Table of contents

1. Introduction 1
   1.1. Project scope . . . . .
   1.2. Objectives . . . . .
   1.3. Methodology . . . . .

2. Theoretical framework . . . . .
   2.1. ARMDROID 2001 robotic arm . . . . .
   2.2. PWM pulse width modulation . . . . .
   2.3. DC chopper . . . . .
   2.4. Development platform . . . . .

3. Implementation of the control module . . . . .
   3.1. Initial implementation phase . . . . .
   3.2. Hardware implementation . . . . .
   3.3. Software implementation . . . . .

4. Validation of the control module . . . . .

5. Conclusions and recommendations . . . . .
   5.1. Conclusions . . . . .
   5.2. Recommendations . . . . .

Bibliography

A. User manual 1
   A.1 Start up . . . . .
   A.2 Operation of the ARMDROID 2001 robotic arm . . . . .

B. Developer's manual . . . . .
   B.1 Operation of the ARMDROID 2001 robotic arm . . . . .

C. Schematic diagrams . . . . .

Index of figures

2.1 ~~ARMDROID 2001 robotic arm~~ . . . . . .
2.2 PWM operation . . . . . .
2.3 Schematic diagram of a DC chopper . . . . . .
2.4 Equivalent circuit of a DC motor . . . . . .
2.5 Arduino Uno development platform . . . . . .

3.1 Diagram of the control module . . . . . .
3.2 DC motors utilized by ARMDROID 2001 . . . . . .
3.3 Pieza usada para el nal de carrera del hombro . . . . . .
3.4 Moving parts of the ARMDROID 2001 . . . . . .
3.5 Input stage of a single motor . . . . . .
3.6 Output stage of a single motor . . . . . .
3.7 Integrated L298n module . . . . . .
3.8 Schematic for a complete circuit for a single motor . . . . . .
3.9 Block diagram of the software unit . . . . . .
3.10 Signal filtering and normalization . . . . . .
3.11 Obtaining the position signal . . . . . .
3.12 D flip-flop implementation . . . . . .
3.13 Graphic interface for the control module . . . . . .
3.14 Simplified block diagram of control module . . . . . .

4.1 Respuesta de lazo abierto del brazo . . . . . .
4.2 Open-loop response of shoulder . . . . . .
4.3 Closed-loop response of the position control system for the shoulder . . . . . .
4.4 Manual control response . . . . . .
4.5 Closed-loop response with proportional controller . . . . . .
4.6 Error message displayed by program . . . . . .

A.1 Arduino connection with hardware stages for a single motor . . . . . .
A.2 Graphical interface for control module. . . . . .

B.1 Program required to control a single motor . . . . . .
B.2 Obtaining the position signal . . . . . .
B.3 Automatic/manual control block . . . . . .
B.4 Code used to take corrective action . . . . . .

C.1 Complete schematic diagram of the input stage . . . . . .
C.2 Complete schematic diagram of the output stage . . . . . .

1. Introduction

The Laboratory of Automation, located in the School of Electrical Engineering of the University of Costa Rica, has a great variety of resources which have the potential to be used in several courses that are taught at the School. We intend to design a completely new control module with the aim of taking advantage of the LabVolt ARMDROID 2001 robot arm of, because the control module it currently has is not very versatile. That is to say, the manual control is uncomfortable to use, and if you want to load movement routines to the robot arm they must be entered through a 3.5-inch floppy disk (this technology is presently practically obsolete).

The control module to be designed is intended facilitate communicate communication to the robot arm with a desktop computer, in doing so is intended to facilitate the control, monitoring, and programming of the robot arm.

1.1 Project scope

The project focuses on the design of a control module for the robot arm ARMDROID 2001, with which it is possible to manage the robot from a computer. The control module is intended to be controlled through a graphical user interface that allows the user to monitor the status of the robot at all times. A user manual will also be made with the aim that this control module can be used in other courses taught by the School of Electrical Engineering.

The design of a new control module is proposed because the modulus used by the robot is presently impractical, because the monitoring of the robot arm is not efficient, manual control is cumbersome and the programming of the robot using a diskette is obsolete. In summary, the project will revolve around the design of a control module for the ARMDROID 2001 robot arm.

1.2 Objectives

General objective

Design a graphical user interface and control system for the ARMDROID 2001 robot.

Specific objectives

For the development of this project the following objectives were established:

- Investigate the principle of mechanical operation of the ARMDROID 2001.
- Analyze the state of the ARMDROID 2001 robot arm and perform repairs, if necessary.
- Implement the electronics necessary for the control of the robot arm motors.
- Select a development platform for the manipulation of the electronics that controls the ARMDROID 2001 arm motors.
- Develop a graphical user interface such that a user can operate the robot arm safely.
- Create a user manual to explain the aspects in the use of the robot arm.

1.3 Methodology

Because of the way in which the objectives of the project were formulated, the methodology followed was as follows:

1. A general review of the condition of the arm was carried out to investigate the operating mechanism used.
2. Relevant repairs were made based on the results of the review.
3. An electronic circuit capable of controlling the various motors that generate the movements of the robot arm is designed and implemented.
4. A development platform was used that could control the power circuit designed above, and in turn be able to send the necessary signals to the main program located in the computer.
5. A graphical user interface was developed in which the signals sent by the development platform can be received. It must also be able to send control signals to the development platform based on what the user or routine requires.
6. A user manual and a developer's manual can be used as reference for those who need to use the robot.

2. Theoretical framework

In this section the theoretical concepts necessary for the creation of the control module are described. Section 2.1 gives a brief account of what a robot arm is, its applications and features. Section 2.2 discusses pulse width modulation (PWM), which is a widely used technique for the control of electric motors. Section 2.3 discusses power control units known as DC chopper, which provide the power required by the loads that connect to these units. Finally, section 2.4 talks about development platforms, in particular we talk about the Arduino UNO and its usefulness for this project.

2.1 ARMDROID 2001 robotic arm

A robot is a virtual or mechanical device which performs a specific function. In particular it can be said that robot arms are the most frequently manufactured robots because of their ability to perform repetitive tasks quickly and intelligently, which is very useful in large assembly lines. The configuration of a robot arm depends directly on the function to be carried out. In general, these robots use electric motors to perform the different movements that are required, as Khatib and Sicilian (2008) describe.

The ARMDROID 2001 robot arm (see figure 2.1) manufactured by LabVolt can be classified as a spherical arm. That is, the set of all the individual movements that the robot is capable of performing form a sphere. This robot uses six 24V DC motors to perform these movements. It also has four switches, which are activated when the arm reaches a maximum position. When one of these switches is activated, the control module completely stops the movement of the robot at that moment, then reverses the direction of movement that the robot performed until the signal sent by the switch is turned off.

2.2 PWM pulse width modulation

According to University of Madrid Polytechnic (2013), Pulse Width Modulation (PWM), is a switching strategy, usually used to manipulate the duty cycle of a signal to a fixed operating frequency. This is done with the aim of modifying the effective power supplied to a given load, such as a DC motor. The duty cycle can be defined as the effective time in which a given periodic signal is active with respect to its period, whereby the duty cycle can be expressed as:

$$D = \tau / T$$

Where T corresponds to the period of the signal and τ corresponds to the time in high of the square signal. Figure 2.2 shows how PWM is used to modify the average DC value that is generated.

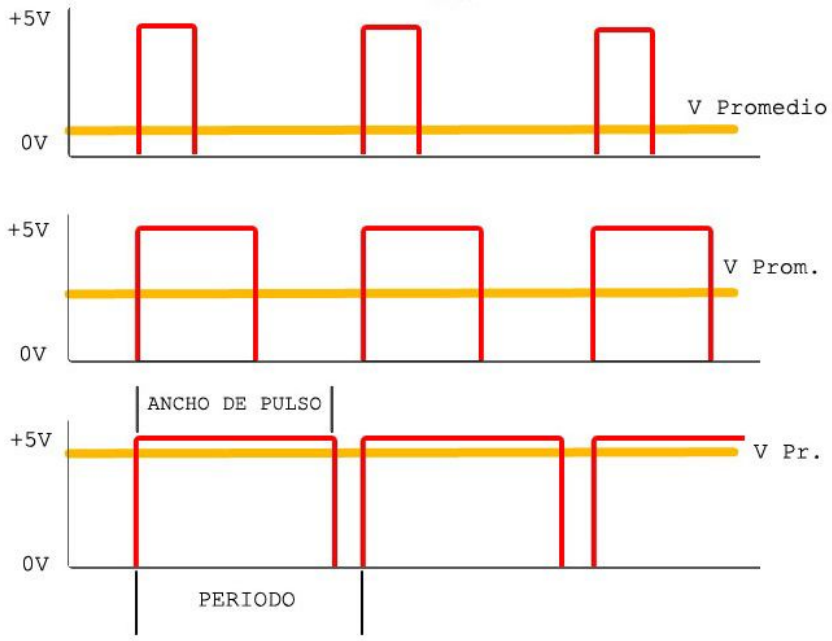


Figure 2.2: PWM operation.

2.3 DC chopper

The DC chopper is a power control unit which allows a bidirectional connection of the electric current using only a voltage source as mentioned by Husain (2003). Figure 2.3 shows the

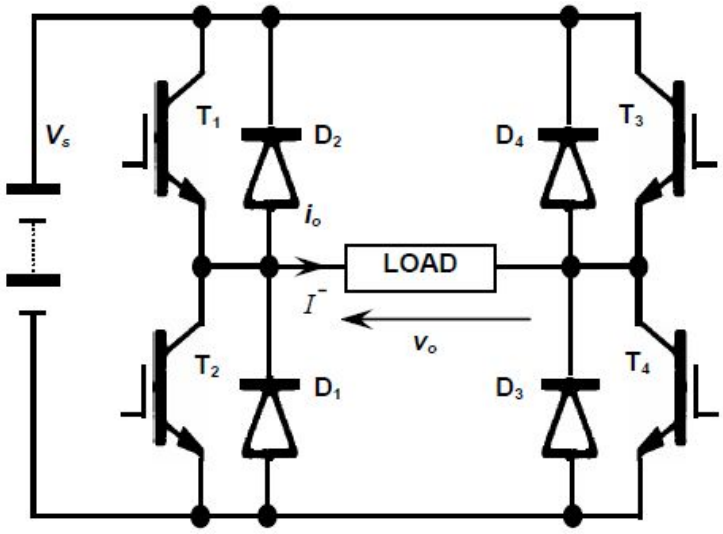


Figure 2.3: Schematic diagram of a DC chopper

configuration of a DC chopper. The transistors are activated or deactivated depending on the control signal being used. In principle it can be said that the parts that make up a DC chopper are the power supply, the control unit for the transistors, the power handling unit or H-bridge and, optionally, a passive filter can be considered to allow for less pronounced voltage changes.

Voltage signal V_s is the source of power voltage, while the array of transistors controls the current through load.

H Bridge

The arrangement of transistors shown in Figure 2.3 is known as an H-bridge. The H-bridge is an electronic circuit generally used to control DC motors. With this scheme it is possible to control the direction of rotation of a motor using only a single power signal, which may be a PWM signal.

DC Motor

A DC motor can be considered as a machine that transforms a DC signal into a mechanical torque. The equivalent circuit considered for the DC motor is shown in Figure 2.4.

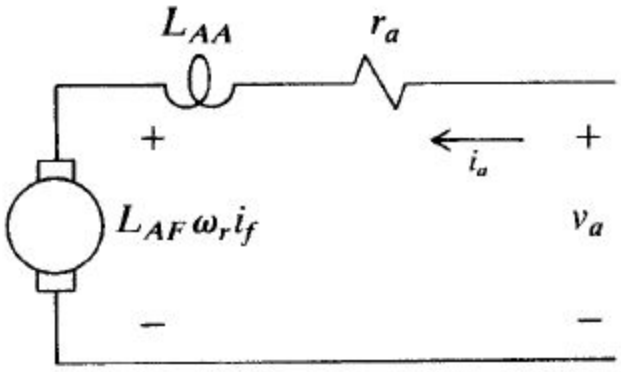


Figure 2.4: Equivalent circuit of a DC motor.

Where L_{AA} represents the motor armature inductance and the armature resistance or resistance of the winding, while the $L_{AF}\omega_r i_f$ factor represents the counter-electromotive force (back EMF)

produced by the motor. The voltage and current $_a$ correspond to the supply voltage of the armature and the armature current that feed the motor respectively.

2.4 Development platform

A development platform is a hardware unit that allows the development of a given project. Arduino is an open-source development platform based on a board with a microcontroller which has several inputs and outputs (CENDITEL, 2013). There are several applications in which you can use an Arduino, such as connecting to a computer program to be used as a data acquisition card. There are several models of Arduino in the market, each one has different characteristics, such as the number of inputs or outputs, greater processing capacity among others.

Figure 2.5 shows an Arduino Uno, which has 14 digital inputs / outputs, of which six can provide a PWM signal, six analog inputs and a 32-bit ATmega328 processor clocked at 16MHz.

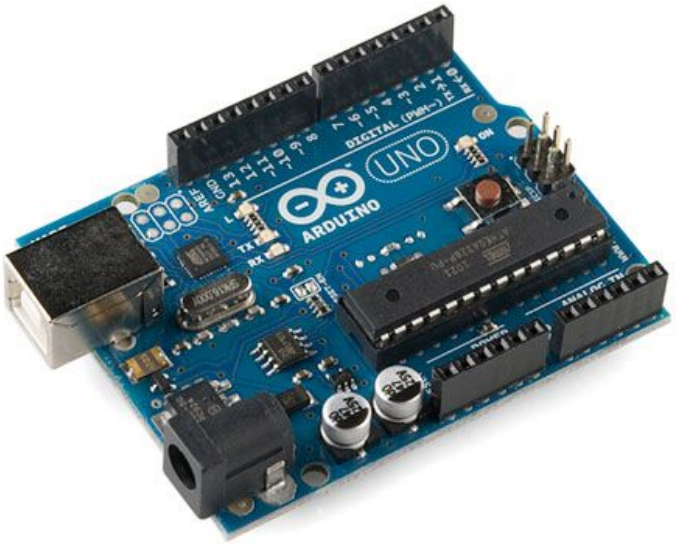


Figure 2.5: Arduino Uno development platform.

3. Implementation of the control module

In this chapter we talk about the process of designing the control module, along with the software and hardware that compose it. Section 14.1 discusses the initial conditions with which the project started. Section 14.2 describes the setup of the data acquisition system that is used to facilitate communication between the robot and the control system. Lastly, section 14.3 discusses the program developed to monitor and control the robot arm.

The control module must be able to perform the following tasks:

- Monitor the position of the motors that make up the robot arm at all times.
- Handle the power electronics needed to power the robot's motors.
- Handle the interruptions generated by the robot switches.
- Allow the user the choice between either automatic or manual operation of the robot.
- Perform user-defined routines.

The control module to be implemented is shown in Figure 3.1.

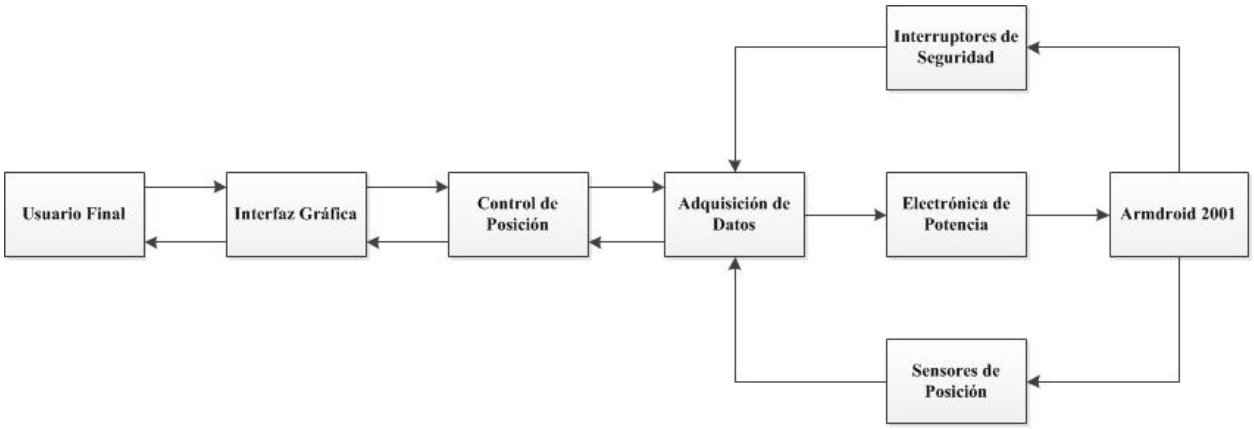


Figure 3.1: Schematic of the control module.

3.1 Initial implementation phase

In the initial phase of the project, the operation of the ARMDROID 2001 robot arm was examined in order to detect possible faults and to determine the causes of them. From the examination it was found that two of the DC motors were not functioning correctly because the gears connecting the motor shaft to the gear system of the robot arm were not properly coupled to the motor shaft, This caused the gear to slide on the motor shaft when it was being fed, causing the desired movement not to take place.

Figure 3.2: DC motors used by the ARMDROID 2001.

The DC motors used by this robot arm are Pittman 24V brushless motors with permanent magnets. Figure 3.2 shows a motor similar to those used by ARMDROID 2001. It was observed that these DC motors have a HEDS 9100 encoder, with which it is possible to detect the position of the robot. This encoder is located on the back of the motor. It was found that one of these encoders did not work on one of these motors.

It was also noted that the piece shown in Figure 3.3 did not work properly, this piece is responsible for marking the end of stroke of the shoulder on the robot arm.

Figure 3.3: Part used for the shoulder stroke.

Once these faults were detected, they were immediately serviced. Firstly, the motor gears were replaced, and one of these motors was replaced with one that has a less pronounced "dead zone". Finally damaged encoder was replaced with a fully functional one. It is important to note that a replacement could not be found for the part shown in Figure 3.3. However, it can be constructed using a 3D printer.

The ARMDROID 2001 robot arm is a spherical arm. That is to say, the combination of all the movements that it performs form a sphere. Figure 3.4 details the parts that this robot is able to move, making use of it's six DC motors.

By making use of this force, the motor movements were described as follows:

1. Base
2. Shoulder
3. Arm
4. Hand
5. Clamp

It is important to mention that the robot arm hand is able to make two movements: rotate and bend. In this way you get the six axes of movement with which the robot is able to form a sphere.

At the end of this phase a position control system is chosen using the encoders as position sensors, an Arduino One as a data acquisition system and as the output of the microcontroller to a graphic interface. LabVIEW is chosen for the control software because it facilitates both communication with the Arduino and the creation of the graphic interface.

3.2 Hardware implementation

The hardware unit to be deployed must fulfill the following functions:

- Provide the robotic arm with the necessary power so that it can move without complications, depending on what the position control loop requires.
- Translate the signals from the various encoders and switches of the robot into signals that can be read and acted upon.

It is for this reason that the hardware unit is subdivided into two stages: an input stage and an output stage. The operation of both stages is detailed below.

Input stage

In the input stage is all the electronics needed to transmit the positions and switch signals that the robot arm needs to send to the computer. Figure 3.5 shows the a schematic diagram of said circuit.

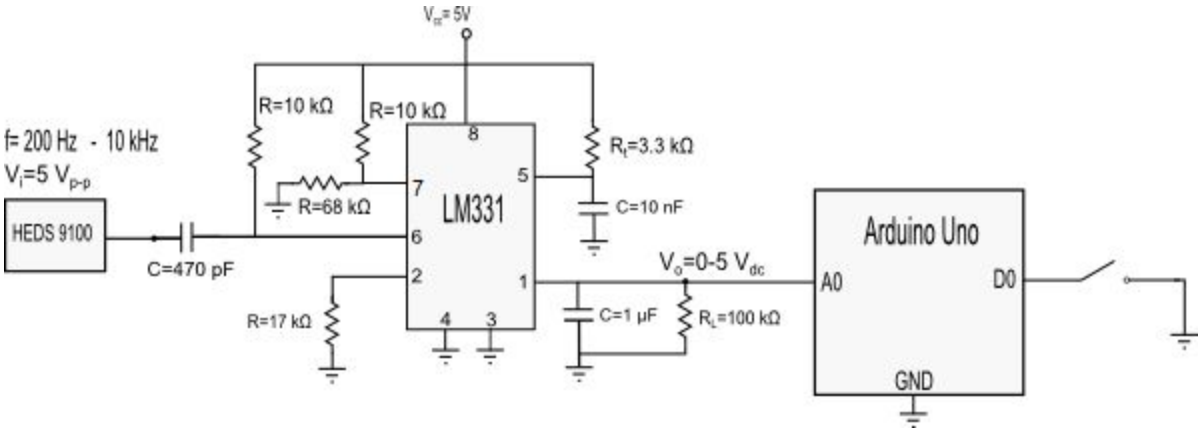


Figure 3.5: Entry stage for a single engine.

An LM331 is used to convert the variable frequency signal from the HEDS-9100 encoder into a variable DC voltage signal. This is done because the frequency range generated by the encoder for these motors (200 Hz to 10 kHz) exceeds the sample rate of the Arduino. The V_o voltage signal is read using the analog Arduino ports. It is important to note that this circuit should be used for each motor that owns the robot arm. The signal from the various switches is read using the Arduino's digital ports, as shown in Figure 3.5. It is important to note that the signal read by the Arduino is a signal proportional to the speed of rotation of the motor, so it is necessary to derive the position from this signal.

Output stage

In the output stage the PWM signal of the Arduino must be used to feed the various motors of the robot.

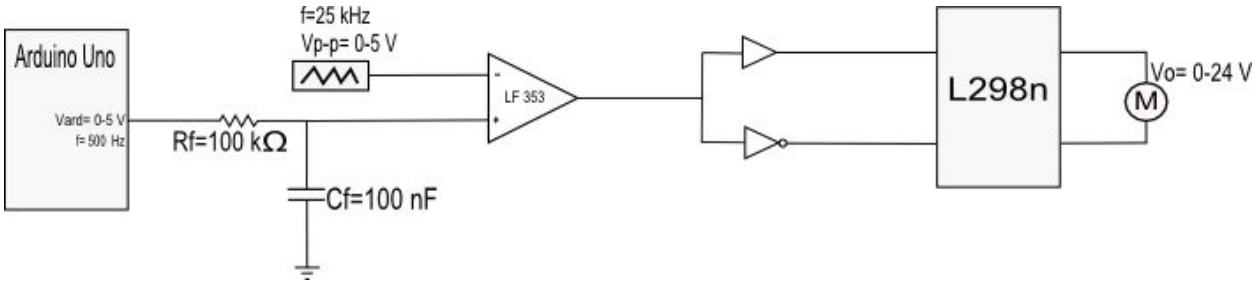


Figure 3.6: Output stage for a single motor.

Figure 3.6 shows the hardware unit implemented. The V_{ard} voltage signal is a 5V square signal with a variable duty cycle and a frequency of approximately 500 Hz. The capacitor C_f is charged at a value proportional to duty cycle D? Vard. This signal is fed to a comparator along with a triangular signal of 25 kHz, with amplitude of 5 V_{p-p}. This is done with the n to obtain a high frequency signal, since the frequency of the Arduino signal is very low, causing problems with current consumption due to motor impedance, also producing vibrations and sound in the motors These unwanted effects are eliminated by increasing the frequency of the motor power supply. The effects of motor impedance, vibrations and sound are completely eliminated. At the output of the comparator there would be a square signal, with a pulse width equal to the signal produced by the Arduino, and a frequency of 25 kHz. This signal is passed through a buzzer, Er and by an investor.

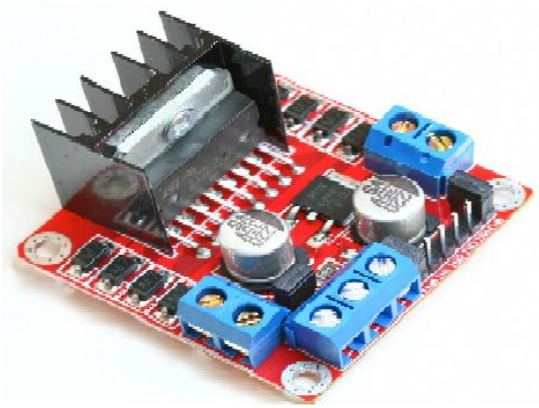


Figure 3.7: Integrated module of the L298n.

The L298n is an integrated circuit containing two H-bridges. Due to its internal configuration this component requires two signals to control a single motor. To control the working cycle of a single motor the L298n must be Control signal, and the same signal denied, thus achieving bidirectional control of a DC motor. Figure 3.7 shows one of the three integrated modules that are required to implement the output stage of the hardware unit.

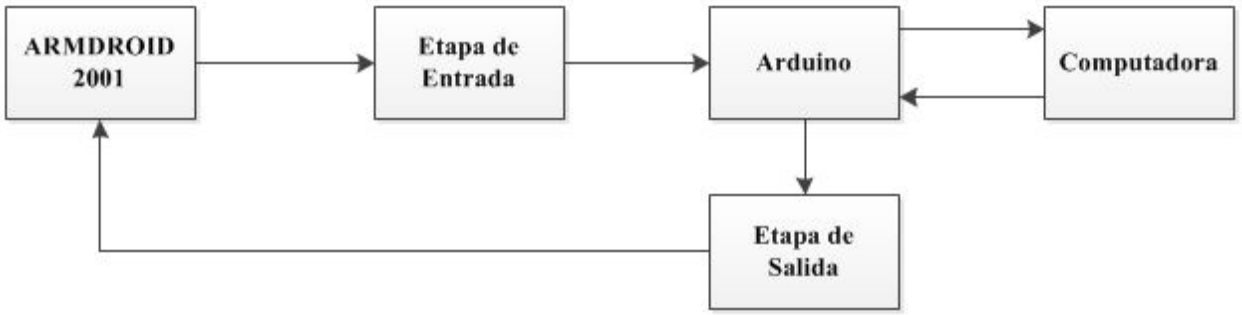


Figure 3.8: Schematic of the hardware unit for a single motor.

By joining the output stage and the input stage, the operating scheme shown in Figure 3.8 is obtained. In summary, the overall control module will be obtained by repeating the scheme shown in Figure 3.8 for each of the motors. It is important to note that the robot has four switches. That is, the robot arm has four non-permitted operating positions corresponding to the motors that control them.

3.3 Software implementation

Using the signals received by the Arduino the software program must perform the following functions:

- Implement the position control system.
- Monitor the status of the robot arm at all times.
- Take the safety measures required for the safe operation of the robot.

LabVIEW 2010 was used for the implementation of the software program, which is a development environment created by National Instruments to design systems using a visual programming language co. This program allows the creation of a graphical user interface for the user, as well as creating the control system and communicating with the Arduino, either to receive signals from the robot arm or to send voltage outputs with variable pulse width.

Figure 3.9 shows the block diagram of the software unit. The first task of the main program is to obtain a position signal from the speed signal obtained from the hardware unit. Once the position signal is obtained, the control loop is implemented, for which a PID controller with two degrees of freedom is used. Subsequently, the software unit must implement a graphic interface that allows the monitoring and control of the robot arm.

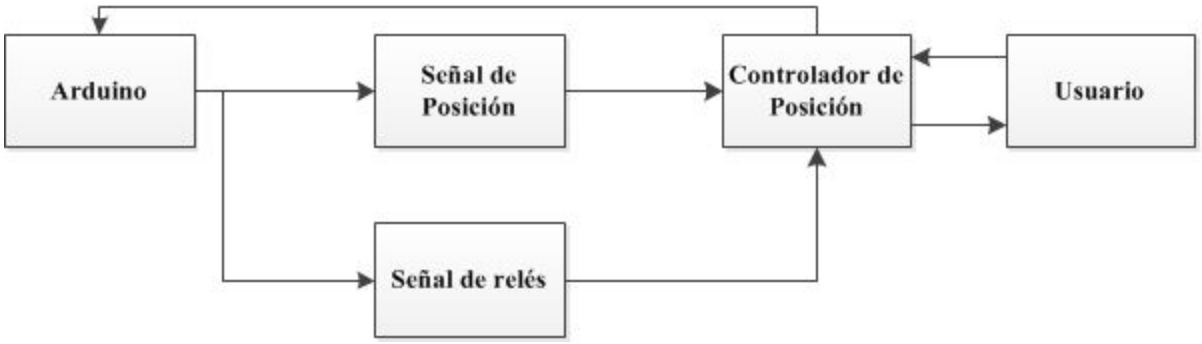


Figure 3.9: Block diagram of the software unit.

Position controller

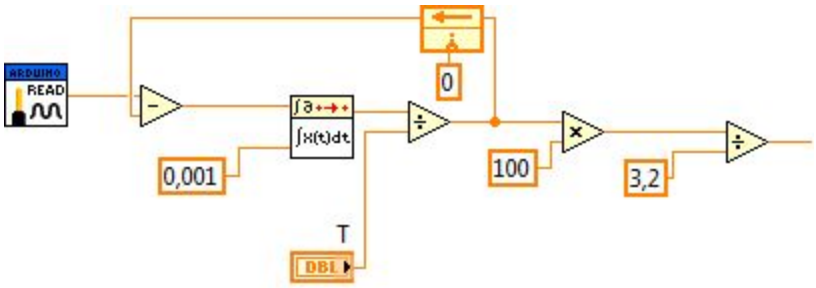


Figure 3.10: Filtering and normalization of the obtained signal.

The block diagram shown in Figure 3.10 shows (representatively) the way in which the voltage signal obtained is normalized. However, it must be taken into account that this signal will always have a positive slope. To solve this problem it is necessary to take into account that in the first instance, a 50% duty cycle marks the point at which a motor is at rest. A greater duty cycle implies that the motor turns in a direction, while a duty cycle of less than 50% indicates that the motor will rotate in the opposite direction.

From this fact, two D flip-flops are implemented as shown in Figure 3.12, which will receive as a clock signal a 50% crossing in the duty cycle (controller output). Figure 3.11 shows the block diagram from which the position signal is obtained, where the control signals receiving the flip-flops correspond to the crosses of the controller output by 50%. As a result of these control signals, either the positive slope signal or the negative slope signal is selected. The gray boxes contain a small script which limits the position signal between 0 and 100%. This is the resultant signal that enters the two degree of freedom controller. Finally the output of the controller is normalized and will be sent to the hardware output stage.

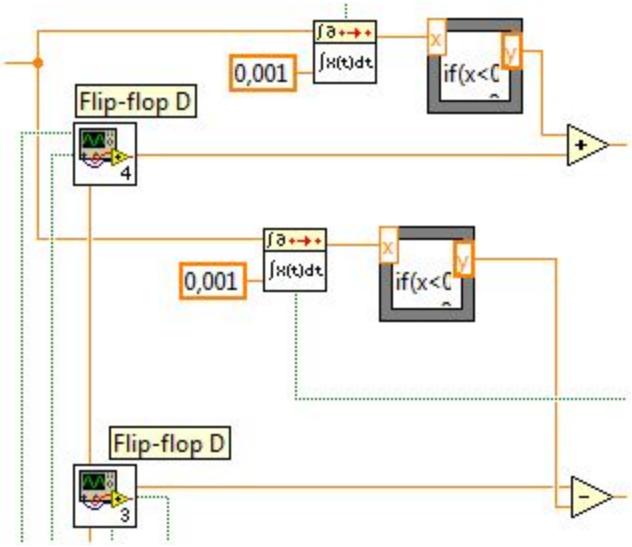


Figure 3.11: Obtaining the position signal.

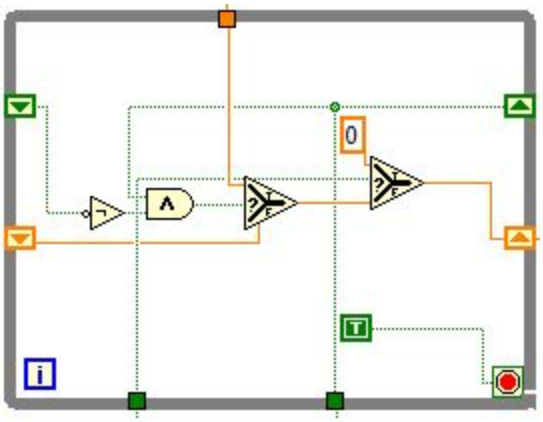


Figure 3.12: D flip-flop implemented.

For the safety stage the main program constantly monitors the robot arm switches, which are normally open. Once they are activated the main program will detect a zero, completely stopping the normal operation of the arm. In this condition, the program will not allow the limit position marked by these switches to be exceeded. It should be noted that the manual control loop performs manipulation on the output of the PID controller directly, i.e. the manual control loop manipulates the duty cycle that are sent to the motors.

Finally, in Figure 3.13, the LabVIEW graphical user interface is shown. The operation of the interface is detailed in appendix A.

In summary, the model of the control module for the robot arm can be simplified in the way it is seen in Figure 3.14.

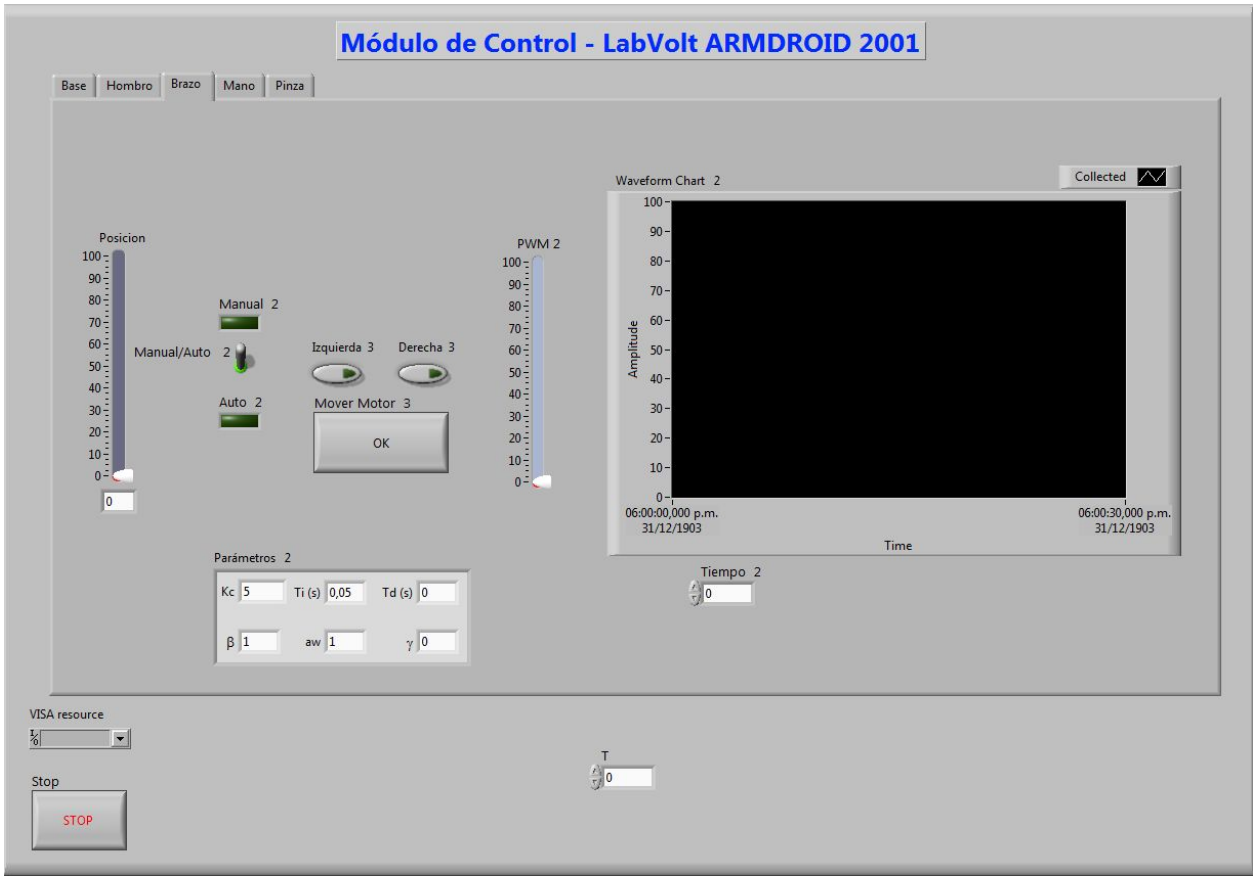


Figure 3.13: Grid interface in view of the control module.

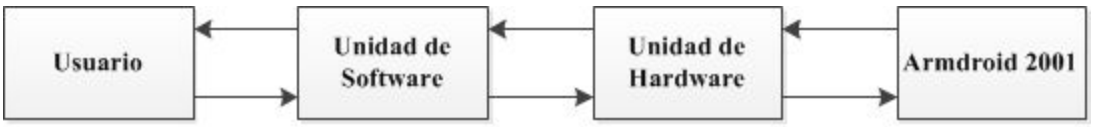


Figure 3.14: Block diagram of the simplified control module.

4. Validation of the control module

This chapter shows the results obtained when using the control module implemented to handle the ARMDROID 2001 robot arm.

In the first instance two open-loop tests were performed in order to obtain a model of the process. The results shown in Figures 4.1 and 4.2 suggest that the model is an integral plant irrespective of the part of the arm that is required to be moved. It is worth mentioning that for the open loop test performed on the arm, 100% duty cycle is applied to the output of the controller.

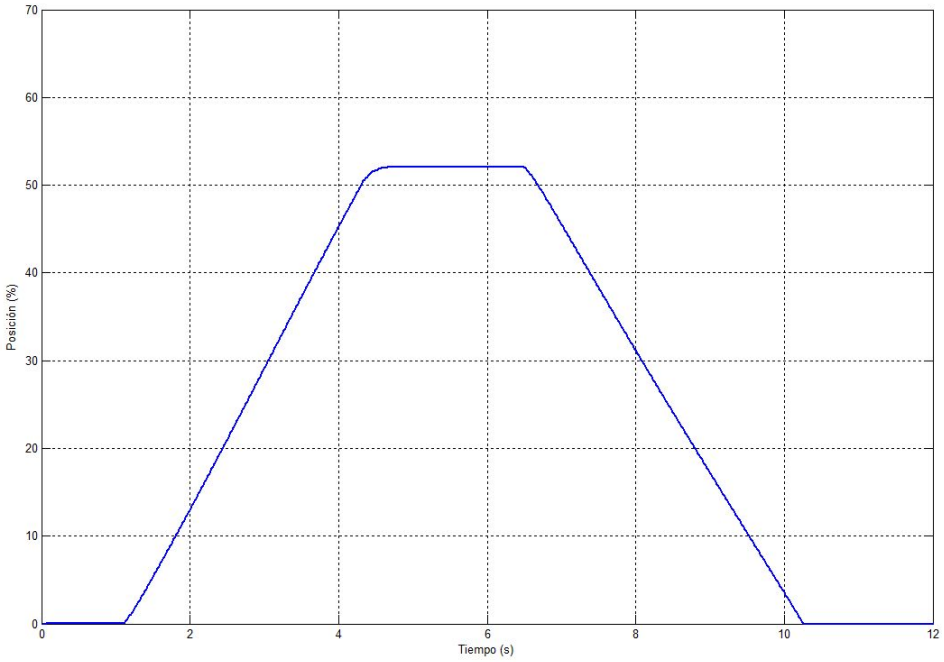


Figure 4.1: Open loop response of arm.

It is important to note that the two slopes differ. This suggests that gravity has some effect. However, the difference in responses can be considered negligible. From these open loop curves, models were created for both the shoulder and the arm which are shown in equations 4.1 and 4.2, which correspond to integral plants without timeout.

$$Pb(s) = 12.36 / s \tag{4.1}$$

$$Ph(s) = 29.43 / s \tag{4.2}$$

Figure 4.3 shows the closed-loop response of the motor that moves the robot's shoulder. It can be seen that the response tends to be of the first order. However it is possible to see that the system is not very entirely linear, due in large part to the force that the motor must exert to lower the shoulder of the robot arm, which would not be the same when it has to lift the shoulder. In this graph, the positive amplitude scaling represents the shoulder drop. We can see that this response is faster than that obtained for the negative scale. Since the dynamics of all motors is similar to that shown in figure 4.3, it can be said that the motors will have similar results. Of course, taking into account that in a sense a movement will be achieved Faster than in the opposite direction, this in function of the load that the motors must move.

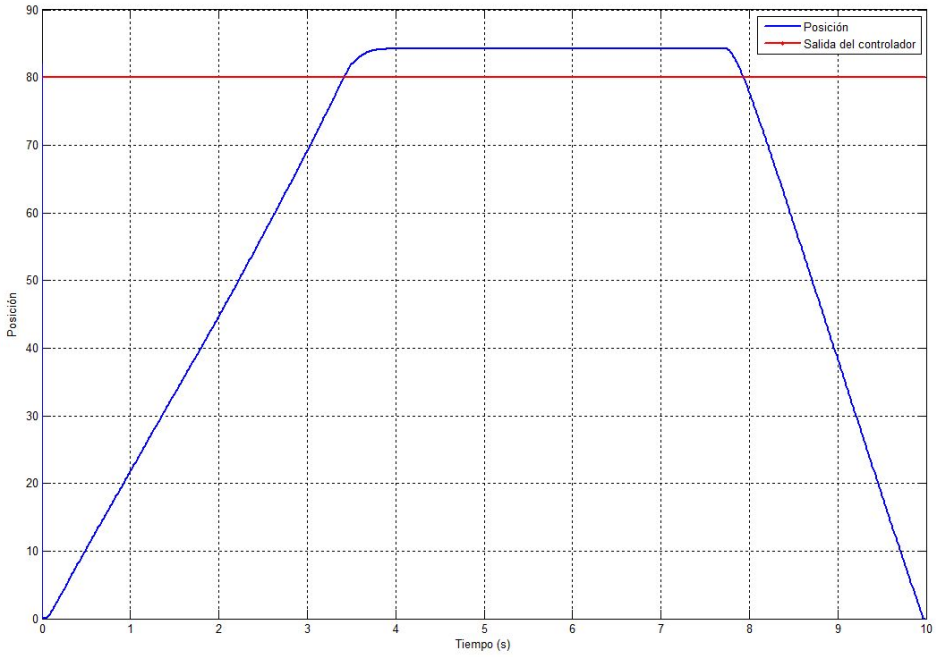


Figure 4.2: Open shoulder response

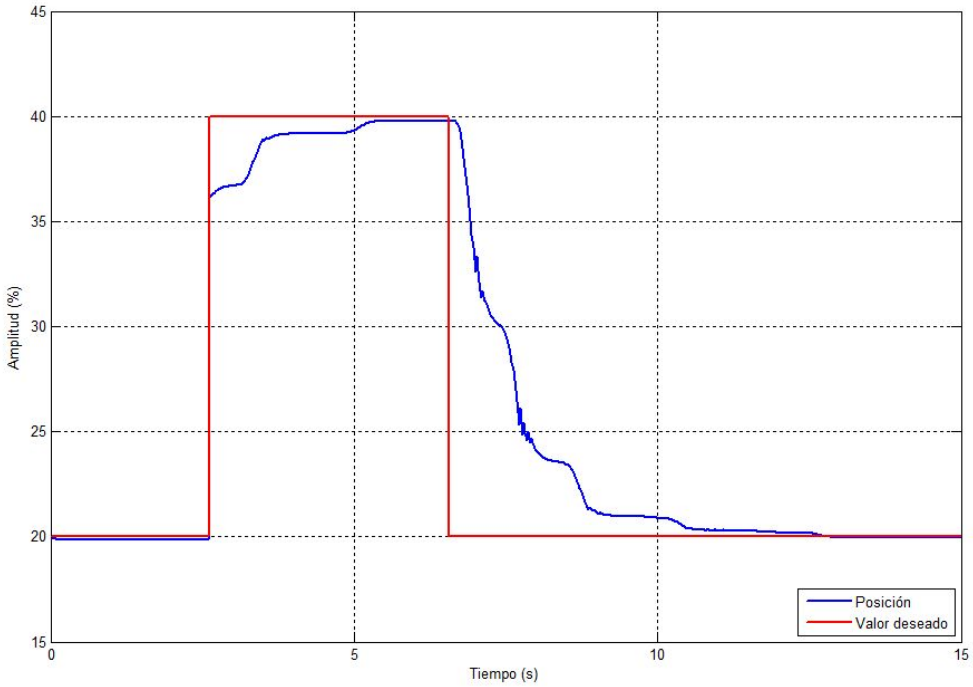


Figure 4.3: Closed loop response of the shoulder position control system.

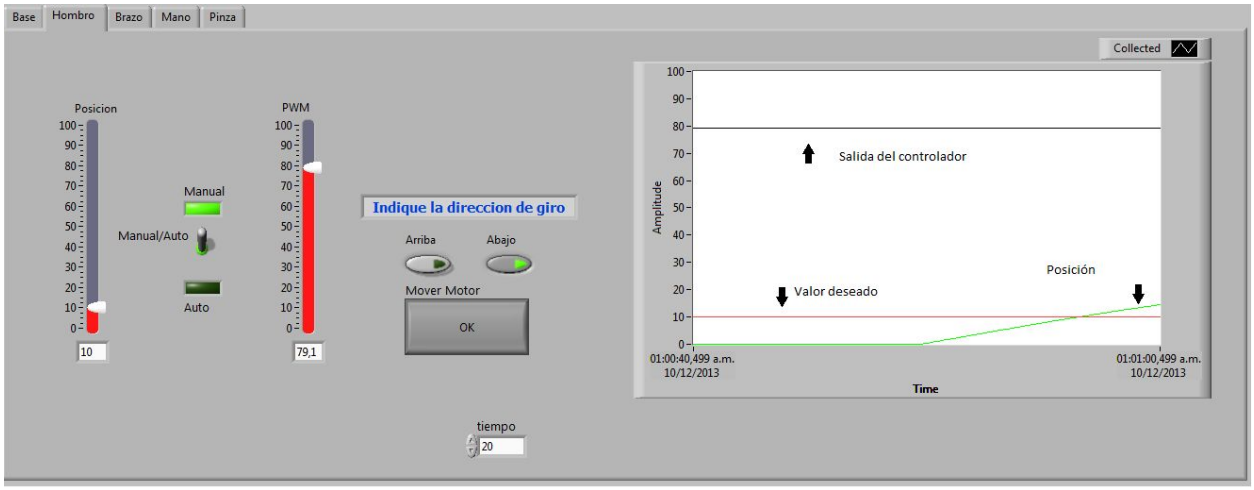


Figure 4.4: Manual control response.

Figure 4.4 shows the open loop behavior of the position control loop in conjunction with the interface operation. To facilitate the reading of the chart the background color is changed, however in the actual interface the output of the controller will be a white line, the desired value will be shown in red, and position will be shown in green. To move the shoulder downwards you

must select the direction and then press the "Move Motor" button, otherwise the desired movement can not be made.

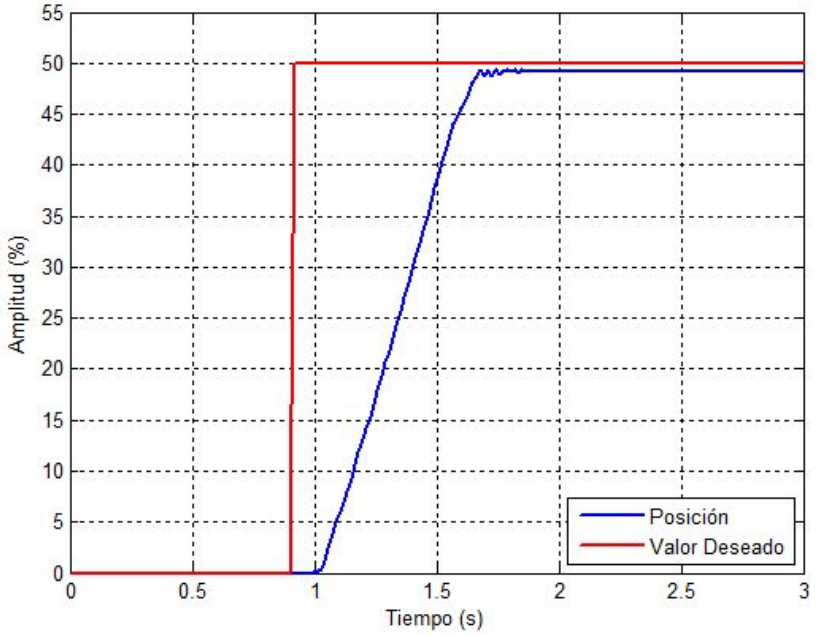


Figure 4.5: Closed loop response with a proportional controller.

Subsequently a proportional controller was tuned which guarantees a permanent error of less than 1% with fi = 1. The transfer function of this controller is shown in equation 4.3. Figure 4.5 shows the response obtained.

$$C(s) = 3.36 \tag{4.3}$$

It can be seen how the response is greatly improved in terms of its speed and the reduction in oscillations. However it is recommended to use a proportional-integral controller that ensures a permanent null error.

Finally, the operation of the implemented safety system must be checked so that if the arm is moved to an unallowed position the stroke switches will be activated, displaying the message shown in the Figure 4.6. To perform this test the switch on the shoulder was manually activated.

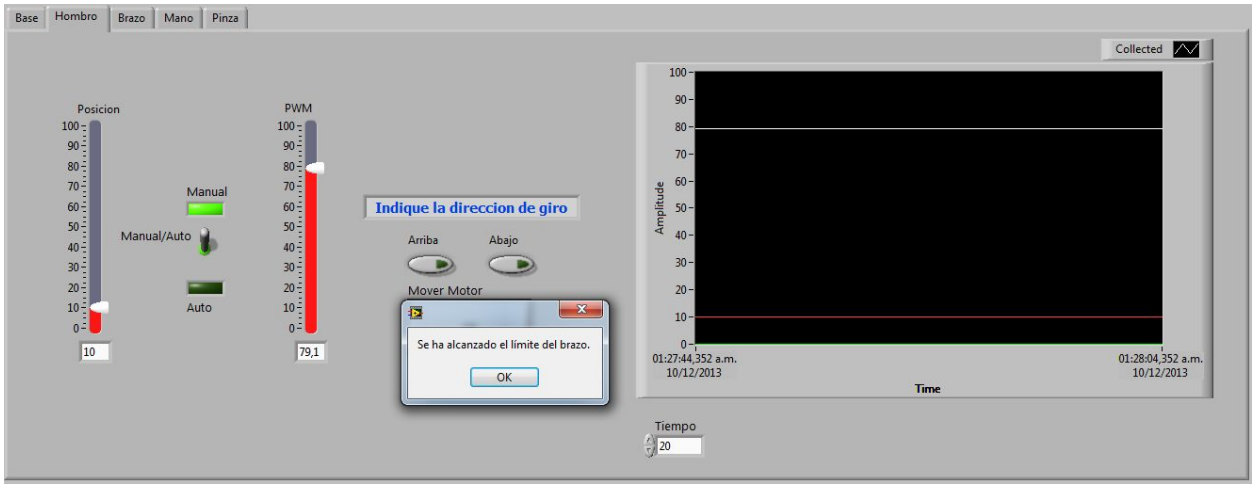


Figure 4.6: Error message displayed by the program.

5. Conclusions and recommendations

5.1 Conclusions

- They were able to repair and to replace the parts of the robot arm so that the arm can operate properly. However, the only part that does not have a replacement is shown in Figure 3.3.
- It was possible to implement the necessary electronics to be able to perform a position control for the various motors of the robot arm.
- The sampling frequency of the Arduino Uno is so low that it is required to implement an electronic stage that converts the signals from the position sensors into readable signals by the Arduino.
- The safety signals from the switches were properly integrated.
- A graphical user interface was created with which the state of the ARMDROID 2001 can be monitored at any moment, as well as manipulation of the arm in either automatic or in manual mode.
- It was able to meet the objectives established at the beginning of the project.

5.2 Recommendations

- In order to avoid the use of an input electronic stage, a data acquisition platform with a higher sampling frequency can be used.
- The signal to be fed to the motors must have a frequency equal to or greater than 20 kHz, this will ensure that the effects of the impedance of the motors are minimized with this frequency being reduced to the consumption current in the motors when they are in steady state.

A. User manual

This appendix details the aspects to be considered for the start-up and operation of the robot arm. For the start-up, the various connections to be made are discussed, while for the operation the use of the program, as well as the the various elements that make up the graphic user interface.

A.1 Start up

To start the robot arm, run the main program, called control module. The hardware units must be supplied with 5V and -5V in the case of the comparator, and with 5V and ground in the case of the inverters and the LM331. It is important to note that the Arduino ground references and the hardware unit ground references should be interconnected in order to obtain coherent logical values. The comparators must be fed with the PWM signal from the Arduino and with a triangular signal of 25 kHz with 5 V_{p-p}. It is important that this signal does not have a set in the voltage, since this may alter the result of the comparison. Figure A.1 shows a basic schematic of the Arduino connection to the hardware units.

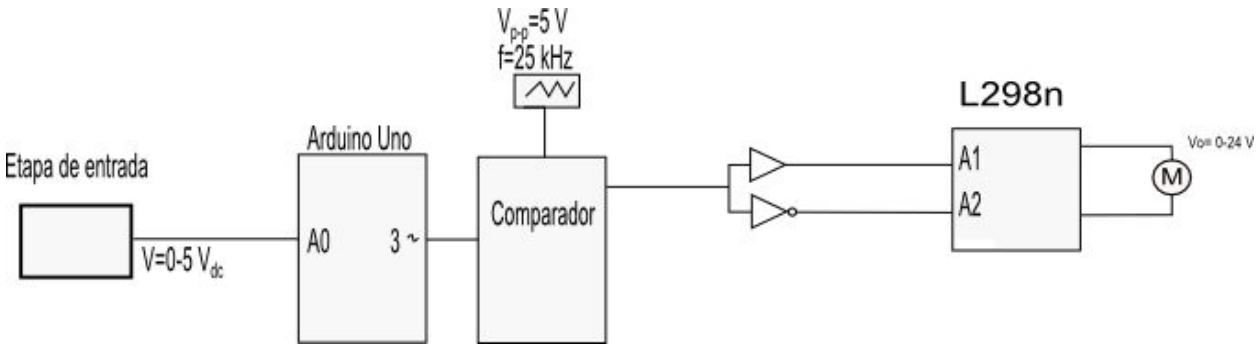


Figure A.1: Connecting the Arduino to the hardware stages for a motor shaft.

Note that the hand requires two analog inputs and two PWM outputs due to the rotational and bending movements. Table A.1 shows the connections that must be made to start the robot arm.

Table A.1: Relation of connections

Movement	Analog input	PWM Output	Digital input
Arm	A0	6	D8
Shoulder	A1	9	D12
Pin	A2	10	-
Hand (bend)	A3	5	D7
Hand (rotate)	A4	3	-
Base	A5	11	D4

A.2 ARMDROID 2001 robotic arm operation

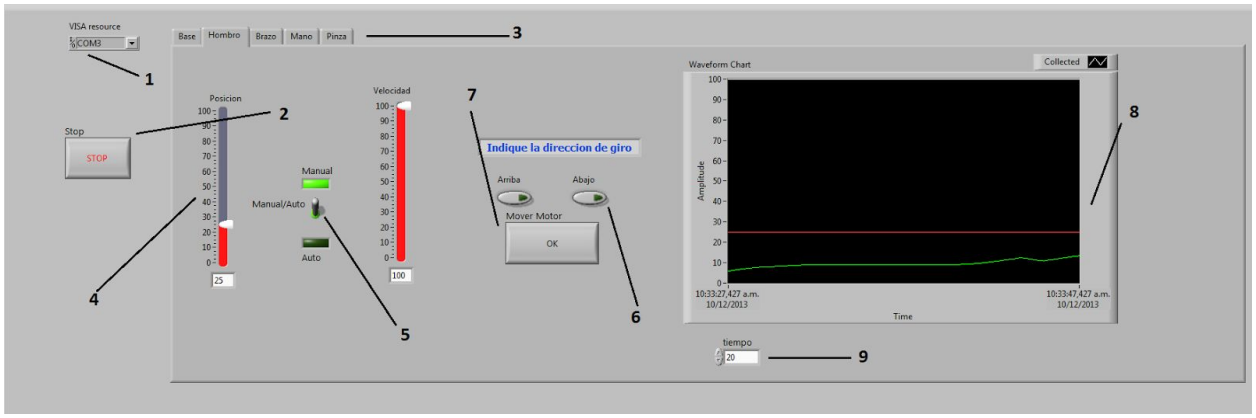


Figure A.2: Grid interface of the control module.

In this figure all the elements necessary to operate the program in an appropriate way are indicated.

1. Port Selector: This selector selects the port on which the Arduino is located.
2. Stop Run: Stops running the program.
3. Monitoring clocks: in these tabs the part of the robot that you want to monitor and control is chosen.

4. Desired value selector: this bar specifies the desired position to which you want to move a certain part of the robot arm
5. Automatic or manual mode selector: it allows to change the mode of operation from automatic mode to manual mode or vice versa.
6. Direction selectors: these buttons are only enabled in manual mode. They allow the user to choose the direction in which it is desired to move the part of the robot that is being controlled.
7. Move motor: holding down this button activates the movement of the arm. Releasing it will stop the robot.
8. Monitoring status: this monitor monitors the status of the robot arm. The red line indicates the desired position (desired value), the white line shows the output of the controller and the green line represents the actual position of the part of the robot that is monitored.
9. Grid display time: This constant specifies the sampling time interval in seconds that you want to display in the monitoring chart.

B. Developer's manual

In this appendix the most important blocks for the operation of the position control loop, as well as the structure of the program itself are mentioned.

B.1 ARMDROID 2001 robotic arm operation

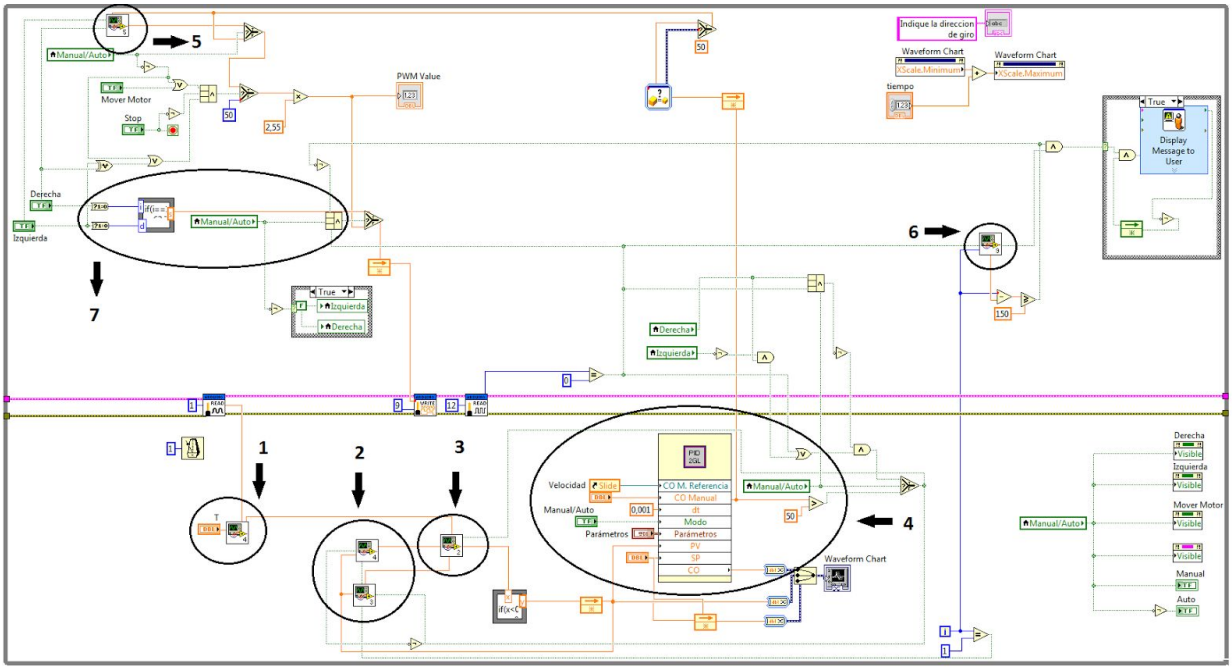


Figure B.1: Program required to control a single engine.

Figure B.1 shows the code required to control a single motor. In order to facilitate reading and understanding of the program, and in order to minimize the size of the program, several custom subblocks are used. The blocks of interest are shown in the figure.

1. In this block the voltage signal read by the Arduino is filtered and normalized. The constant T indicates the sampling period for the internal integral (see figure 3.9).
2. These blocks are two D flip-flops required to obtain the position signal as mentioned in Chapter 3.
3. The structure of this block is shown in Figure B.2.

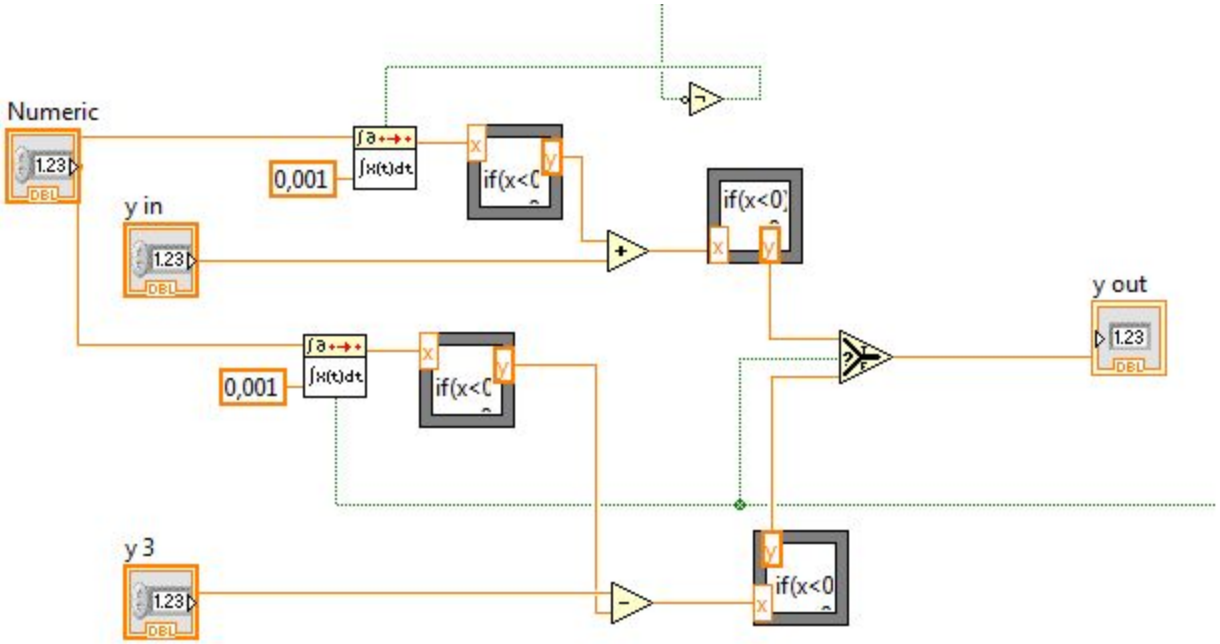


Figure B.2: Obtaining the position signal.

From the control signals received by this block, a curve with positive or negative slope will be obtained. Depending on the case, the position signal received by the controller is obtained in this block.

4. This block is the PID controller with two degrees of freedom. At the output of this block are generated different control signals necessary for the operation of the block used to obtain the position signal received by the controller. For this reason It is recommended to handle these signals with care.
5. The function of this block is to choose between the automatic control and the manual control. It is also responsible for taking the percentage signal from the controller and transform it into a signal that can be written at the PWM outputs of the Arduino, which is an integer number between 0 and 255. Figure B.3 shows the internal structure of this block.
6. This block is a D flip-flops used to generate the error message after a certain amount of iterations in case the monitored switch is activated. 150 iterations are selected (approximately 2 seconds), however the corrective action in the robot operation will be performed as soon as the switch is detected.

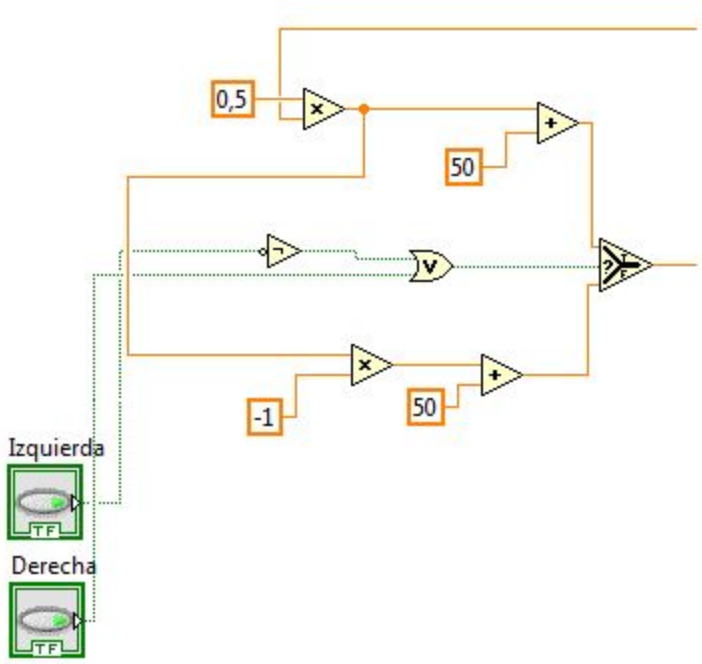


Figure B.3: Automatic / manual control block.

7. In this section of the code the corrective action is taken once it has been verified that the monitored switch has been activated, Figure C.2 shows the internal structure of the code located in the gray box .

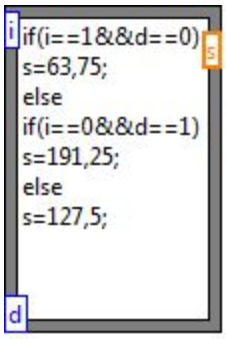


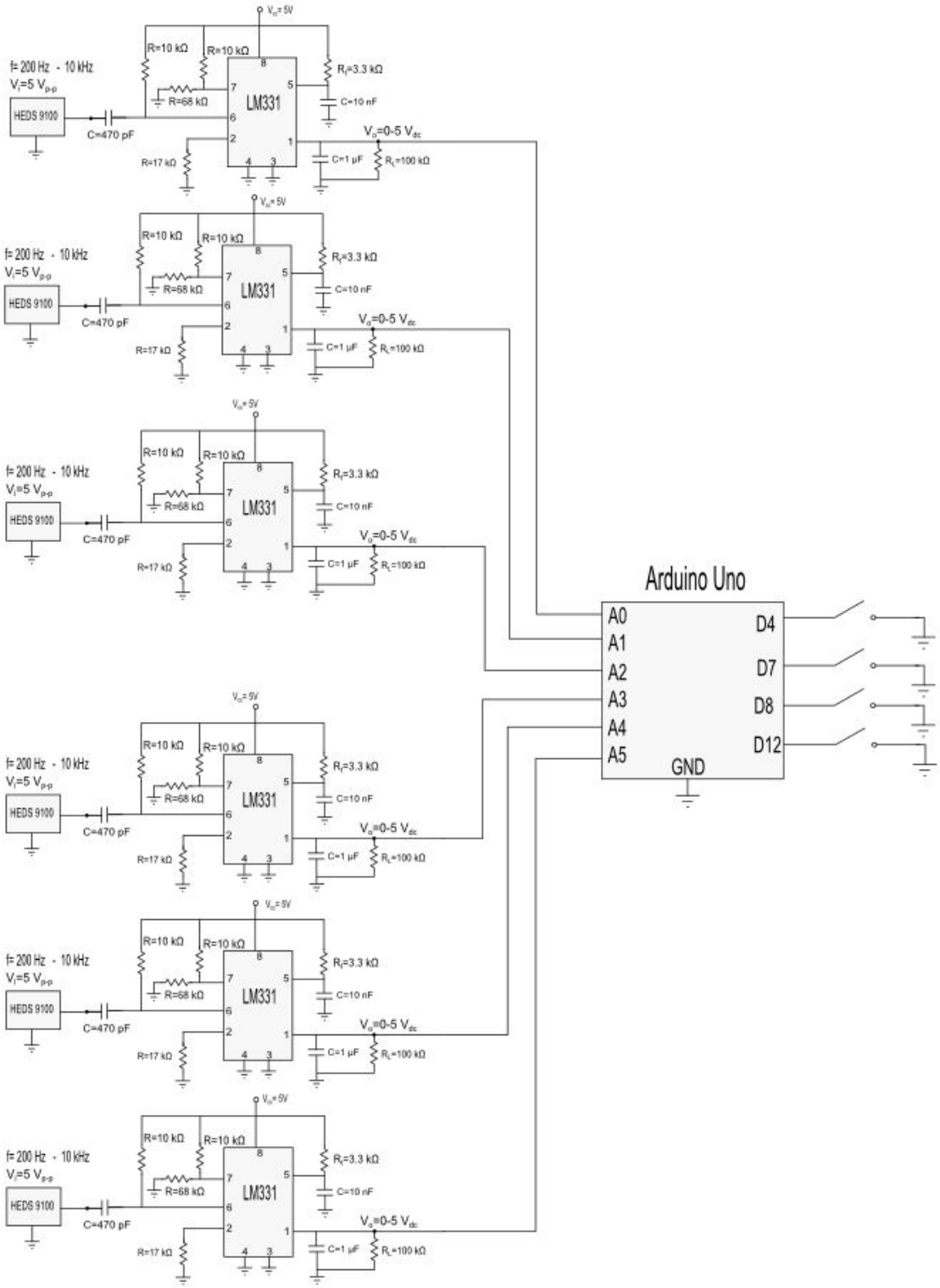
Figure B.4: Code used to take corrective action.

C. Schematic diagrams

Taking into account that all the electronics developed for the input and output stages can be implemented on a breadboard, the complete schematic diagrams of the input and output stages are presented. Table C.1 shows the components required to implement these circuits.

Table C.1: List of components

Quantity	Component	Value
1	Arduino Uno	-
1	USB A-B cable	-
6	LF353	-
3	L298n H-bridge	-
6	NOT comparators	-
6	Buffers	-
6	LM 331	-
6	Capacitors	470 pF
6	Capacitors	10 nF
6	Capacitors	100 nF
6	Capacitors	1 uF
12	Resistors	10k
6	Resistors	68k
6	Resistors	17k
6	Resistors	3.3k
12	Resistors	100k


f= 200 Hz - 10 kHz
V_i=5 V_{p-p}
HEDS 9100
C=470 pF
R=10 kΩ
R=10 kΩ
R=68 kΩ
R=17 kΩ
LM331
R_f=3.3 kΩ
C=10 nF
C=1 μF
R_L=100 kΩ
V_o=0-5 V_{dc}
Arduino Uno
A0
A1
A2
A3
A4
A5
GND
D4
D7
D8
D12

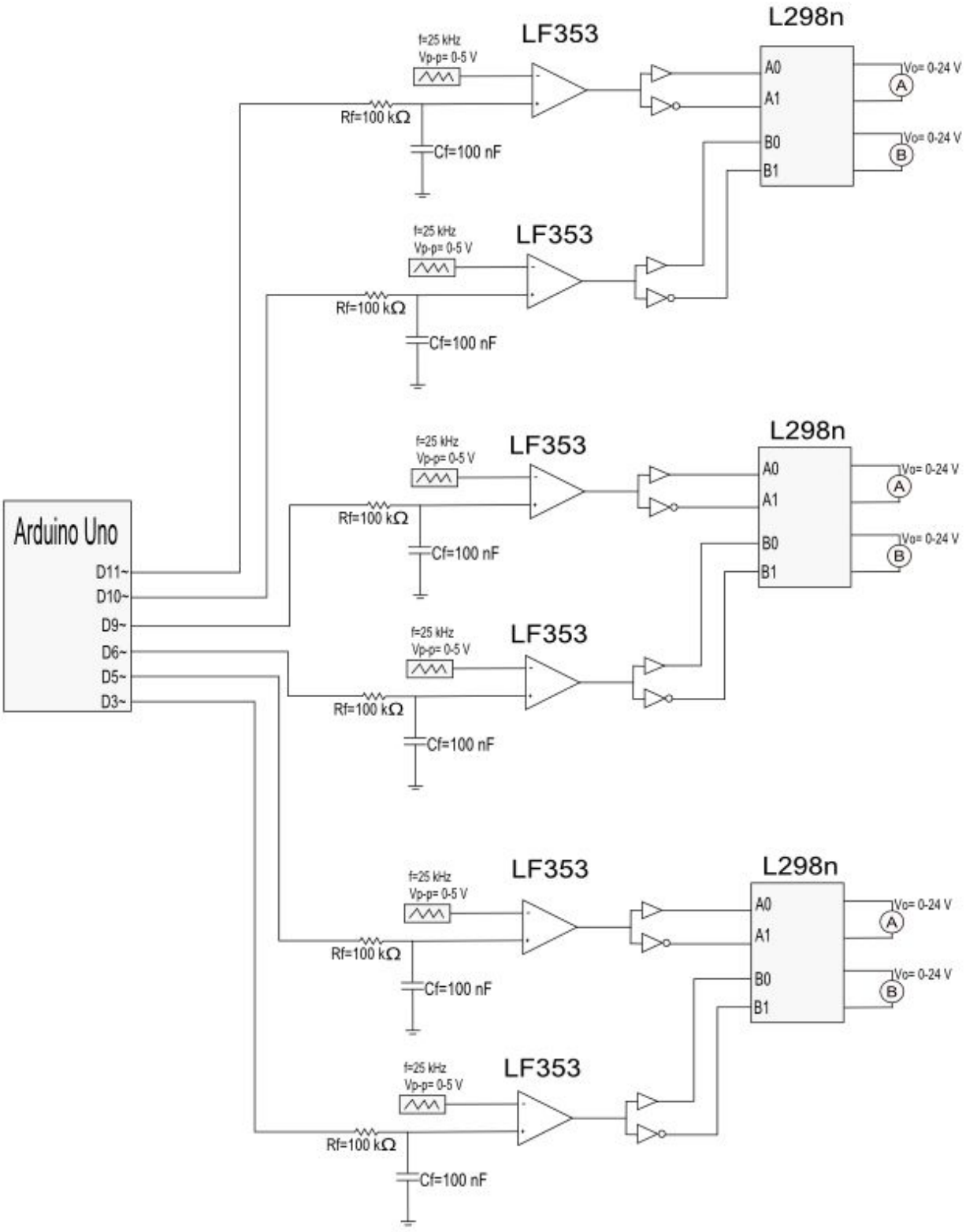


Figure C.2: Complete schematic diagram of the output stage.