product?

9. How can this product be further improved?

After the survey, we concluded that the first reaction to the product is very positive and that it is easy to use and install. Also, we can say that the Simulink models haven't change a lot which is another positive point.

The most liked things of this product are the easy to use and the openness of the design. While as future improvements we can say that some exercises require a fastening change. Those exercises are quite repetitive (taking measures for a range of k_p, different T_s, different input frequencies...), and the system takes a lot of time to recompile for every iteration in the measurement process therefore, lot of time is lost. Along with this, cables may not fit beautifully in the casing, so a redesign of the equipment can be done.

Chapter 6

Costs

This document shows one of the most important analysis of the project: the budget. In it, the costs of materials, equipment and computer programs necessary to carry out the project are analyzed independently (hardware and software); along with the time it takes the engineer to carry out the project (manpower). Finally, the total cost of the project is explained, that is, what the client has to pay in the established periods.

6.1 Material

6.1.1 Software

In order to make the project, it was necessary to use MATLAB/Simulink program. In order to make the data transfer too, `Simulink Coder`, `MATLAB Coder` and `Embedded Coder` toolboxes must be installed.

If the client owns the necessary programs, this cost will not be charged.

SOFTWARE			
Material	Quantity	Unitary cost	Total cost
MATLAB (perpetual license)	1	2,000.00	2,000.00
Simulink	1	3,000.00	3,000.00
Simulink Coder	1	3,000.00	3,000.00
Embedded Coder	1	5,000.00	5,000.00
MATLAB Coder	1	5,000.00	5,000.00
		TOTAL	18,000.00€

Table 6.1: Total software cost

6.1.2 Hardware

This section describes, as previously mentioned, the cost of computer and electronic equipment necessary for the execution of the project. It should be noted that the selected computer is not of the highest gamma but, nevertheless, it has all the features to work properly.

HARDWARE			
Material	Quantity	Unitary cost	Total cost
Computer	1	600	600
MS15 D.C. Motor Control Module	1	2,112.00	2,112.00
PS40 Power Supply Unit	1	468.00	468.00
4mm Connection Lead Set	1	132.00	132.00
PCB	1	27.00	27.00
Arduino Due	1	36.39	36.39
TMPM 04212	1	18.64	18.64
TMV 1205D	1	6.90	6.90
TME 0503S	1	6.04	6.04
TL074 opamp	1	0.69	0.69
TLC2272 opamp	2	1.80	3.6
HD74LS86P XOR	1	4.67	4.67
ADG508P MUX	1	6.45	6.45
LM385Z	1	0.65	0.65
T910Y 10k potentiometer	1	0.32	0.32
1N4148 switching diodes	10	0.024	0.24
MKP273310-Capacitor.MKP 27nf	6	0.15	0.90
CC10463-multilayer capacitor 100nf	9	0.05	0.45
20k resistor	18	0.054	0.972
5k1 resistor	1	0.054	0.054
6k5 resistor	4	0.054	0.216
4k22 resistor	2	0.054	0.108
22 resistor	2	0.054	0.108
2285D40-1 row female pin strip 40cts wrap	4	1.90	7.60
2752-strps 2 c 5.08mm	3	0.20	0.60
		TOTAL	3,434.60€

Table 6.2: Total hardware cost

6.2 Manpower

In this section, the number of hours required by the engineer to carry out the project is taken into account. In addition, the engine installation and the proper functioning of it are also taken into account.

Manpower				
Concept	Detail	cost/hour	Hours	Total cost
Design and Programming	programming	40	150	6,000.00
	changes and adjustments	40	80	3,200.00
Initialization	Initial approach	30	10	300.00
Launching	installation	55	5	275.00
	checking	55	15	825.00
			TOTAL	10,600.00 €

Table 6.3: Total manpower cost

6.3 Summary and total cost

In this section, the price of each part of the project and the total cost are explained.

SUMMARY	
Concept	Total cost
Hardware	3,434.60
Software	18,000.00
Manpower	10,600.00
Partial cost	32,034.60
VAT (%21)	6,727.27
TOTAL COST OF THE PROJECT	38,761.87 €

Table 6.4: Summary and total cost

Therefore, the total cost of the project would be: **Thirty-eight thousand seven hundred seventy-one euros with eighty-seven cents (47,110.87€)**

As mentioned previously, the price of the total cost may vary according to the customer's needs.

It is also worth to comment that, in this project, for each plant **we have saved 214.33 €**. This comes from the fact that the PCI-1711-BE costs 366.93 €and that the designed Arduino Due shield along with an Arduino Due board only costs 112.6 €.

Chapter 7

Environmental impact

In this section we try to identify, describe and evaluate, in an appropriate manner, the direct and indirect effects of a project on the environment.

The purpose of this project has been to design an Arduino shield so that it can be used as a support tool for learning in the ETSEIB's automatic control laboratory. Thanks to this, we could motivate the students in the learning of the theoretical concepts acquired in class, besides being able to help the teaching staff in finding a linked tool to explain and show some other desired notion. With all this, the social impact is positively valued.

In terms of environmental impacts, this project uses several components whose production could generate pollution. That is why we have purchased the minimum number of possible components and during their welding to the PCB, we have tried to waste the minimum tin and tried not to have the tinner long time switched on. In addition to this, the designed PCB complies with the RoHS protocol. Along with this, taking into account the system's power supply has a nominal power consumption of $40W$ and a personal laptop used to control the motor of $20W$, we have a system that consumes $60W$. Taking into account that the complete system has been switched on approximately for 1,315 hours during the project, we have an electrical consumption of 78.9 kWh on a year. So, taking into account that, 370 gCO_2/kWh is emitted in Spain [3], we arise to 29.2 $kgCO_2$ emitted in all the project, which should try to be reduced for the following years.

Chapter 8

Planning and scheduling

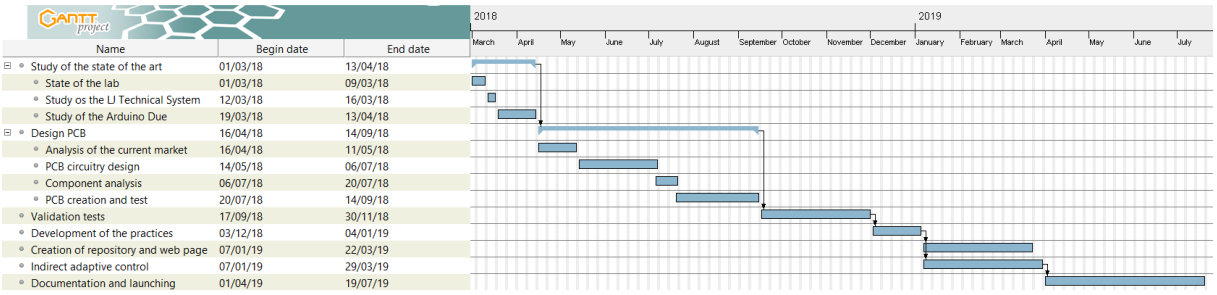
Before starting the project, the objectives were analyzed and some milestones were fixed. For that, the main ideas where extracted and ordered in sequential order.

1. Study of the state of the art of the project (150 hours).
    - Current state of the laboratory.
    - Study of the Arduino Due microcontroller.
    - Study of the LJ Tachnical System's servo system.
2. Designed PCB's initial setup (250 hours).
    - Due to incompetence of the I/O ranges of the arduino due and the sorvosystem, a signal adapter will be designed.
    - Analysis of the current market Arduino shields.
    - PCB circuitry design.
    - Different component analysis.
    - PCB creation.
    - Documentation of the steps followed.
3. Development of small validation tests of the new hardware (50 hours).
    - Analyze Simulink's analog input and output blocks using the new hardware.
4. Development of the current practices of the automatic control subject (350 hous).
    - Redo the Simulink models and compare the results obtained with those obtained with the previous setup.
    - Launch of the new equipment.
5. Creation of repository and web page (320 hours).

- Creation of video tutorials explaining the steps to follow and the results together with the conclusions.
- Creation of MATLAB's Live Scripts to facilitate the coding.
- Design and creation of the web page.

6. Indirect adaptive control (130 hours)
   - Simulation of indirect adaptive control for velocity and position control.
   - Rapid control prototyping of the designed adaptive controllers.
7. Writting of the document that collects the data obtained in all the work and final exposition of the complete development (65 hours).

Through the following Gantt chart, see Figure 8.1 you can follow the execution of the steps explained above to carry out this project.



Name	Begin date	End date
Study of the state of the art	01/03/18	13/04/18
State of the lab	01/03/18	09/03/18
Study os the LJ Technical System	12/03/18	16/03/18
Study of the Arduino Due	19/03/18	13/04/18
Design PCB	16/04/18	14/09/18
Analysis of the current market	16/04/18	11/05/18
PCB circuitry design	14/05/18	06/07/18
Component analysis	06/07/18	20/07/18
PCB creation and test	20/07/18	14/09/18
Validation tests	17/09/18	30/11/18
Development of the practices	03/12/18	04/01/19
Creation of repository and web page	07/01/19	22/03/19
Indirect adaptive control	07/01/19	29/03/19
Documentation and launching	01/04/19	19/07/19

Figure 8.1: Gantt diagram of the project's planning.

Conclusions

After done all the research and analyze the results obtained, it could be concluded that we have achieved all the initial purposes, leaving several possible modifications and improvements that could be made.

First of all, a versatile Arduino Due shield has been designed. Apart from working perfectly on our plants, by modifying the components values, we can have a shield that can work on another kind of system with different I/O ranges.

A workstation of the automatic control laboratory is already working with the setup designed and implemented in this work and a sort of students and teachers have been working with it. After the questionnaire, we can conclude they are very satisfied and that they had a positive first reaction to the change.

Along with this, it can be stated that the new Simulink models have not changed too much and so we can keep on working as they worked before this project began and reuse the manual developed by the teachers of the subject.

Apart from the practical sessions carried out, on one hand, we have learned how to reschedule the PID controllers so that the temporal responses are improved. On the other hand, we can see how to design and implement an indirect adaptive control. In this last part, we have introduced the RLS algorithm and some other possible adaptive control configurations.

Finally, we have designed a complete web page, where all the practical sessions carried out on the research are explained by MATLAB live scripts and the behavior of the motor can be analyzed by the walk-though videos. This web page is attached to a repository where all the files created while developing the project are available.

As future work, we could say some of the Simulink models (those models that have repetitive exercises), can be improved so that in one simple compilation all the iterations are carried out. The casing could also be improved so that there is no mess of cables.

As a result of this work, the following scientific papers have been presented: [7], [8], [5] and [6].

Acknowledgements

By these words I want to thank everyone who has appeared and left during these last years and has helped me being the person I am.

To my parents and brother for giving me the opportunity to learn everything I wanted and for supporting me during these years. To my father for teaching me to use all kind of work tool to repair daily things, to my mother for teaching me to overcome my own weaknesses and to my brother for being my source of inspiration in life.

To Ramon Costa and Robert Griñó for their wise advice, for having trusted me for the realization of this Thesis and for their invaluable help and teachings, without which it would have been very difficult to carry out this work. They have always been willing to dedicate their time and their knowledge.

To Cristina Lampón, always willing to help and advise me from the very beginning of this project and beyond.

Finally, I want to thank Mathworks and Carlos Sanchis, technical specialist of Mathworks for Spain and Portugal, for giving me the opportunity to work in such an interesting research.

Definitely all those named and not named but no less important, such as friends and family, have given me all the necessary tools to complete my work satisfactorily.

Bibliography

[1] Analog Devices. *8-Channel/4-Channel Fault-Protected Analog Multiplexers*. Available in: `https://www.analog.com/media/en/technical-documentation/data-sheets/ADG508F_509F.pdf`

[2] Arduino. *Arduino Due*. Available in: `https://store.arduino.cc/due`.

[3] CeroCO2. *Cálculo de Huella de Carbono por consumo eléctrico*. Available in: `https://www.ceroco2.org/calculadoras/electrico`

[4] Diodes incorporated. *1N4148/1N4448 fast switching diode*. Available in: `https://www.diodes.com/assets/Datasheets/ds12019.pdf`

[5] E. Lerma, R. Costa Castelló, R. Griño, C. Sanchis. *Duino-based learning (DBL) in control engineering courses*. Conference on Emerging Technologies and Factory Automation (ETFA) 2019, Accepted.

[6] E. Lerma, R. Costa Castelló, R. Griño, C. Sanchis. *Duino-besd learning(DBL): un proyecto para facilitar el uso de Arduino y MATLAB en la docencia de la automática*. Jornadas de Automática 2019. In process.

[7] E. Lerma, R. Costa Castelló, R. Griño, C. Sanchis. *Implementación de controladores en Arduino mediante Simulink*. Jornadas de Automática Sep 2018. Available in: `http://eii.unex.es/ja2018/actas/JA2018_099.pdf`.

[8] E. Lerma, R. Costa Castelló, R. Griño, C. Sanchis, S. Dormido. *On Teaching Digital Control Systems in a Generic Engineering Degree*. Advances in Control Education, ACE 2019. Accepted.

[9] F.A. Candelas, G.J. García, S. Puente, J. Pomares, C.A. Jara, J. Pérez, D. Mira, and F. Torres. *Experiences on using arduino for laboratory experiments of automatic control and robotics*. IFAC-PapersOnLine, 48(29):105-110, 2015.

[10] Jaroslav Sobota, Roman PiSl, Pavel Balda, and MiloS Schlegel. *Raspberry pi and arduino boards in control education*. IFAC Proceedings Volumes, 46(17):7-12, 2013.

[11] Karl J.Astrom, Bjorn Wittenmark. *Adaptive Control*. Dover Publications (1994).

[12] LJ Create. *D.C. Motor Control Module*. Available in: http://www.ljcreate.com/uk/?option=com_virtuemart&view=productdetails&virtuemart_product_id=313&&virtuemart_category_id=64&

[13] LJ Create. *Power Supply Unit*. Available in: http://www.ljcreate.com/uk/?option=com_virtuemart&view=productdetails&virtuemart_product_id=314&&virtuemart_category_id=64&

[14] LJ Create. *4mm Connection Lead Set*. Available in: http://www.ljcreate.com/uk/?option=com_virtuemart&view=productdetails&virtuemart_product_id=312&&virtuemart_category_id=64&

[15] LJ Technical Systems. *DC Motor Control Module*. User Manual. Hauppauge, NY.

[16] Masato ISHIKAWA and Ichiro MARUTA. *Rapid prototyping for control education using arduino and open-source technologies*. IFAC Proceedings Volumes, 42(24):317-321, 2010.

[17] Mathworks. *MATLAB product description*. Available in: https://es.mathworks.com/help/matlab/learn_matlab/product-description.html?lang=en

[18] Mathworks. *Simulink product description*, Available in. https://es.mathworks.com/help/simulink/gs/product-description.html

[19] Oriol Causí Casamor, Miquel Angel Mañanas Villanueva, Ramon Costa Castelló, and Luis Basañnez Villaluenga. *Control amb Computador. Simulación en entorno MATLAB*. CPDA, Barcelona, Octubre 1999. ISBN 84-95355-04-3.

[20] P. Reguera, D. García, M. Domínguez, M.A. Prada, and S. Alonso. *A low-cost open source hardware in control education. case study: Arduino-feedback ms-150*. IFAC-PapersOnLine, 48(29):117-122, 2015.

[21] Profesores de la asignatura Control por computador. *Pràcticas*. Technical report. Escuela Técnica Superior de Ingeniería Industrial de Barcelona (ESTEIB), Barcelona, 1998.

[22] R. Costa Castelló, E. Fossas Colet, *Sistemes de control en temps discret*. Universitat Politècnica de Catalunya. Iniciativa Digital Politècnica, 2014.

[23] R. Barber, M. Horra, and J. Crespo. *Practices using simulink with arduino as low cost hardware*. IFAC Proceedings Volumes, 46(17):250-255, 2013.

[24] Renesas Electronics. *HD74LS86 Quadruple 2-input Exclusive-OR Gates*. Available in: https://www.renesas.com/us/en/doc/products/logic/001/rej03d0422_hd74ls86.pdf

[25] SR Passives. *T910 Trimming Potentiometers*. https://www.tme.eu/Document/46e9ecf29426285f0ed14cb1dbb4a9ca/SR_Passives-T910.pdf

[26] Texax instruments. *LM185/LM285/LM385 Adjustable Micropower Voltage References*. Available in: http://www.ti.com/lit/ds/snvs741f/snvs741f.pdf

[27] Texax instruments. *TL07xx Low-Noise JFET-Input Operational Amplifiers*. Available in: http://www.ti.com/lit/ds/symlink/tl074b.pdf

[28] Texax instruments. *TLC227x, TLC227xA: Advanced LinCMOS Rail-to-Rail Operational Amplifiers*. Available in: http://www.ti.com/lit/ds/symlink/tlc2274a.pdf

[29] TRACO Power. *TME series*. https://www.tracopower.com/products/browse-by-category/find/tme/3/

[30] TRACO Power. *TMP - TMPM Series*, Available in: https://www.tracopower.com/products/browse-by-category/find/tmp-tmpm/3/

[31] TRACO Power. *TMV series*. Available in: https://www.tracopower.com/products/browse-by-category/find/tmv/3/

Appendix A

Jury's stability test

In control theory Jury's stability criterion is used to determine the stability of a linear system, by analyzing the coefficients of the characteristic equation. The test stands that, given the characteristic equation:

$$D(z) = a_n z^n + ... + a_2 z^2 + a_1 z + a0 \tag{A.1}$$

If the equation fulfills the following four rules, the system will be stable, otherwise, if one of this rules is not fulfilled, the poles will be outside the unitary circle and the system will be unstable.

- **Rule 1:** $D(1) > 0$
- **Rule 2:** $(-1)^n D(-1) > 0$
- **Rule 3:** $|a_0| < a_n$
- **Rule 4:** If the system's order is grater than 2, then Jury's table must be filled and the sufficient conditions obtained.

Going indeep to our particular case, we have a second order system which characteristic equations, having as unknown parameter T_s, is:

$$Q(z) = z^2 + \left[-1 - \alpha + k_p k_{tot} T_s + k_p k_{tot} \tau_{mot} (\alpha - 1)\right] z + \alpha + k_p k_{tot} \tau_{mot} \left(1 - \alpha - \frac{T_s}{\tau_{mot}} \alpha\right) \tag{A.2}$$

so, we will analyze the range of T_s so that the system is stable by fulfilling the first three rules above. In this way:

Rule 1: $Q(1) > 0$

$$1 - 1 - \alpha + k_p k_{tot} T_s + k_p k_{tot} \tau_{mot} (\alpha - 1) + \alpha + k_p k_{tot} \tau_{mot} \left(1 - \alpha - \frac{T_s}{\tau_{mot}} \alpha\right) > 0$$

$$k_p k_{tot} T_s (1 - \alpha) > 0$$

The inequality will be fulfilled if T_s**<0** or T_s**>0**

Rule 2: $Q(-1) > 0$

$$1 + 1 + \alpha - k_p k_{tot} T_s - k_p k_{tot} \tau_{mot} (\alpha - 1) + \alpha + k_p k_{tot} \tau_{mot} \left(1 - \alpha - \frac{T_s}{\tau_{mot}} \alpha\right) > 0$$

$$2 + 2\alpha + k_p k_{tot} + [2\tau_{mot} (1 - \alpha) - T_s (1 + \alpha)] > 0$$

The inequality will be fulfilled if T_s**<0.6793**

Rule 3: $|a_0| < a_n$

$$\left|\alpha + k_p k_{tot} \tau_{mot} \left(1 - \alpha - \frac{T_s}{\tau_{mot}} \alpha\right)\right| < 1$$

The inequality will be fulfilled if **0**<T_s**<0.27958111**

As we can notice, time can not be negative, so the minimum value is 0, while, in order to take the maximum for T_s, we have to take the most restrictive value, the least of all them, in this case 0.279. So the range for $T_s \in [0, 0.279]$.

Appendix B

Anti-windup filter

An inevitable control problem, is the existence of limitations in the actuators. This problem is usually ignored when designing controllers, but after their design we can add new compensation methods to delete those negative effects. One of the main problems comes when we have an integral part on the controller that keeps integrating even if the actuator is saturated, called windup. As most of our controllers have an integral part, this Appendix searches the best anti-windup solution when having an integral part digitized by the trapezoidal approach.

First approach $C_{aw} = \frac{k_{aw}}{z}$:

$$\begin{aligned} I(z) &= \frac{k_i}{2}\frac{z+1}{z-1} \\ C_{aw}(z) &= \frac{k_{aw}}{z} \end{aligned} \tag{B.1}$$

The denominator of this part stands as follows:

$$\text{Den}(z) = 2z^2 + (k_{aw}k_i - 2)\, z + k_{aw}k_i \tag{B.2}$$

So the poles are located at:

$$\begin{aligned} p_1 &= \frac{-k_{aw}k_i}{4} + \frac{1}{2} + \frac{\sqrt{k_{aw}^2k_i^2 - 12k_{aw}k_i + 4}}{4} \\ p_2 &= \frac{-k_{aw}k_i}{4} + \frac{1}{2} - \frac{\sqrt{k_{aw}^2k_i^2 - 12k_{aw}k_i + 4}}{4} \end{aligned} \tag{B.3}$$

In order to have the fastest dynamics possible, we have to place the poles in the real axis, and so have double poles on the same real axis place. For that we make the polynomial on the square root equal to zero and solve it for k_{aw}. In this way:

$$\text{Poly} = k_{aw}^2 k_i^2 - 12 k_{aw} k_i + 4 \tag{B.4}$$

Obtaining two possible relationship

$$\begin{aligned} k_{aw1} &= \frac{2\left(3+2\sqrt{2}\right)}{k_i} \\ k_{aw2} &= \frac{2\left(3-2\sqrt{2}\right)}{k_i} \end{aligned} \tag{B.5}$$

By the first relationship we obtain that the poles are located on

$$\begin{aligned} p_1 &= -2.414178207 \\ p_2 &= -2.414248917 \end{aligned} \tag{B.6}$$

And by the second one we obtain that the poles location is:

$$\begin{aligned} p_1 &= 0.414213562 + 0.000025i \\ p_2 &= 0.414213562 - 0.000025i \end{aligned} \tag{B.7}$$

So, 0.4142 is fastest the poles of the anti-windup can have being as anti-windup structure $C_{aw} = \frac{k_{aw}}{z}$.

Second approach $C_{aw} = \frac{k_{aw}}{z-a}$:

We include a pole in $z = a$ of the anti-windup and analyze its root locus. With this second approach, we know the characteristic equation follows:

$$1 + KGH(z) = 0 \text{ where } GH(z) = \frac{z+1}{(2z-2a)(z-1)} \tag{B.8}$$

$$\text{and so, } K = \frac{-1}{GH(z)} = -\frac{(2z-2a)(z-1)}{z+1} \tag{B.9}$$

And deriving K and solving it to zero, we can obtain its singular points, which are:

$$\begin{aligned} ps_1 &= -1 + \sqrt{2a+2} \\ ps_2 &= -1 - \sqrt{2a+2} \end{aligned} \tag{B.10}$$

We will take the positive singular point and after substituting $z = -1 + \sqrt{2a+2}$ we will obtain the equation that describes K, which is:

$$f_k = -\frac{8}{\sqrt{2a+2}} + 6 - \frac{8a}{\sqrt{2a+2}} + 2a \tag{B.11}$$

And if we plot this equation from $a = [-1, 1]$, we can analyze, ps_1 is always a singular point on the root locus.

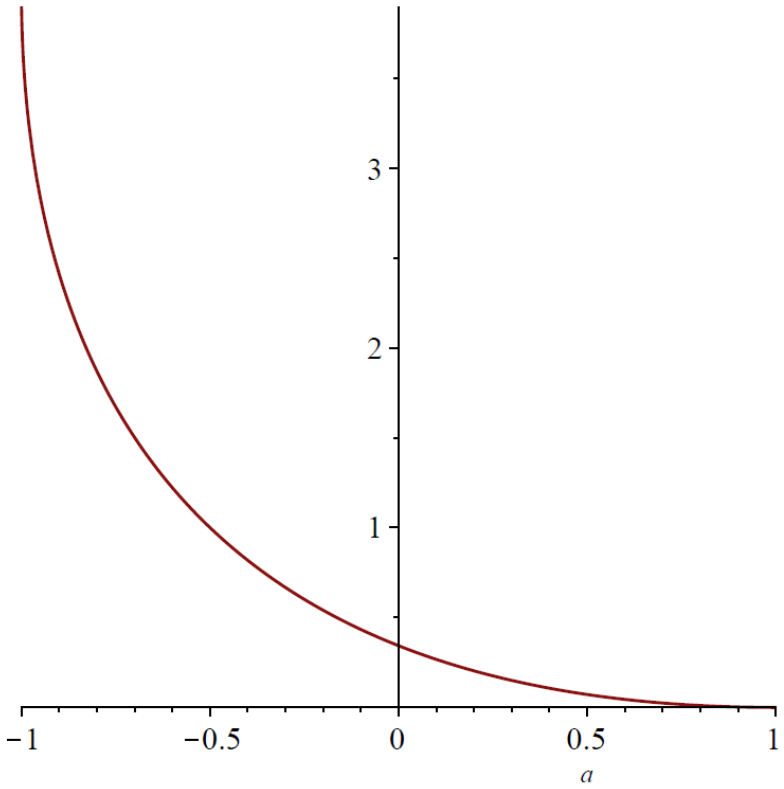


Figure B.1: f_k plot.

Furthermore, as we can see on the following figure, if we plot the graph of ps_1 and choose $a = -0.5$ we can see this singular point is located on 0.

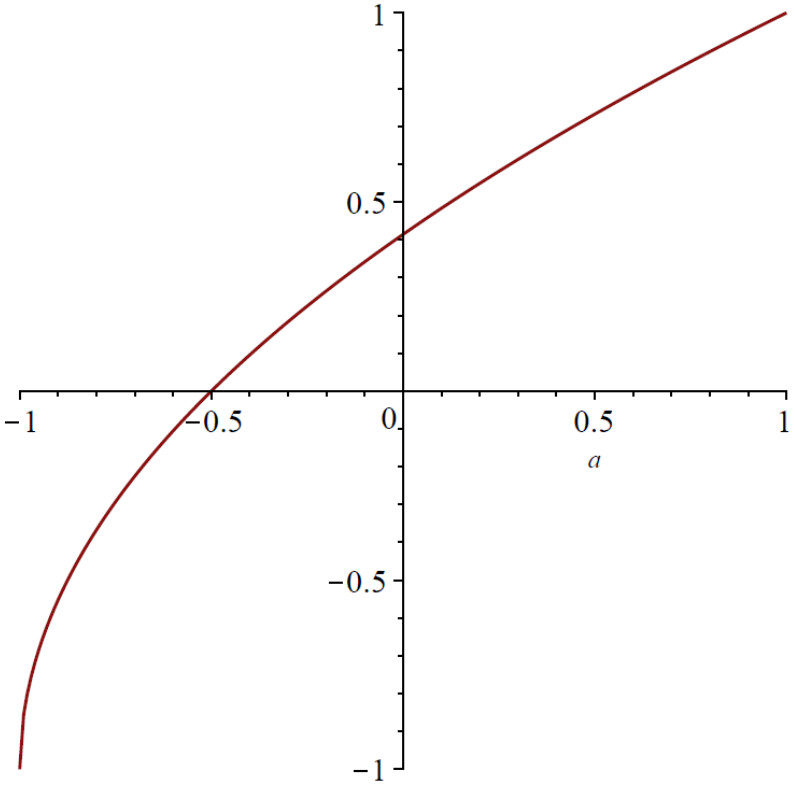


Figure B.2: ps_1 plot.

So, the chosen anti-windup structure will be $C_{aw} = \frac{k_{aw}}{z+0.5}$

If we obtain the closed-loop characteristic equation for this structure, we can analyze where the poles are located. In this manner we have that:

$$\text{Den}(z) = z^2 + (0.5k_{aw}k_i - 0.5)\, z + 0.5k_{aw}k_i - 0.5 \tag{B.12}$$

and so the poles:

$$\begin{aligned} p_1 &= -0.25k_{aw}k_i + 0.25 + 0.25\sqrt{k_{aw}^2k_i^2 - 10k_{aw}k_i + 9} \\ p_2 &= -0.25k_{aw}k_i + 0.25 - 0.25\sqrt{k_{aw}^2k_i^2 - 10k_{aw}k_i + 9} \end{aligned} \tag{B.13}$$

If we chose $k_{aw} = \frac{1}{k_i}$ we find that we are on the singular point and that both poles are located on 0.

Plotting the step response of the closed-loop system with the relationship above, we can see that in two samples, we reach the steady-state.

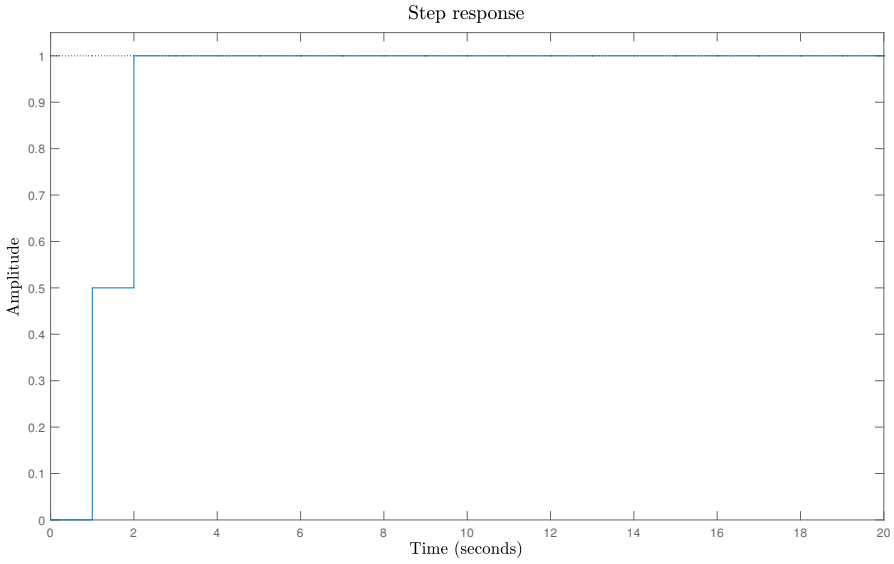


Figure B.3: Step response of the anti-windup structure.

So, if we chose a trapezoidal accumulator, we have to design an anti-windup filter $C_{aw} = \frac{k_{aw}}{z+0.5}$ and select $k_{aw} = \frac{1}{k_i}$. Thus, we will have the two poles of the anti-windup loop located on 0.

Appendix C

Codes

In this section, you will find the codes implemented to tune the aforementioned controllers.

C.0.1 PI/I-P controller design

```
% Plant's parameters
K0=0.82;
tau0=0.26;

% Sampling time
Ts=0.01;

% Especifications
xi=1;
ts_2=0.8; % settling time with %2 criterion
Tdes=ts_2/4;
wn=5.8/(xi*ts_2);

% Controller design
Zdes=exp(-Ts/Tdes);
alpha=exp(-Ts/tau0);
beta=-2*exp(-Ts*xi*wn)*cos(wn*Ts*sqrt(1-xi^2));
gamma=exp(-2*Ts*xi*wn);

ki_star=(gamma-alpha+beta+(alpha+1))/(2*K0*(1-alpha));
ki=(2/Ts)*ki_star;
kp=((beta+alpha+1)/(K0*(1-alpha)))-ki_star;
Kaw=1/ki;
```

C.0.2 PID/I-PD controller design

```
% Definition of the plant's parameters
K0=0.82/0.017;
tau0=0.26;
N=9;
Kpot=1.62;

% Sampling period
Ts=0.01;

% Desired control specifications
Sp=80; % overshoot
Fd=0.5; % frequency

% Poles of the second order continuous system that will
% fulfill the specifications
wd=2*pi*Fd ;
xi=sqrt((log(Sp/100))^2/(pi^2+log(Sp/100)^2))
wn=wd/ sqrt(1-xi^2)
s1=-xi*wn+j*wd
s2=-xi*wn-j*wd

% Poles of the second order discrete system that will fulfill the
   specifications
p1=exp(Ts*s1)
p2=exp(Ts*s2)

% Fix third pole
p3=0.01

% Plant's Z transfere function
Ptas=tf([K0*(1/N)*Kpot],[tau0,1,0]);
Ptaz=c2d(Ptas,Ts,'zoh');

% Denominator and numerator coefficients
[Nz ,Dz]= tfdata(Ptaz,'v')
a1=Nz(2);
a0=Nz(3);
b1=Dz(2);
b0=Dz(3);

% A and B matrix definitions
A=[-1 -a1 -a1 -a1 ; p2+p3+p1 a1-a0 -a1-a0 -a0+2*a1 ; ...
   -p1*p2-p3*p1-p2*p3 a0 -a0 2*a0-a1 ; p1*p2*p3 0 0 -a0 ] ;
b=[p1+p2+p3-1+b1 ; -p1*p2-p3*p1-p3*p2+b0-b1 ; p1*p2*p3-b0 ; 0 ] ;
```

```
% x vector's determination of the controller
x=inv(A)*b ;

p4=x(1)
kp=x(2)
ki=(2/Ts)*x(3)
kd=Ts*x(4)
% Anti-wind-up
Kaw=1/ki;

% Poles dominance calculation
dominancia=log(abs(p4))/log(abs(p1))
```

C.0.3 PIDα controller design

```
% Definition of the plant's parameters
K0=0.82/0.017;
tau0=0.26;
N=9;
Kpot=1.62;

% Sampling period
Ts=0.01;

% Desired control specifications
Sp=80; % overshoot
Fd=0.5; % frequency

% Poles of the second order continuous system that will
% fulfill the specifications
wd=2*pi*Fd ;
xi=sqrt((log(Sp/100))^2/(pi^2+log(Sp/100)^2))
wn=wd/ sqrt(1-xi^2)
s1=-xi*wn+j*wd
s2=-xi*wn-j*wd

% Poles of the second order discrete system that will fulfill the
   specifications
p1=exp(Ts*s1)
p2=exp(Ts*s2)

% Fix poles 3 and 4
p3=0.8
```

```
p4=0.8

% Plant's Z transfere function
Ptas=tf([K0*(1/N)*Kpot],[tau0,1,0]);
Ptaz=c2d(Ptas,Ts,'zoh');

% Denominator and numerator coefficients
[Nz ,Dz]= tfdata(Ptaz,'v')
a1=Nz(2);
a0=Nz(3);
b1=Dz(2);
b0=Dz(3);


%% Method 1: by matrix
% A and B matrix definitions
A=[1 a1 0 0; ...
   b1-1 a0 a1 0;
   b0-b1 0 a0 a1;
  -b0 0 0 a0];
b=[-b1+1-p1-p2-p3-p4;
   -b0+b1+p1*p2-(-p1-p2)*p3-(-p1-p2-p3)*p4;
   b0-p1*p2*p3-(p1*p2-(-p1-p2)*p3)*p4;
   p1*p2*p3*p4];


% Controller
x=inv(A)*b;

alpha=x(1)
c2=x(2)
c1=x(3)
c0=x(4)

A2=[1 1 1; alpha-1 alpha+1 -2; -alpha alpha 1];
B2=[c2;c1;c0];

x2=inv(A2)*B2;

kp = x2(1)
ki = (2/Ts)*x2(2)
kd = Ts*x2(3)

%% Method 2: by equations
% desired char. eq
beta3=-p1-p2-p3-p4;
beta2=p1*p2-(-p1-p2)*p3-(-p1-p2-p3)*p4;
```

```
beta1=-p1*p2*p3-(p1*p2-(-p1-p2)*p3)*p4;
beta0=p1*p2*p3*p4;

% solve equations canonical controller
a0=-(beta0*n1^3-beta1*n0*n1^2+beta2*n0^2*n1-beta3*n0^3-d0*n0^2*n1-d0*n0*
    n1^2+d1*n0^3+d1*n0^2*n1-n0^3)/(d0*n0*n1^2+d0*n1^3-d1*n0^2*n1-d1*n0*n1
    ^2+n0^3+n0^2*n1);
b0=(beta0*d0*n1^2-beta0*d1*n0*n1-beta0*d1*n1^2+beta1*d0*n1^2-beta2*d0*n0*
    n1+beta3*d0*n0^2+d0^2*n0*n1+d0^2*n1^2-d0*d1*n0^2-d0*d1*n0*n1+beta0*n0
    ^2+beta0*n0*n1+d0*n0^2)/(d0*n0*n1^2+d0*n1^3-d1*n0^2*n1-d1*n0*n1^2+n0
    ^3+n0^2*n1);
b1=(beta0*d1*n1^2-beta1*d1*n0*n1+beta2*d0*n0*n1+beta2*d0*n1^2-beta2*d1*n0
    *n1-beta3*d0*n0^2-beta3*d0*n0*n1+beta3*d1*n0^2-d0^2*n0*n1-d0^2*n1^2+
    d0*d1*n0^2+2*d0*d1*n0*n1+d0*d1*n1^2-d1^2*n0^2-d1^2*n0*n1-beta0*n0*n1-
    beta0*n1^2+beta1*n0^2+beta1*n0*n1+d1*n0^2)/(d0*n0*n1^2+d0*n1^3-d1*n0
    ^2*n1-d1*n0*n1^2+n0^3+n0^2*n1);
b2=(beta3*d0*n0*n1+beta3*d0*n1^2-beta3*d1*n0^2-beta3*d1*n0*n1-d0*d1*n0*n1
    -d0*d1*n1^2+d1^2*n0^2+d1^2*n0*n1+beta0*n1^2-beta1*n0*n1+beta2*n0^2+
    beta3*n0^2-d0*n0^2+d0*n1^2-d1*n0^2-d1*n0*n1+n0^2)/(d0*n0*n1^2+d0*n1
    ^3-d1*n0^2*n1-d1*n0*n1^2+n0^3+n0^2*n1);

% solve gains controller
alpha=a0
kd=Ts*((a0^2*b2-a0*b1+b0)/(a0^2+2*a0+1))
ki=(2/Ts)*((b0+b1+b2)/(2*(a0+1)))
kp=-(a0*b0-a0*b1-3*a0*b2+3*b0+b1-b2)/(2*(a0^2+2*a0+1))
```