POLITECNICO DI TORINO

MASTER's Degree in MECHATRONIC ENGINEERING

Politecnico di Torino

MASTER's Degree Thesis

Development of a closed-loop control system for an airborne gimbal camera

Supervisor	Candidate
Prof. Sabrina CORPINO	Mirco VINCIGUERRA

DECEMBER 2021

Alla mia famiglia

Abstract

The SmartBay platform is an invention of Digiky Srl. The company has developed a wing pylon capable of hosting various types of sensors to carry out aerial monitoring. The SmartGimbal is a motorized metal structure designed to hold any type of video camera, equipped with two actuation systems that allow PAN and TILT movements. Being airborne, the movement commands must be given in remote.
The topic covered in this thesis consists of the development of the software so that the system meets the desired requirements. The used approach is the Model-based one concerning embedded systems, as it allows the generation of the C/C++ code of a model designed in MATLAB/Simulink and its actual implementation in a microcontroller, in this case, Arduino Mega2560. One of the project scopes is the system calibration, which automatically activates a motor until the Hall effect sensor finds the magnet. The designing of a closed-loop system for speed control of a DC motor using an Encoder such as a feedback sensor is the main part, moreover, the serial communication between two Arduino boards has also been performed. This last argument relates to the fact that the control console, containing the joystick for managing the movements, sends signals to the actuation part and, vice versa, this sends the data to be shown on the display.
The parts making up the software have been detailed, with the wiring diagrams showing the connections made and a list of the components used reporting their technical specifications.
In conclusion, thanks to the design of an integrated circuit board and the camera connection, the support for the video cameras has been made operational and therefore, it can be mounted under the wing of an airplane and controlled via a console, located inside the cabin.

Table of Contents

List of Tables

List of Figures

Acronyms

IMU Inertial measurement unit

PTZ Pan Tilt Zoom

MIL Model In The Loop

SIL Software In The Loop

PIL Processor In The Loop

HIL Hardware In The Loop

LTI Linear Time-Invariant

PWM Pulse With Modulation

DAC Digital to Analog Converter

PCB Printed Circuit Board

Chapter 1

Introduction

1.1 Digisky company

Figure 1.1: Digisky Logo

DigiSky Srl works on aerial ground monitoring technologies for Earth preservation, emergency management, fire detection, surveillance, and asset-intensive companies. Technology innovations are applied in Avionics Systems, Special Mission Aircrafts, and advanced Aerial Monitoring Projects. The aim is thus to transfer innovative technologies from the ICT and automotive sectors to the General Aviation industry, offering embarkation solutions that are easy to install, low costs, and can be applied in several different fields.
They provide customers with turnkey integrated services to qualify in-flight avionic systems during R&D and design phases, standardized system for the fast boarding of sensors, low-cost aerial ground monitoring systems to be applied in precision farming, fire detection, remote medical emergency mission, surveillance, and asset-intensive companies. To facilitate this process, DigiSky has its fleet of airplanes equipped with avionic systems to guarantee in-flight tests of different types of technology [1].

1.2 SmartBay Project

The flagship product is a platform created and developed in-house, called SmarBay. This is a carbon fiber pylon located under the wing of an airplane, designed to have a low aerodynamic coefficient and to support heavy loads up to 40 kg. In addition, it is equipped with sensors as IMU and GPS.
It has been developed to perform rapid sensors suite configuration on different flight platforms, standardize and automate the sensors management during the mission, facilitate post-processing operation of the data collected, optimize the sensor maintenance operations. Each module is designed to be plug and play and could contain a stabilized camera, a chemical sensor for air quality monitoring, an audio sensor for noise control, or any other solution.
Moreover, a feature of this structure is the capability of boarding up three different modules at the same time [2].

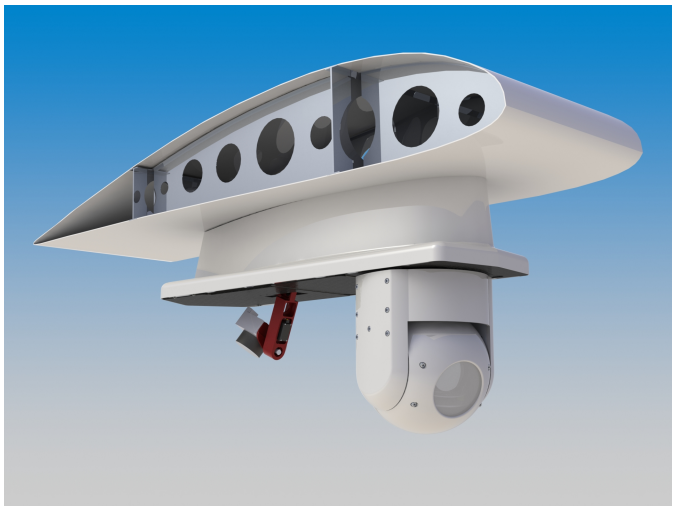

Figure 1.2: SmartBay pylon

The figure below shows the application of this platform, the aircraft in question is a Tecnam P92, a single-engine light aircraft with high-wing technology.

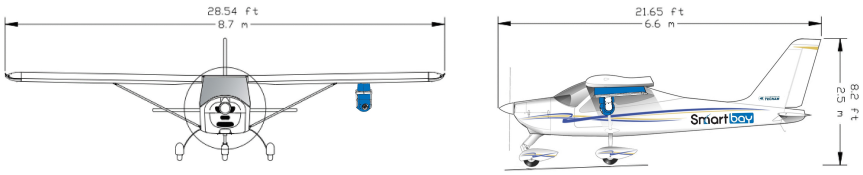

Figure 1.3: SmartBay allocation

1.3 State of Art

A particular module compatible with the SmartBay platform just described, is the SmartGimbal, a project that is also developed and built internally and is positioned on a carbon fiber trolley equipped with connectors to ensure the possibility of plug and play installation.
Conceptually it is similar to a PTZ camera, those found in the video surveillance sector, due to the two axes of movement and the remote control. The substantial difference, apart from the size, is the separation between motion management and image capture, due to the interchangeable sensor. This type of support has been designed to hold professional video cameras weighing up to 3 kg.

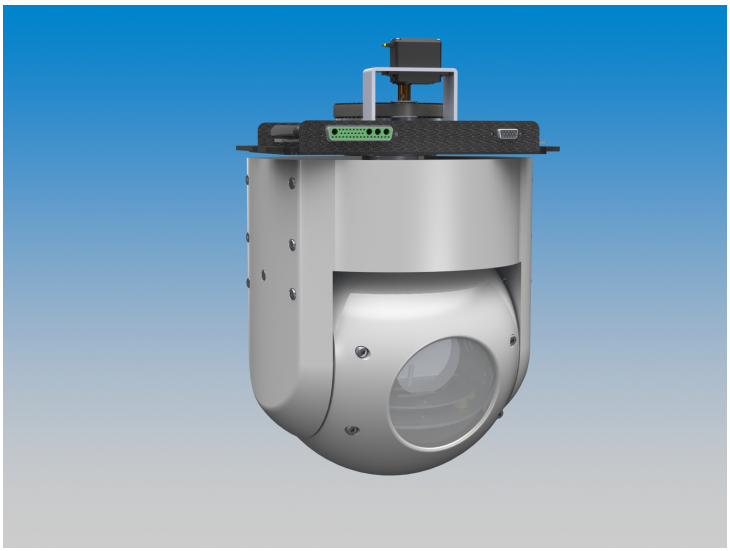

Figure 1.4: SmartGimbal render

The structure of the SmartGimbal, shown in the previous figure, has been the subject of past studies. The obtained results have been to lead the creation of a mathematical model of the gimbal system governed by the equations of dynamics. Thanks to simulations in the MATLAB/Simulink environment, were also found the maximum torque and angular velocity values of DC motors can provide. Finally, the calculations were performed to obtain a PI controller capable of controlling the actuation system. The C code related to this component was generated, so that could be installed on Arduino, allowing to verify the compatibility between the software and the embedded processor. This last phase is called Processor-in-the-Loop, according to the Model-based approach, which will be shown in detail in the following chapter [3].

1.4 Thesis objective

The main argument of this thesis is the development of control software for SmartGimbal and its implementation to obtain a functioning system.
The structure consists of a metal skeleton equipped with a system of gears and belts that allows the camera to move on two axes. The continuous rotation around the Pan angle is guaranteed by the presence of a slip ring while the inclination must be limited between ± 90 degrees to avoid cutting the cables of inserted camera.
Each axis is equipped with a DC motor coupled to an encoder, the first component generates the movement, the second is a sensor that allows to obtain the position of the system in degrees and create a local reference frame.
Commands must be managed via a remote control equipped with a joystick, a display, and buttons. This console must be able to be located inside the cockpit. The working environment is that of MATLAB/Simulink, as allows excellent interfacing between the block diagrams logic and Arduino, the chosen microcontroller. For this project, two Arduino Mega2560 boards were used that communicate with each other via a serial communication protocol. One manages the motor driver, their control loop, encoders, and Hall sensors while the secondary, located inside the console, controls the command components.

Figure 1.5: SmartGimbal mounted on the airplane

Chapter 2

Model-based approach

2.1 Basic knowledge

Nowadays, systems design is changing, and the Model-Based approach is spreading. This method is intended to increase the simplicity of the design process of complex control systems, signal processing, and communication systems, while also achieving an increase in productivity. It is widely used in many motions control, industrial, aerospace, and automotive applications [4].
This methodology is also applied in the design of embedded software. According to the software development model, called V-model, each phase of software development can be related to its test phase. This process differs from the waterfall model because instead of descending in a straight line, after the programming phase, it rises in the typical V shape, as shown in the next figure.

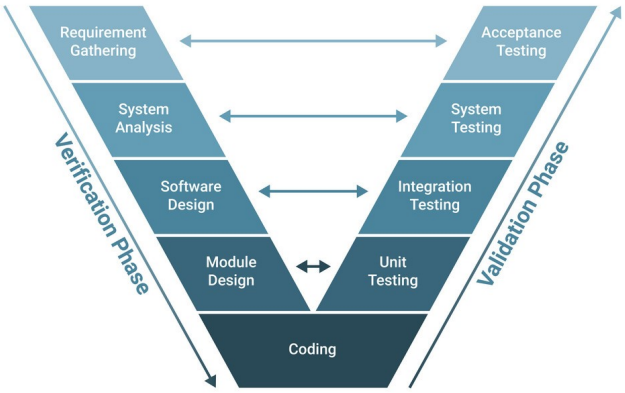

Figure 2.1: V-model architecture

The left side invokes the verification phase, in which a static analysis is performed without executing any type of code. On the other hand, in addition to dynamic analysis, tests are performed on the code to determine if it meets the customer's expectations and needs.
Before the model is deployed to hardware for production, a few verification steps are performed. MIL, SIL, PIL, and HIL tests are simulations, which can be performed with the Simulink environment and are part of the Model-Based Design approach.

Model-In-The-Loop: First, a real plant model in a simulation environment is needed. After that, it is moving on to the controller model development phase and check this can control the plant according to the requirements. This step is called Model-in-Loop (MIL) and the controller logic is tested on the simulated model of the plant. If the controller works as desired, it is needed to register the controller input and output that will be used in the verification step.

Software-In-The-Loop: Once the model has been verified in the MIL simulation, the next phase is Software-in-Loop (SIL), only code of the controller model is generated which replaces the block with the same name. A simulation will be performed with the Controller block (containing the C code) and with the Plant, which is still the software model (similar to the first step).
This step is useful to check if the controller model can be converted to code and if it is implementable on hardware. If there is a difference between input and output, it is necessary to go back to MIL and make any changes. If, on the other hand, the performance is acceptable, it is possible to move on to the next step.

Processor-In-The-Loop: The Processor-in-the-Loop (PIL) test consists of deploying some controller code on an embedded processor. By running a closed-loop simulation with the simulated plant, the controller subsystem can be replaced with a PIL block, it will be possible to identify if the processor can execute the developed control logic. If there are any problems, it will be necessary to go back to the previous steps to correct them.

Hardware-In-The-Loop: In a further step, is possible to have a simulated plant model in a PC in real-time before interfacing with the real hardware, in which there will be proper analog communications between the Plant and the Controller. This step is known as the HIL test, the controller is typically located on a production board/controller. It serves to identify communication channel problems, such as attenuation and delay that could be introduced by an analog channel and can make the controller unstable. This step is typically performed to test safety-critical applications [5].

2.2 Preliminary simulations

In the work carried out in the past, the so-called representation in the state space was obtained, this was used to know the dynamics of the system to be controlled. Unlike describing a system via a transfer function, this method made it possible to no longer consider the system as a black box, since its internal components are known exactly.
The state-space model is a mathematical method to represent the dynamics of a real physical system, through the relationship between input and output, defining state variables which, if correlated with each other, generate the n differential equations that led to the definition of the system.
The gimbal system was treated as an LTI system, (i.e., linear invariant-time), this was just an approximation around an equilibrium point to derive a linearized model. To motion control, it was decided to operate with a subdivision into two subsystems: the first dedicated to rotation around the pan angle, the second to rotation around the tilt angle. This decision was dictated by the fact that the actuators, which are the two DC motors, work independently of each other and therefore are controlled by two different controllers [3].
The C code of the control elements was also generated, to be able to perform the deployment on the Arduino board and test it with simulations in the MATLAB/Simulink environment, these steps are described above as SIL and PIL.

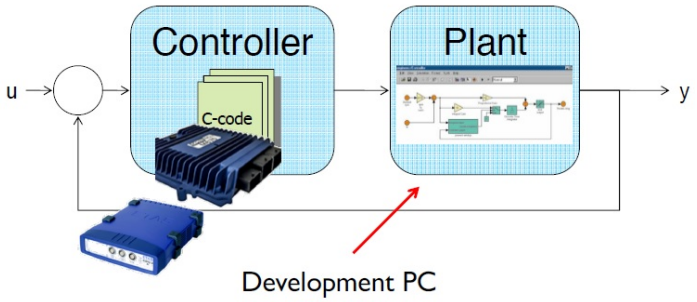

Figure 2.2: PIL block diagram

This was the starting point for the work done in this paper. It was possible to replace the model of the plant with the original hardware, a structure present in the laboratory dedicated to the execution of tests. This made it possible to deal with the interface between the microcontroller and the simulation environment but, the most important thing was the interface between the block diagram logic (and the code that is generated by it) and the physical used hardware elements, such as the various types of sensors or actuators.

2.3 Code generation

The MATLAB/Simulink workspace offers the possibility, thanks to toolboxes present within the program and external compilers, to generate and execute C and C++ code from Simulink diagrams, Stateflow graphics, and MATLAB functions to be implemented on the embedded controller.
The source code that is produced is generally readable, compact, and fast, can be used for real-time and non-real-time applications, including rapid prototyping and hardware-in-the-loop testing. Finally, it can be tuned and monitored in the Simulink environment or outside. Controls deployed on the real-time target can be used to communicate with Simulink using external mode and can also provide data exchange of process variables and block parameters to other applications [6].

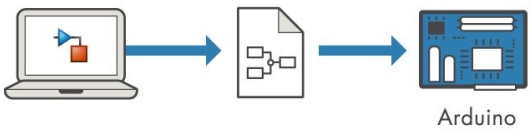

Figure 2.3: Deploying code on Arduino

2.3.1 Arduino and Simulink interface

First, a preliminary step is needed, include the Add-Ons to the MATLAB/Simulink environment. The operation is not complex and is shown in the following figure.On the Home screen, select "Get Add-ons" in the submenu, this will open an external page where is possible to search the desired toolboxes.

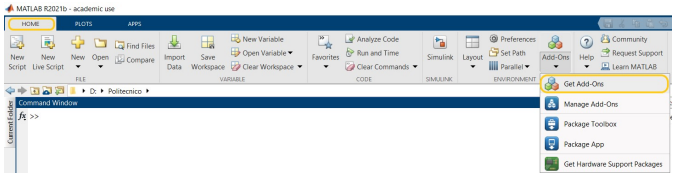

Figure 2.4: MATLAB Add-Ons

The Add-ons that have been included and used are:

- MATLAB Support for MinGW-w64 C/C++ Compiler
- Simulink Support Package per Arduino hardware

At this point, it is possible to pass to the code generation phase in C or C++ language from the Simulink environment. It is necessary to modify some configuration parameters of the model by clicking on the appropriate button, which will open the relative window.

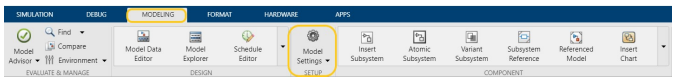

Figure 2.5: Simulink model and setting

Starting with the Solver setting, to make the task similar to a periodic task, that is evaluated at the beginning of each fixed time interval, this must be forced to fixed-step. By doing so, the discrete-time integration interval can be chosen to evaluate the system model at regular time intervals. The accuracy of the results is inversely proportional to this value, which means that the smaller it is, the better the accuracy, the only drawback is the more it is reduced, the longer it will take the system to deliver the results.

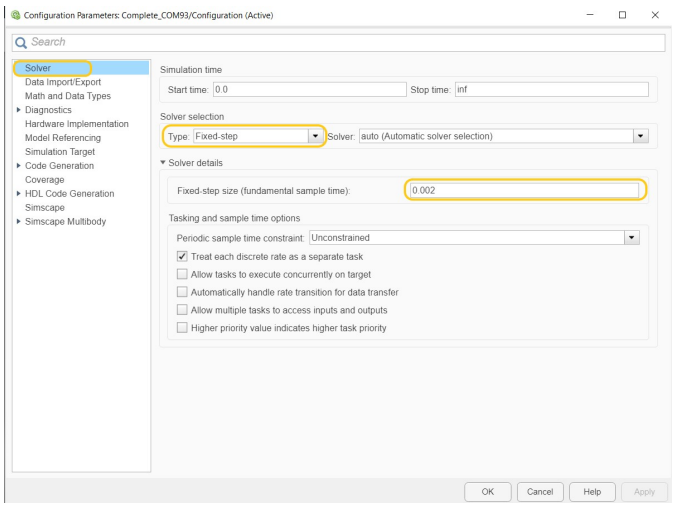

Figure 2.6: Simulink solver panel

The hardware implementation panel allows a wide choice of embedded processors on which the generated code can be uploaded. Once the microcontroller has been established, for this project Arduino Mega2560, Simulink will take care of generating the target file of the system defined as ert.tlc, passing subsequently to the actual generation of the code. Two Arduino boards were used for this project, an important detail is to manually specify the COM port for the shield.

The figure below highlights the significant steps of the procedure just described.

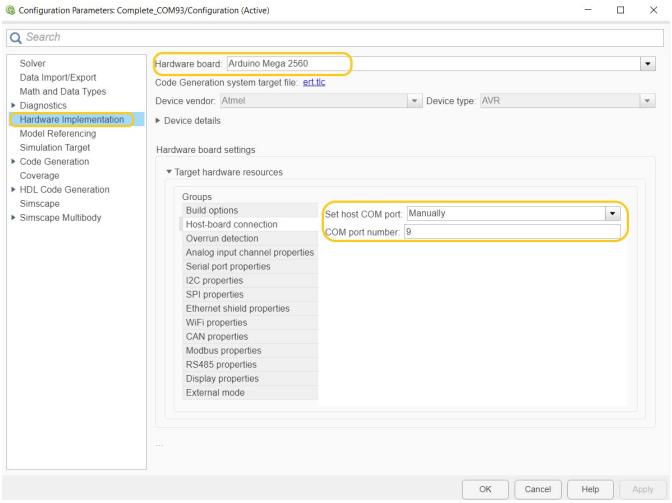

Figure 2.7: Simulink hardware implementation panel

Once all the configuration parameters have been set, a new tab called "Hardware" appears, in which there is a submenu, called" Run on board (External mode) ", which must be selected to enable the code loading function on the Arduino via the USB connection cable. The steps to be performed are shown in the figure below.

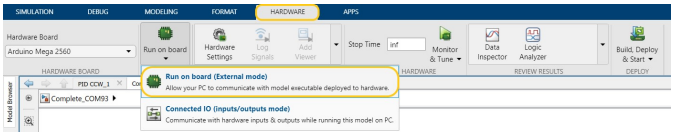

Figure 2.8: Simulink external mode

Finally, the Monitor & Tune option allows getting an infinite time simulation, in which some parameters can be changed in real-time, immediately verifying the system reactions, while the Build Deploy & Start option allowing to load the code on the embedded processor definitive so that it can be used even without a laptop connection.

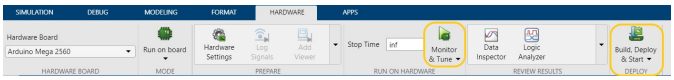

Figure 2.9: Deploy on the target hardware

Chapter 3

SmatGimbal system

3.1 General purpose of control system

A control system manages, commands, directs, or regulates the behavior of other devices, the main purpose is to form a set of control actions, after which the system will behave as desired automatic, respecting the requirements such as speed reaction, overshoot, attenuation, generating a quality control. The control of the systems can range from an approach to a single controller as in the case of domestic heating to multiple solutions as in large systems defined as dynamic, such as the control of a robot in industrial applications. There are two different approaches to a control system: open loop and closed loop. The main difference between an open-loop system and a closed-loop system is that the second one can self-correct while the open-loop system does not. Consequently, closed-loop systems are often called feedback control systems while open-loop systems are also known as non-feedback controls. Open-loop systems tend to be simple and inexpensive as they do not provide feedback to the controller. In other words, open-loop systems are based solely on input and do not use the output feedback for self-correction when running the test. Therefore, the test procedure inserted in an open-loop controller can vary due to external disturbances, such as noise, without the operator noticing.
An example would be that of a system at a constant speed, if there is no direct feedback to monitor this value, the speed of the machine could change during a test for various reasons and the open-loop system does not have a feedback control to inform the controller of these changes [7].

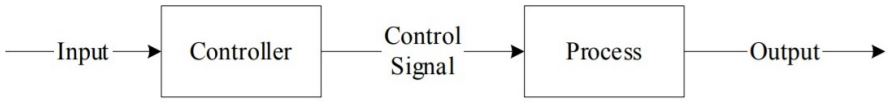

Figure 3.1: Open-loop block diagram

3.1.1 Closed-loop method

In the closed-loop control system, the output signal has a direct impact on the control action, this is due to the presence of a fundamental element: feedback. The error signal produced is the difference between the input and the feedback, it is sent to the controller to be reduced and bring the system output to the desired value. This approach is used in control methods because can indicate the changes in performance necessary to take corrective actions to maintain the activities necessary to arrive on or before the requested date [8].
It consists of some mandatory elements such as the error detector, the controller, the feedback elements, and the installation. The correct positioning of these elements is shown in the block diagram below.

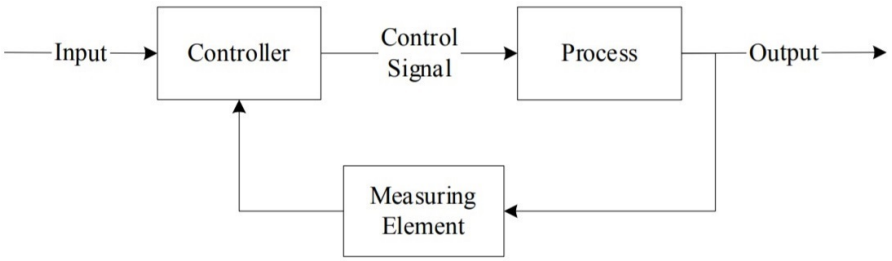

Figure 3.2: Closed-loop block diagram

However, a distinction can be made based on the sign of the feedback, specified in the summation node, it could be positive or negative. In this project, as in the theory of automatic control systems, negative feedback was applied.

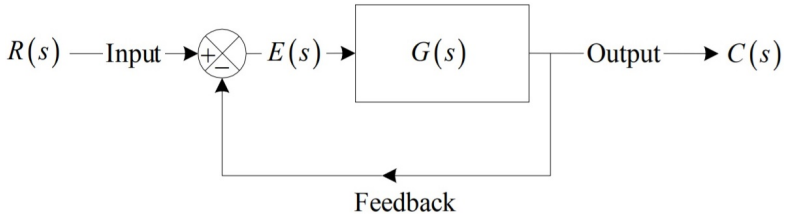

Figure 3.3: Adding node block diagram

- Positive Feedback: A clear example is the circuit of a non-inverting amplifier, where a part of the output voltage is connected to the input of the non-inverting terminal through a feedback circuit using a resistor.
- Negative Feedback: These systems are used to control electronic machines such as power generators, voltage generators, and also to control the speed of machines and it is the approach that has been chosen.

3.2 PID control technique

The PID controller is a very simple and powerful method to control a variety of processes, its acronym stands for "Proportional, Integral, and Derivative". It is composed of a predefined architecture for any type of system and thanks to its simplicity of use, combined with its effectiveness, it is the most used control algorithm in industrial applications.
From a mathematical point of view, the PID regulator is a dynamic system that at each cycle processes an input signal called error, obtained as the difference between the reference and the measured output, providing the control signal at the output. This value is obtained as the sum of the following three values:

- *Proportional term*: it multiplies the constant K_p by the error.
- *Integral term:* it multiplies the constant K_i by the cumulative total error
- *Derivative term:* it multiplies the constant K_d by the rate of change in error

The transcription of these concepts can be represented in a mathematical expression, generally called the PID control law [9].

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{d}{dt} e(t) \tag{3.1}$$

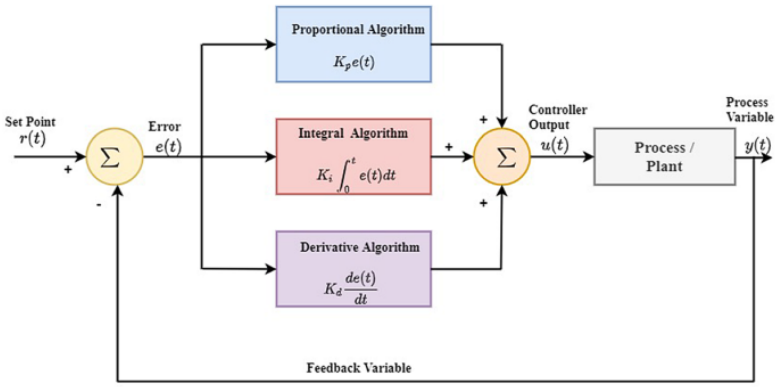

Figure 3.4: PID controller in a feedback loop

The synthesis of PID controllers consists in choosing the most suitable configuration for the application. In most cases, it is done manually through experimental tests on the system. The tuning part is not trivial, the choice of control variables must be weighted to minimize the error variable by the controller. The effects of each action on any closed-loop system are summarized in the table below.

	Rise Time	Settling Time	Overshoot	Steady-state error
P	Decrease	Small Change	Increase	Decrease
I	Decrease	Increase	Increase	Eliminate
D	Small Change	Decrease	Decrease	Small Change

Table 3.1: Effects of PID parameters on the system

Proportional action algebraically links the input e(t) and the output u(t) according to a constant K_p, called proportional gain. In these conditions, the regulator generates an error correction, if the constant increases, the speed of the system will do the same. Once, a certain limit is exceeded, oscillations will be created which will lead to a reduction instability.

The main feature of **Integral action** is eliminating the steady-state error caused by the proportional block, which occurs when dealing with a reference input step. The output of this block is proportional to the time integral of the input error e(t), which means that thanks to the feedback, is returned to the input and mitigates the influence of the reference signal r(t) on the system. The integral of the error increases as long as it is positive and decreases when e(t) takes negative values. For this block, the variable term to be defined is the integration constant K_i. By increasing the constant, the integral will rise faster towards the error, paying for this speed with strong fluctuations that take time to settle, vice versa when it decreases and tends to zero it results in its elimination.

The function of **Derivative action** consists in deriving the signal found at its input; therefore, it takes into account the rapid variations of the error and tries, in some way, to anticipate future corrective action. In practice, since it cannot be used as a single action to insert a constant reference due to the zero it introduces into the origin, its function is realized in a reduction of the signal oscillations around the output value. The properties of the derivative are those of increasing proportionally to the rate of change of the quantity to which it refers, therefore in the case in which the signal r(t) assumes a step shape, the derivative action will act in such a way as to follow the error and correct it quickly. Unfortunately, the derivative term introduces the drawback of amplifying signals with harmonic content at high frequencies. Since generally, this type of signal corresponds to electromagnetic noise superimposed on the useful signal, the use of the derivative term must be evaluated with caution.

In addition to the classic PID configuration, other combinations are also used industrially, each with its peculiarities and specific to the data processes.
The basic configuration is composed of the P action, which is used in systems in which deviations between the steady-state value of the controlled quantity and the desired one are allowed.
The combinations between P and I modules allow for greater accuracy without worsening the degree of stability of the system, combined with a greater response speed. They are used when a modest steady-state error is required together with a good response speed to variations in stress; therefore, they are inserted above all in systems in which load variations occur slowly. Sudden load changes can lead the system to instability when the coefficient of the integral action is not chosen appropriately.
The PD regulators tend to anticipate the error and therefore can be used in systems in which there are sudden variations in load such as in some control systems for servomotors, or even in systems that do not have stability problems. but that only requires a good response speed.
Finally, PID is the controllers are used in slow processes, which do not need immediate responses to the solicitation, where however the presence of oscillations creates problems [10].

3.2.1 Controller the in discrete-time domain

Although studying the controller in the continuous (or analog) domain makes it easier to understand what is happening to the system, to generate the C or C++ code to be implemented on an embedded processor, like Arduino, is necessary to work in a discrete-time environment.
In this way, by defining a time interval, it is possible to sample data regularly. The control law is changed as follows:

$$u[k] = K_p \left(e[k] + \frac{T}{T_i} \sum_{j=0}^{k} e[j] + \frac{Td}{T} \left(e[k] - e[k-1] \right) \right) \tag{3.2}$$

Note that, T is the time step or sampling interval, the proportional term is just the gain multiplied by the discrete error signal, now the integral term becomes a summation, and the differential term is simply the difference between the error current and previous error.

3.3 Closed-loop control for DC Motor

3.3.1 Dynamic model

Establishing the mathematical model that describes the system to be controlled is the basis for the analysis and design of control systems. It starts from the physical equations to obtain mathematical expressions that relate inputs, outputs, and other internal variables of the system. This approach, then, was used to relate the supply voltage of a DC motor to its speed [11].

The first parameter represented is the input voltage V_a, which supplies the series between the electrical equivalent of armature coil L_a and the armature resistance R_a. The second voltage e is opposite to the source because it referred to the back emf (electromotive force), the variable J stands for the inertia moment and finally, f is the friction coefficient.

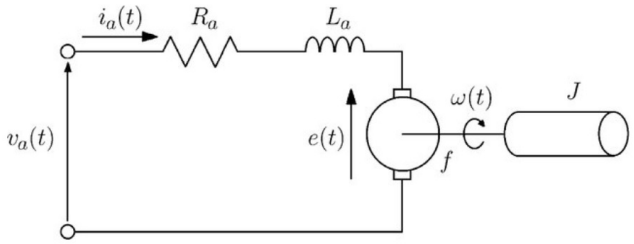

Figure 3.5: Equivalent Circuit of a DC motor armature

If the torque T increases, the armature current also does the same, exactly by a constant K_t. Moreover, the increase in motor emf produces an increase of rotational velocity ω, by a constant K_e.

$$\begin{aligned} T &= K_t i_a \\ e &= K_e \omega \end{aligned} \tag{3.3}$$

By a combination of Newton's law with Kirchhoff's law is possible to obtain the mechanical and electrical behavior of DC motor:

$$\begin{cases} L_a \frac{di_a}{dt} + R_a i_a = e - K_e \omega \\ J \frac{d}{dt} + f\omega = K_t i_a \end{cases} \tag{3.4}$$

In control systems, analysis and design of technologies are applied for development in real-time. The analysis of DC motor can be done easily by Laplace transform.

$$\begin{cases} L_a s I_a(s) + R_a I_a(s) = E(s) - K_e s\omega(s) \\ Js\omega(s) + f\omega(s) = K_t I_a(s) \end{cases} \tag{3.5}$$

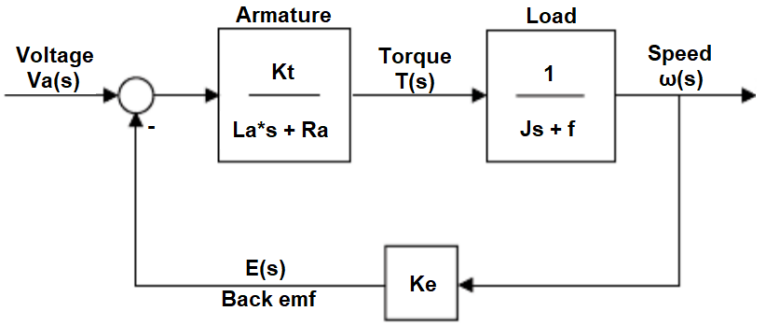

Figure 3.6: Closed-loop model of DC motor

The dynamic model shown in the figure above, relating the supply voltage to the speed of the motor. The simulation parameters could be set on Simulink, and the simulation is run to see the step response in presence of a PID controller[12].

3.3.2 PWM speed control

The Arduino microcontroller is unable to produce true analog output signals, thus the board does not have a driver necessary to create a voltage. To compensate, the PWM technique is used.

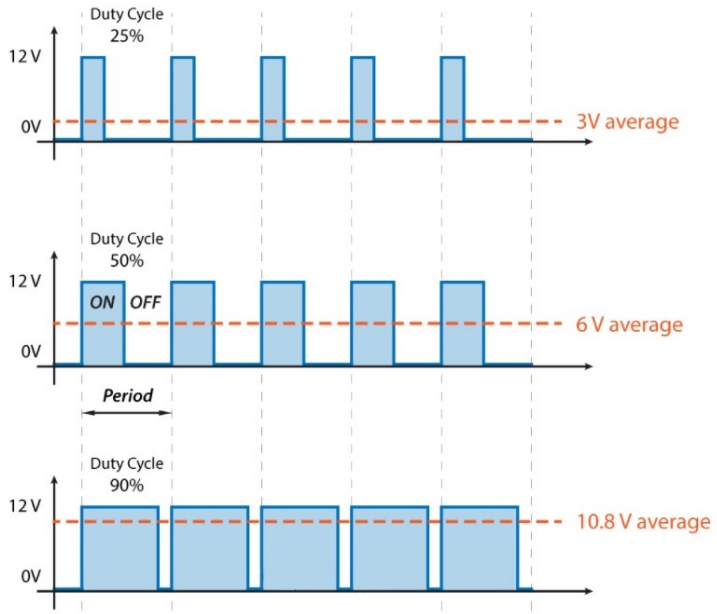

Figure 3.7: Pulse With Modulation

Pulse Width Modulation (PWM), where the amplitude is the duration of the pulse, is a technique of varying the duration of a square wave pulse to control the power supply of any connected device. Using this technique is possible to simulate an analog output from a digital one. Generally, a square wave is obtained with a succession of on (digital high) and off (digital low). So, to replicate the full range of analog values is needed to rely on the time for which a pulse is HIGH and LOW. However, there are two important parameters for a PWM signal: the duty cycle and the frequency.
A duty cycle represents the fraction of a period in an active signal. Specifically, it is the percentage deriving from the ratio between the amplitude of the pulse signal and the total period. Frequency, on the other hand, is defined as the number of oscillations of a wave per unit of time. Once the period is known, the frequency is calculated using the following formula: Frequency = 1/Period. This value refers to the speed at which output can be obtained from the device.
Microcontrollers such as Arduino, not being able to produce an analog voltage, that is a voltage that varies continuously from 0 volts to V_{cc}, are already equipped with PWM outputs, thus facilitating the task. At the pins will be passed a value between 0 and 255 corresponding to a duty cycle varying between 0% and 100% of a square wave [13].

The direction of rotation can be controlled by reversing the motor current flow, the most common method of doing this is by using an H-Bridge.

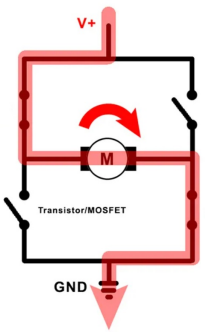

Figure 3.8: H-Bridge structure

This circuit contains four switching elements, transistors or MOSFETs, with the motor in the center, forming a configuration similar to H. By simultaneously operating two particular switches the direction of the current flow changes, with it also the direction of rotation of the motor. Once these two methods, PWM and H-Bridge are combined, is possible to have complete control over the DC motor [14].

3.4 Sensor operation

3.4.1 Hall effect

Hall Effect Sensors are devices that are activated by an external magnetic field, this has two important characteristics: the flux density B and polarity (North and South Poles). The output signal from a Hall effect sensor is the function of magnetic field density around the device. When the magnetic flux density around the sensor exceeds a certain pre-set threshold, the sensor detects it and generates an output voltage called the Hall Voltage, V_H.
This output voltage can be quite small, only a few microvolts even when subjected to strong magnetic fields so most commercially available Hall effect devices are manufactured with built-in DC amplifiers, logic switching circuits, and voltage regulators to improve the sensors sensitivity, hysteresis, and output voltage. This also allows the Hall effect sensor to operate over a wider range of power supplies and magnetic field conditions [15].

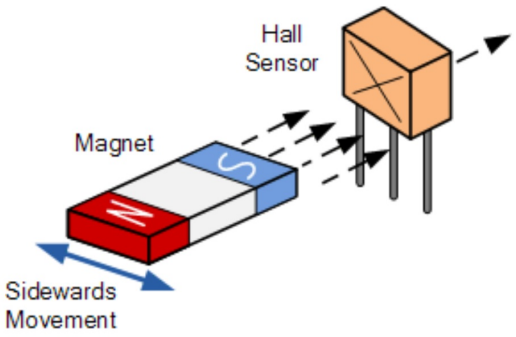

Figure 3.9: Hall effect sensor

The Hall effect sensors, used as limit switches in this project, provide an analog output also called linear. The output signal is then taken directly from the output of the operational amplifier with the output voltage directly proportional to the magnetic field passing through the Hall sensor.
During the calibration phase, the system rotates automatically until the magnet, placed integrally with an axis of movement of the gimbal and strictly at a fixed distance, is detected by the sensor. This automatism is made possible by the reading of the output voltage by the Arduino DAC, subsequently giving the stop command to the motor when this digital value was below a predetermined threshold.

3.4.2 Magnetic encoder

The encoder is a displacement and speed transducer that transforms an angular or linear mechanical movement into a series of digital electrical pulses. These electrical impulses can be used to control the mechanical displacements that generated them and to derive the corresponding speed of displacement.
In magnetic encoders, a signal detection system is used based on the variation of the magnetic flux generated by a magnet placed in rotation in front of two Hall sensors generally fixed on a PCB [16].

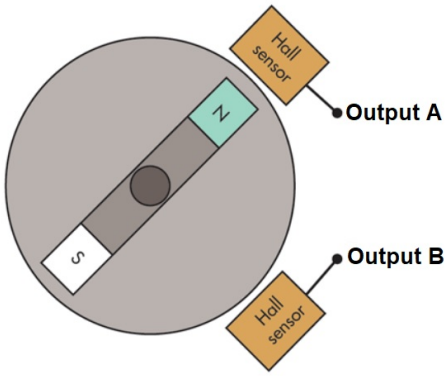

Figure 3.10: Magnetic encoder

By simultaneously tracing the behavior of the two output channels, it is possible to determine the direction of rotation, since they are out of phase with each other by 90 degrees, if the encoder rotates clockwise, output A will precede the output B, otherwise vice versa.

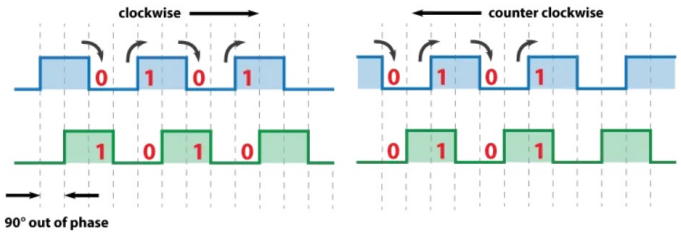

Figure 3.11: Two channel of encoder

The advantage of the magnetic system is mainly the absence of contact in the detection, which prevents wear of the device making it excellent from an economic point of view, as it does not require maintenance. These encoders are thus suitable for applications in harsh environments that require high strength, speed, and thermal resistance while ensuring reliability [16].

3.5 MATLAB/Simulink implementation

By combining the concepts just described it is possible to create an actuation system for the gimbal. The closed-loop guarantees optimal control in combination with the encoder which detects the speed and re-enters it into the system by placing itself in feedback, while the H-bridge takes care of reversing the direction of rotation. Everything is controlled remotely via the joystick which sends signals relating to the speed according to how it is moved.
The block diagram shown below shows how the components interact with each other in a closed loop.

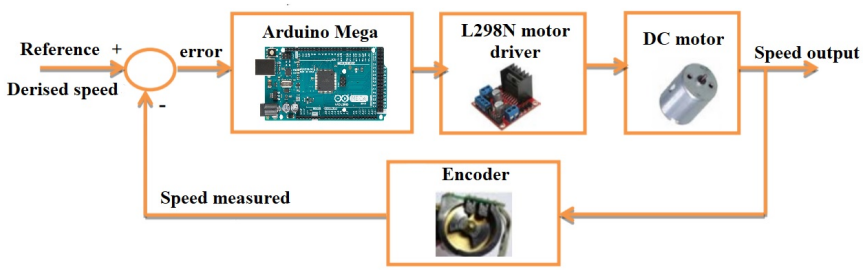

Figure 3.12: General closed-loop of DC motor

The implementation in MATLAB is the transcription of the previous scheme. The encoder function is capable of counting the pulses through the use of Arduino interrupts, this does not allow to lose any impulses. This data is sent to an adjustment block, which through appropriate transformations returns a speed value in RPM and a value in degrees. With the PID controller block, it is possible to choose the discrete-time configuration in advance and thus make it easier to perform the parameter tuning. The output block is the interface between the simulation and the real system, it allows the sending of an 8-bit signal to the DC motor allowing the application of the PWM technique.

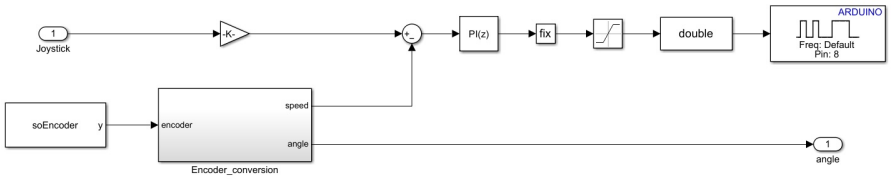

Figure 3.13: Closed-loop Simulink model

3.6 Serial communication protocol

A topic covered in this project, besides the closed-loop control for the actuation system, is the issue of serial communication between two Arduino boards. This concept made it possible to divide the control console module from the rest of the system, avoiding both the passage of long power cables and the noise that could be created on the analog lines.
Serial communication is generally based on a stream of bits transferred over a single cable, in which the bits are transferred one at a time. Often, there is a second communication channel called CLK (clock) which marks the reading time of every single bit. Therefore, the serial communication is based only and always on only two cables. Many types of serial protocols have followed one another to largely meet the needs. Among these, some, such as USB (Universal Serial Bus) and Ethernet, have been very successful, while others are better known in the world of electronics such as SPI and I2C. In general, all serial protocols can be divided into two groups: **synchronous and asynchronous**.

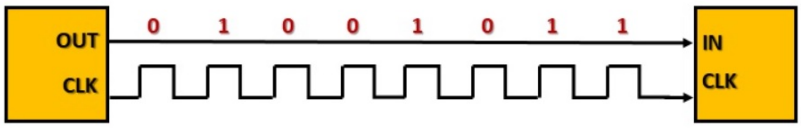

Figure 3.14: Serial communication

Synchronous serial communication is always coupled with a CLOCK signal so that all devices on the serial bus share a common clock. This particular configuration allows a simpler and often faster data transfer, but also requires an extra channel to carry out the communication. I2C and SPI are two synchronous serial communication protocols. Asynchronous serial communication does not use an external CLOCK signal and therefore requires fewer I/O channels for communication. At the expense of this, a reliable system for transmitting and receiving data must therefore be implemented.
The asynchronous serial protocol, like the one used, is based on an intrinsic mechanism of some rules that must be respected to make the data transfer more reliable, the data must therefore respect a certain sequence called framing.

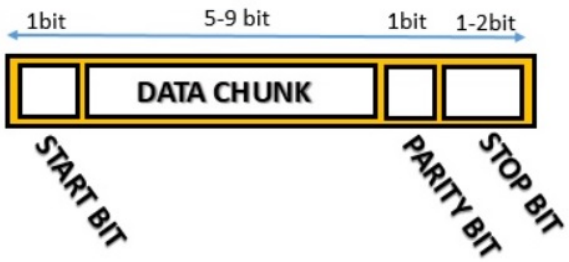

Figure 3.15: Data package

Baud Rate is a rule that specifies how fast data is sent over the serial line. It is generally expressed in bps (bits per second) and its inverse value will indicate how much time a single bit takes in data transmission. This avoids the use of CLOCK since the length and speed of each single data bit are known in advance.

Data to be transmitted is then divided into blocks and incorporated into serial packets where other particular sequences of bits are also added. The number of bits inserted in every single packet can vary, generally between 5 and 9 bits, that is the length of a byte. Therefore, first of all, the two communicating devices must be configured to accept and interpret the same quantity of bits, subsequently, the choice of the most significant bit MSB (most significant bit) must also correspond. That is, they will have to agree whether this will be the first or the last of the bits transmitted.

Synchronization bits are divided into start bits and stop bits and define the beginning and the end of each data packet. The start bit is always indicated when the data line is idle goes from state 1 to state 0, while the stop bits restore the idle state by returning the line to state 1.

Checksum is a very simple method to check for errors during transmission since it has only two possible states: even or odd. It consists of adding all the bits of the message being transmitted and storing the resulting value in the sent frame. To verify the integrity of the message it will be sufficient to perform the same sum operation on reception and compare it with the checksum stored in the frame. If the two values coincide, the data can be considered intact [17]

The used approach is a dual asynchronous serial communication to send the control data on one line and the value of the degrees to be shown on the display on the other. The following figure shows this principle, an important notion is that the two boards must have the ground in common.

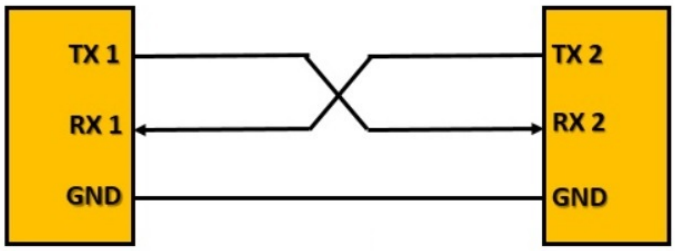

Figure 3.16: Duplex communication

Below is the realization of the principle described above, the serial port used is Serial-3 therefore with pins 14 and 15.

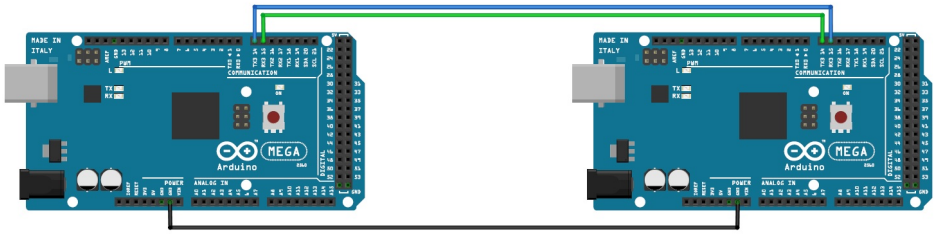

Figure 3.17: Arduino physical connections

Chapter 4

Software development

4.1 General concepts of software architecture

This project was entirely developed in the MATLAB/Simulink environment, using the 2021b release. As mentioned previously, additional packages have been used, the most important in this phase is the Simulink Support Package for Arduino Hardware since it contains the main blocks for interfacing between the hardware and the software. The next figure shows the "common" section of this library, which is the one used.

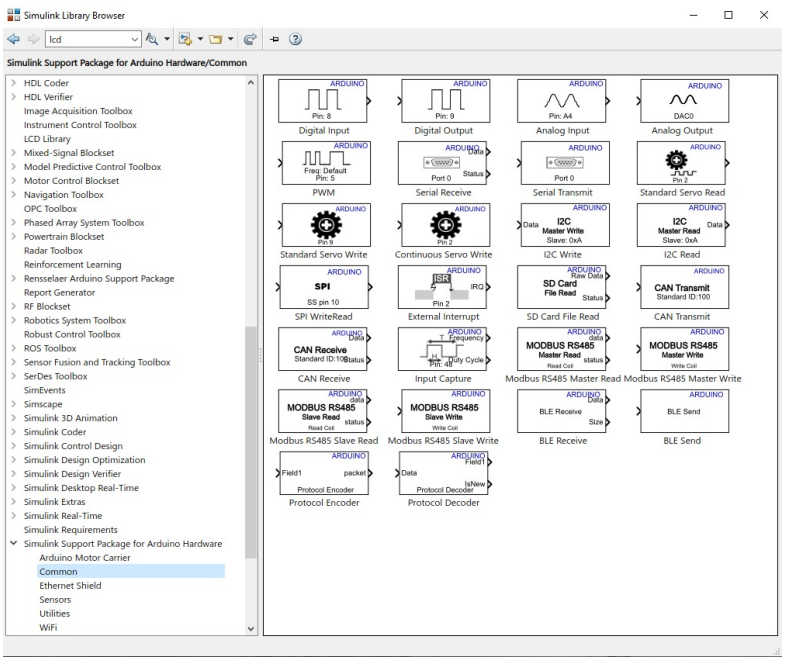

Figure 4.1: Simulink Support Package for Arduino Hardware

Since two Arduino boards were used that communicate with each other via a serial protocol, the software structure was mainly divided into two models. Simulink offers the possibility to bind only one file to each board by specifying, as already said, the COM port manually. The advantage of this type of approach is that it makes the composition of the general project more understandable, in this way the first model is loaded on the Arduino which deals with the management of the movement while the second is loaded on the board located inside the console.

4.2 Motion control model

The following figure shows the Simulink architecture for the motion system, it consists of four fundamental parts:

- Input management on the left side (received in serial)
- Calibration phase in the upper central part
- Motors control in the central part
- Outputs management on the right side

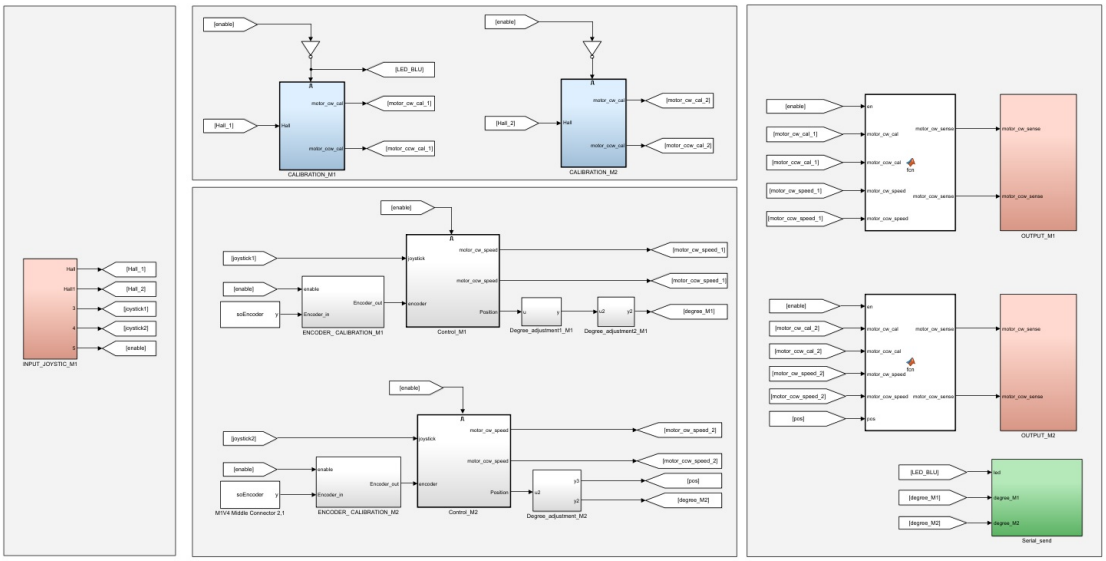

Figure 4.2: Motion control global overview

The system is controlled by a general enabling signal from a button placed on the console, its role is to activate the calibration phase when its status is high and the movement phase via joystick when the status is low.

4.2.1 Input signals

The left side, containing the system inputs, has two Analog Input blocks to read the values from the Hall sensors and a Serial Receiver block for the data coming from the console. The latter is equipped with a status signal that indicates through a Boolean variable when the transmission is active or not, it is used as an enabling of the subsystem to allow data flow.

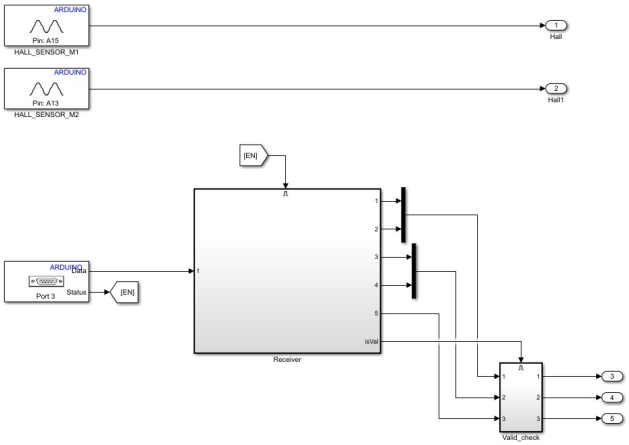

Figure 4.3: Input signals

The Decoder Protocol block is the only thing present inside the subsystem Receiver, it allows the reception of data, deciphering which symbols are used at the beginning and the end of the communication, and verifying the checksum. It also has a status signal called isValid which was used as an enable for further verification. There can be many problems with the serial transmission, these measures require that only the correct data strings be evaluated.

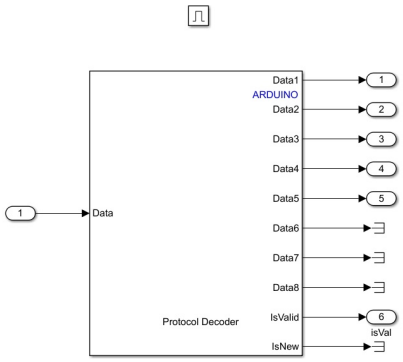

Figure 4.4: Serial signal transmitter

4.2.2 Calibration phase

In the calibration phase, the value of the Hall sensor is read and subsequently filtered through the use of a Median Filter. What is obtained is compared with a threshold, if it is greater the output will be 1 otherwise it will be 0. In the case of a unitary value, it will be the leftmost subsystem that will be enabled, containing a closed-loop system in which the speed and direction, thus causing the system to rotate at a predetermined speed and direction. Movement is guaranteed until the digital value of the sensor voltage obtained from the ADC falls below the present threshold enabling the rightmost subsystem, which contains two zero constants to stop the motor.

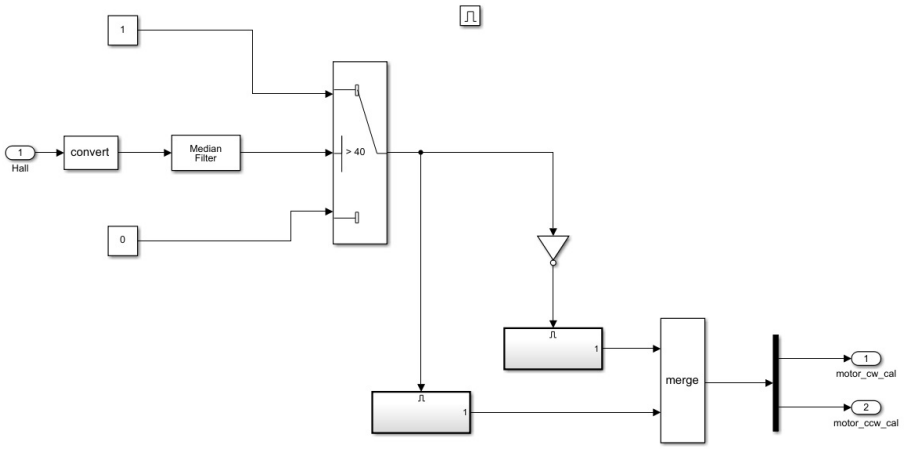

Figure 4.5: Calibration components

The next figure shows the content of the left subsystem, that is the closed control loop equal to that described in the previous chapter.

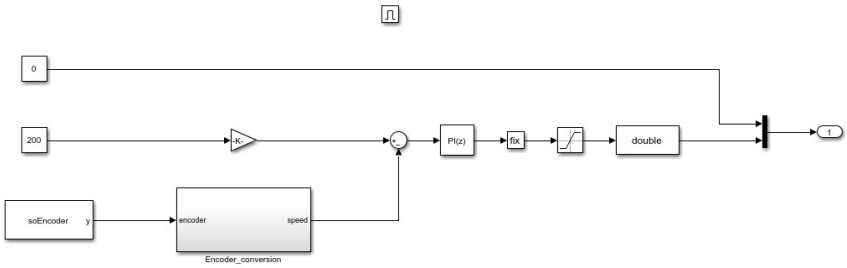

Figure 4.6: Closed-loop in calibration part

4.2.3 Motion control

The set of these blocks, in the central part of the model, includes a series of very important functions to have complete control of the actuation system. A first subsystem resets the value counted by the encoder once the calibration phase ends, a second subsystem manages the two closed loops that a single motor has for direction control, and finally a set of two functions that manipulate the output data and prepare it for sending to the display. The figure below shows what has been done for the first motor, as far as the second one is concerned, the process is identical, the only difference being in the manipulation part of the data sent to the display.

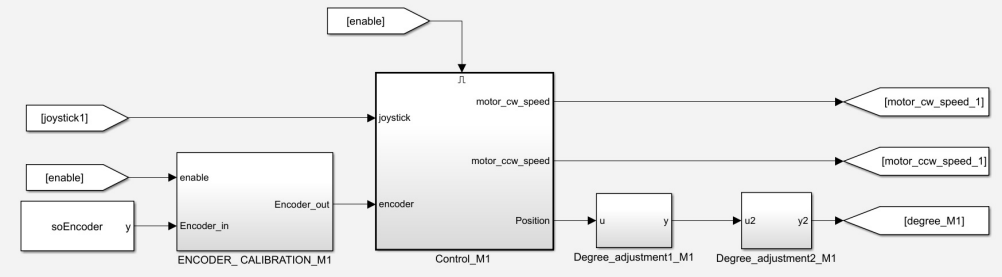

Figure 4.7: Single-motor motion control

Encoder calibration

The "soEncoder" block implements the interface with the encoder sensor, internally calls the Arduino functions for managing interrupts, in this way the microcontroller temporarily interrupts the normal flow of code to execute these instructions. This is a very useful method to ensure that no impulses are lost. This function is enabled only on certain Arduino pins, the first sensor is connected to pins 2 and 3, while the second to pins 18 and 19.
Since this function starts capturing pulses as soon as the program is started, those during the calibration phase are also counted, this value must therefore be reset to zero when a falling edge is detected on the enable signal. The process that is carried out in the “Encoder_Calibration_M1” subsystem consists in subtracting itself from the value that was detected.

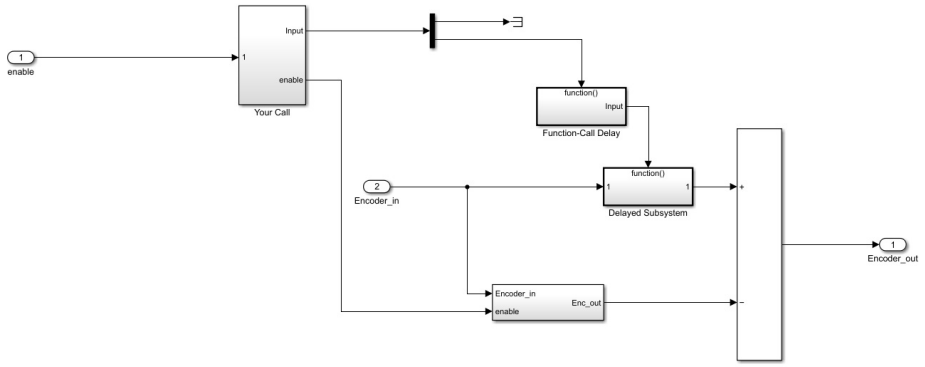

Figure 4.8: Encoder calibration

Two direction control

Inside the Control_M1 block, the two subsystems are containing the two closed loops for motor control, each of which receives in input both a signal from the joystick and the encoder value and output the values for the speed control. The first subsystem also has an output to monitor the grades.
The Joystick input depends on its movement it has been converted into a signal from -255 to 255 with the zero-value corresponding to the rest position of the command, in this way it is possible to know which direction to give to the motor, whether clockwise or counterclockwise. The part with negative numbers is taken as an absolute value by replicating the check carried out on the positive part.

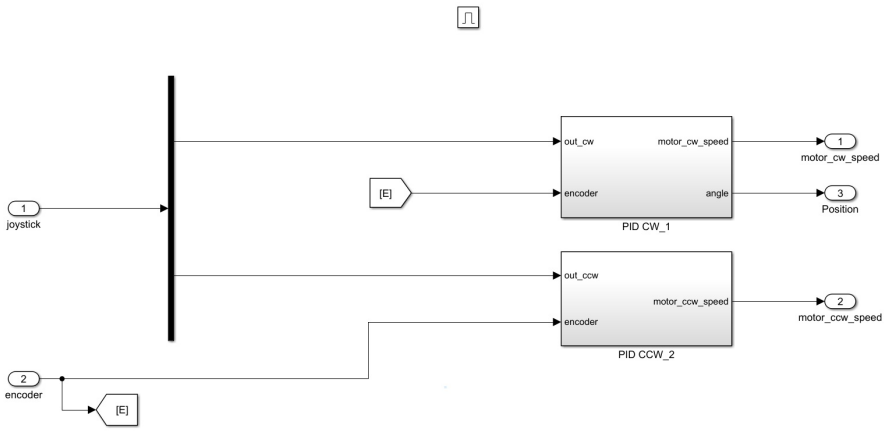

Figure 4.9: Two-direction control

Closed-loop of DC motor

The motor control present in the PID_CW_M1 subsystem and described below is replicated twice, once per direction, for each motor, for a total of four closed loops. The principle on which they are based in the same as described at the end of the previous chapter. The value coming from the joystick is scaled using a constant, as a speed value must be supplied to the input system, while the encoder sensor positioned on the feedback branch reports the actual speed of the system in the summing node. The difference between these two values is processed by the controller, which has the task of regulating the system to bring the error to zero and outputs a command for the motor.

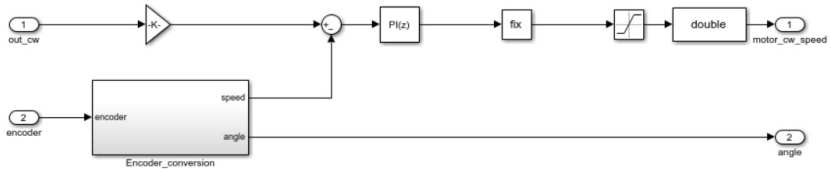

Figure 4.10: Closed-loop of DC motor

The use of the PID block in Simulink facilitates the implementation of the controller's mathematical function and simplifies general system modifications and parameter tuning. It is possible to choose in advance whether to operate in the continuous or discrete-time domain, in addition to the fact that, from a drop-down menu, it is possible to set the type of combination to be applied (PID, PI, PD, only P or only I), having only to act on the numerical parameters.
The figure below shows the screen of the properties of the block just mentioned, through this it is possible to carry out ad-hoc customization for each circuit. In this project, a controller with the following characteristics was chosen:

- Discrete time domain, with the sampling time unspecified but set to -1, by default, this means that the block inherits the sampling time from the model settings.
- Proportional-Integrative type (PI)
- A proportional action with constant K_p equal to 10 and an integrative action with constant K_i equal to 0.002

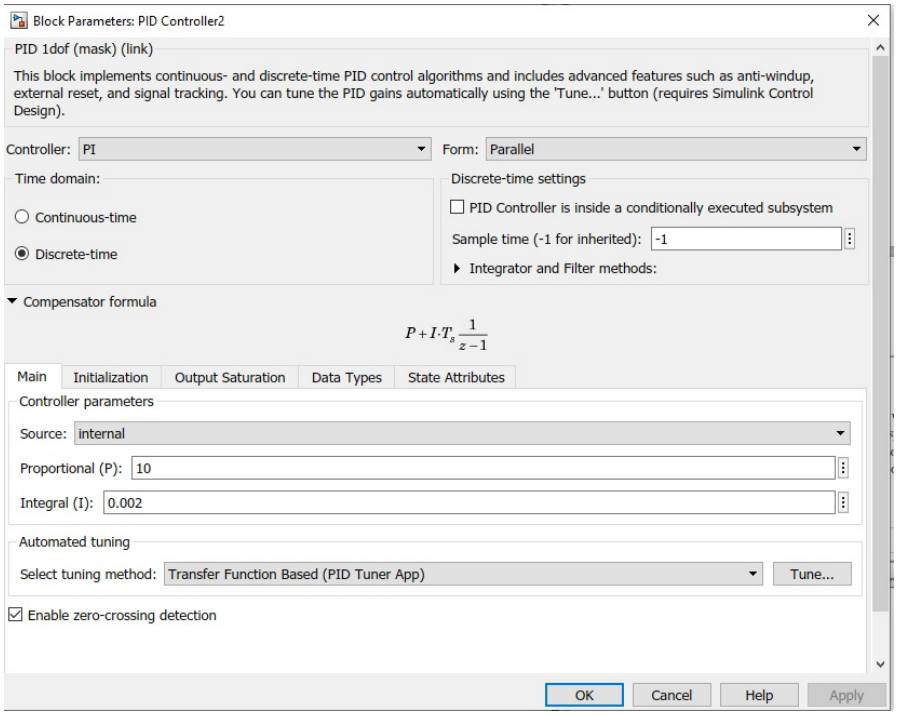

Figure 4.11: PI controller parameters

The following image represents an example of how the output can follow the input when the command is given by the joystick.

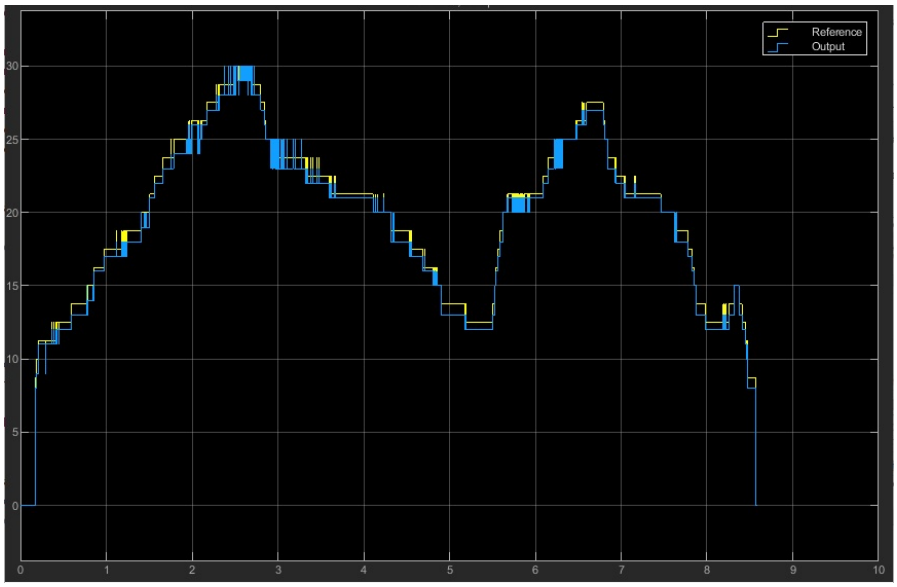

Figure 4.12: Input-output response

Encoder adjustment

A focus should be made on the subsystem that deals with the conversion and adjustment of the encoder data (Figure 4.10) because, to have a speed value in RPM as feedback, it is necessary to manipulate the data starting from the count of pulses coming from the sensor.

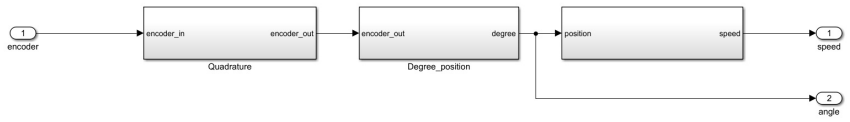

Figure 4.13: Encoder adjustments

The first subsystem contains a gain that divides the count by a factor of four since the encoder is in quadrature. The second subsystem transforms the output data from the first into degrees, multiplying by 360 and dividing by a constant which is given by the product of the number of pulses per second and the motor reduction factor. This is useful for the next step but also for sending data to the display.

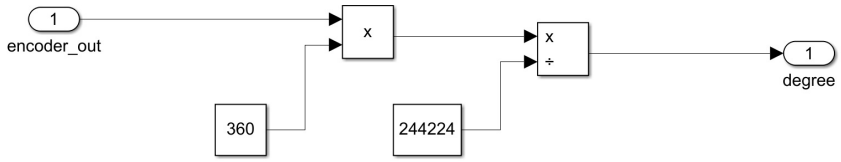

Figure 4.14: Derive Degrees

Finally, the last step is to transform the obtained degrees into a speed value, this is done using the speed measurement block, a component present in the default Simulink library.

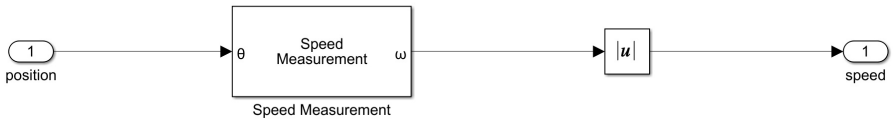

Figure 4.15: Speed conversion

4.2.4 Degree manipulation

Pan angles Adjustment

The last two blocks are shown in figure 4.7 respectively Degree_adjustment1_M1 and Degree_adjustment2_M1 are used, as previously mentioned, for data transmission to the display. The first subsystem contains the addition of a constant, this value considers the position of the system after the calibration phase since at the end of it the system begins to count the degrees from zero, this is due to the arbitrary position of the Hall effect (located based on physical limits). A function is then implemented that allows to resetting the value once a full circle is completed.

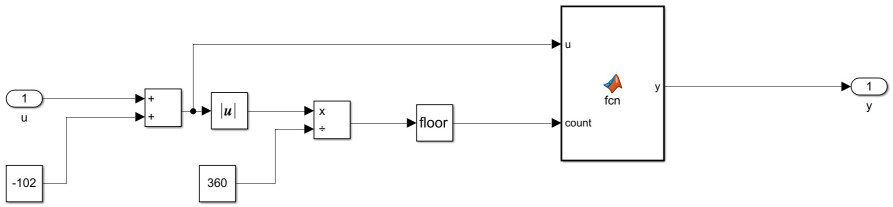

Figure 4.16: Degrees adjustment

The code written inside the Matlab Function,mentioned in the previous figure, is detailed below.

```
function y = fcn(u,count)

if(u>0)
    if (u>360)
        y=u-count*360;
    else
        y=u;
    end

else
    if (u < -360)
        y=u+360*count;
    else
        y=u;
    end
end
```

The second block performs multiplication by a factor of ten, bringing the data, which assumes ±360 as its maximum value, in the order of thousands. This passage is used for the serial transmission, when receiving the data, it will be enough to divide by the same quantity obtaining a floating-point number, thus avoiding the transmission of data of the float type.

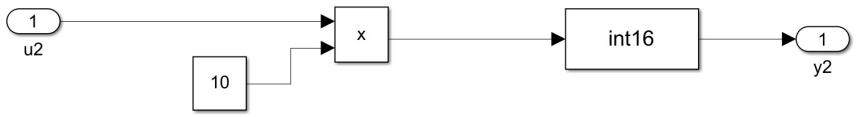

Figure 4.17: Preparing data

Tilt angles adjustment

For the second motor, there is no need to reset the degree count, as the system must be limited between ±110 degrees.

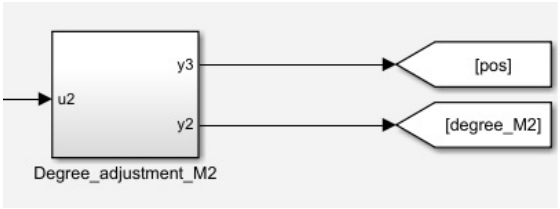

Figure 4.18: Degree adjustment for the second motor

Therefore, it only includes the constant to be added due to the positioning of the Hall sensor and the method of data transmission via serial. The difference between the previous one consists of two outputs: one for displaying and one for monitoring the position to limit it.

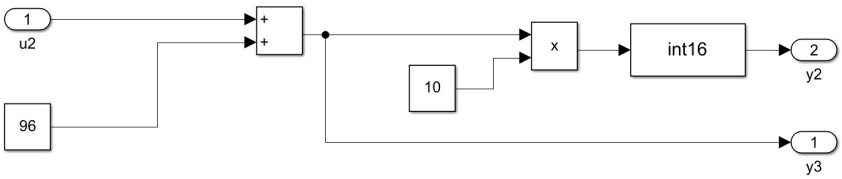

Figure 4.19: Adjustment and preparing data

4.2.5 Output command

First motor

Finally, on the right side of the Simulink model, all system outputs are implemented. The command of the motors, concerning both the calibration phase and the gear movements, consists of a Matlab function that receives these as inputs and provides the correct output based on the state of the enable signal.

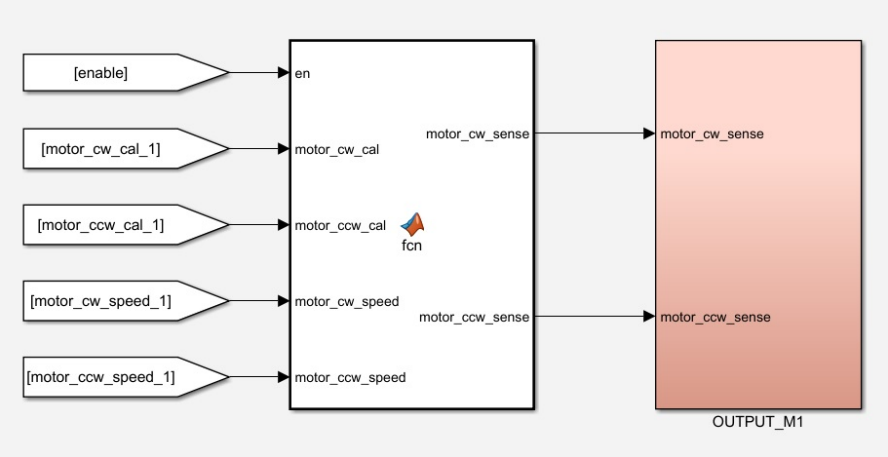

Figure 4.20: DC motor output menage

The code written inside the Matlab Function, shown in the previous figure, is described below.

```
function [motor_cw_sense, motor_ccw_sense] = fcn(en,motor_cw_cal,
    motor_ccw_cal,motor_cw_speed,motor_ccw_speed)


    if (en==0)
        motor_cw_sense=motor_cw_cal;
        motor_ccw_sense=motor_ccw_cal;

    else
        motor_cw_sense=motor_cw_speed;
        motor_ccw_sense=motor_ccw_speed;

    end
```

This solution has been applied to have a single signal be sent in input to the PWM blocks. Note that only some Arduino pins allow sending this type of signal, the following figure shows it.

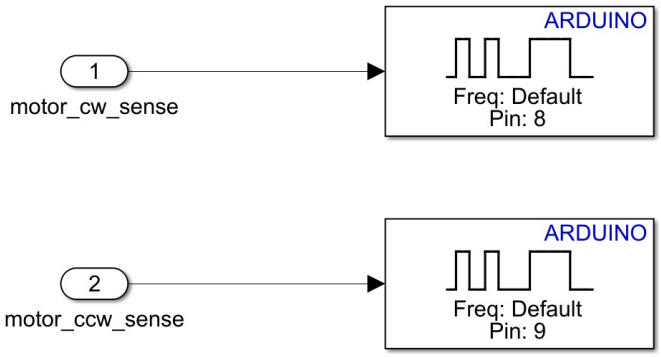

Figure 4.21: Output blocks

Second motor

For the second motor, the configuration is the same, the only difference is that there is an additional input that brings the numerical value in degrees to allow the function to stop the motors after a certain threshold, exactly -110 and +96 degrees.

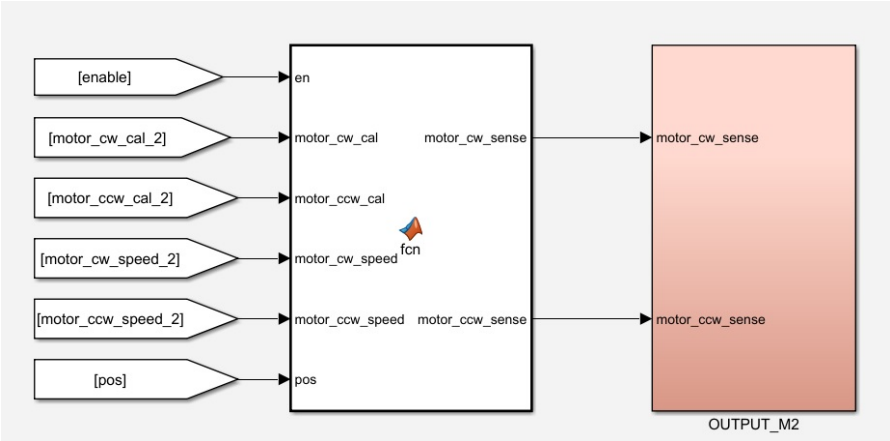

Figure 4.22: Second motor output menage

This function is slightly more complex than the previous one as it manages the possibility of stopping the motor running after 96 degrees making sure that afterward it can only be moved clockwise, vice versa, once the -110 degrees have been exceeded, the system can only rotate in a counterclockwise direction.
The outputs of this function are then connected to two Arduino PWM blocks, as shown in figure 4.21, specifically to pins 10 and 11.

```
function [motor_cw_sense, motor_ccw_sense] = fcn(en,motor_cw_cal,
    motor_ccw_cal,motor_cw_speed,motor_ccw_speed,pos)


if (en==0)
    motor_cw_sense=motor_cw_cal;
    motor_ccw_sense=motor_ccw_cal;

else

              if( pos > 0 )
                   if (pos < 96 )
                        motor_cw_sense=motor_cw_speed;
                        motor_ccw_sense=motor_ccw_speed;
                   else
                        motor_cw_sense=motor_cw_speed;
                        motor_ccw_sense=0;
                   end

              else
                   if (pos > -110 )
                        motor_cw_sense=motor_cw_speed;
                        motor_ccw_sense=motor_ccw_speed;
                   else
                        motor_cw_sense=0;
                        motor_ccw_sense=motor_ccw_speed;
                   end
              end

end
```

4.2.6 Serial transmission to Display

The last part present in this model is the one dedicated to sending data via the serial protocol. In this phase, three data are sent, two numerical values relating to the angles of PAN and TILT and one of the Boolean types. The latter is the enable signal that will allow the lighting of a LED located in the command console.

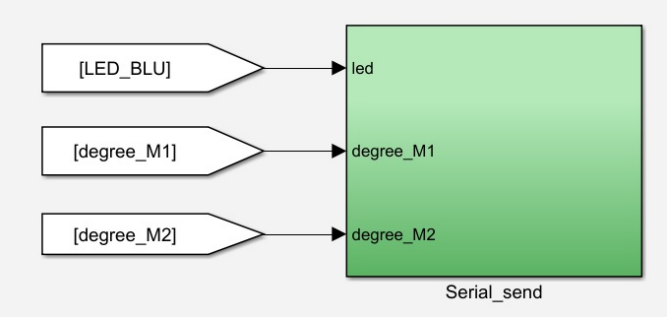

Figure 4.23: Transmitter section

The subsystem contains the Simulink block called Protocol Encoder, which takes care of data packaging and sending. In this phase, it is possible to choose both the start and end communication characters and the presence or absence of a checksum. These values must agree with those present in the Protocol Decoder, already mentioned in sub-paragraph 4.2.1.

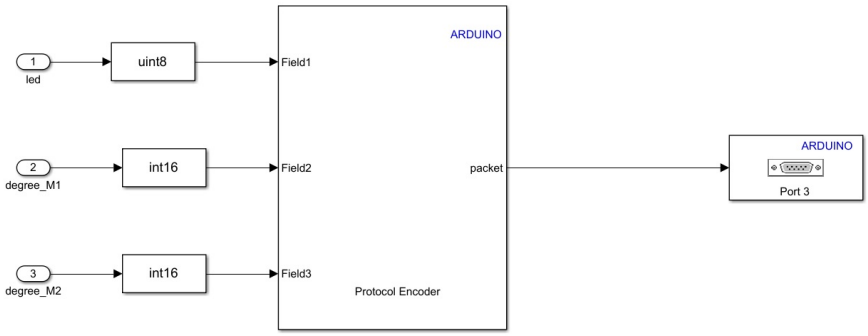

Figure 4.24: Serial communication interface with Arduino

It is important to note that, both in the serial receiver block and in the serial transmitter block, the Arduino pins must not be specified but only the serial port, in this case the third, which corresponds to pins 14 and 15. The wiring must be done in based on which device is in charge of transmitting or receiving.

4.3 Remote control model

The second file that is part of the architecture of this project is deployed on the second Arduino, the one located inside the command console. The next figure highlights the two parts division:

- Input commands on the left (sent in serial)
- Outputs management on the right (received in serial)

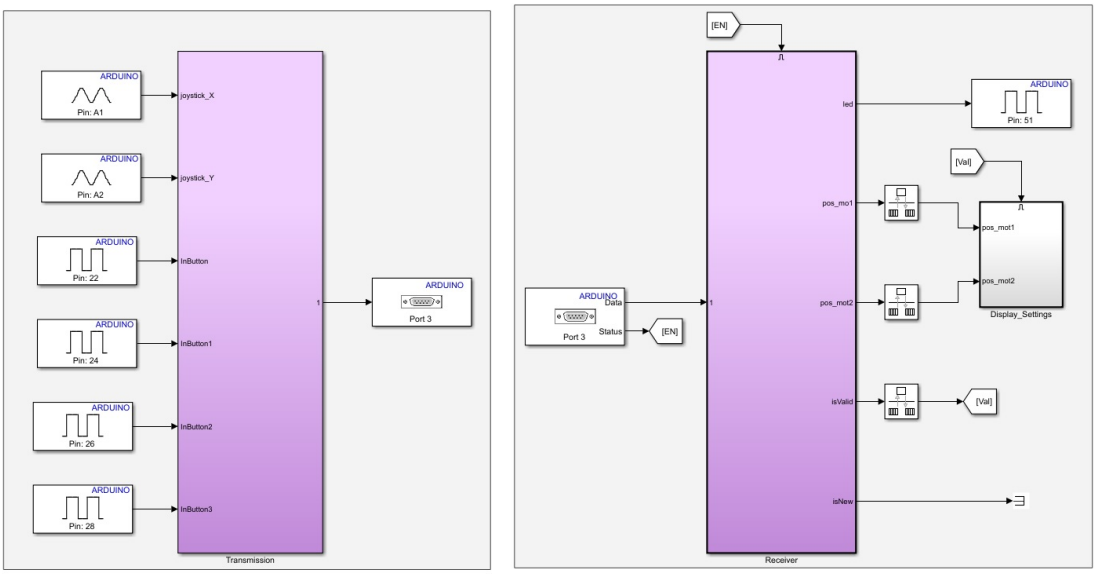

Figure 4.25: Remote control global overview

The two analog inputs of the Joystick, which represent the movement on the X and Y axes, converge on the left side. To these are also added the four buttons, which have been interfaced through the use of the support package that allows adding the Pull-up resistor to the Arduino digital pins.

On the other hand, again through serial communication, the data arrives at the console and is enabled by the transmission status as already mentioned before. The data in question are the system degree values and the control variable for the calibration LED.

4.3.1 Joystick and buttons input signals

Once the correct transformations have been made on the signals of the joystick and buttons, they all converge in a protocol Encoder, this allows serial transmission and the use of them in the main process.

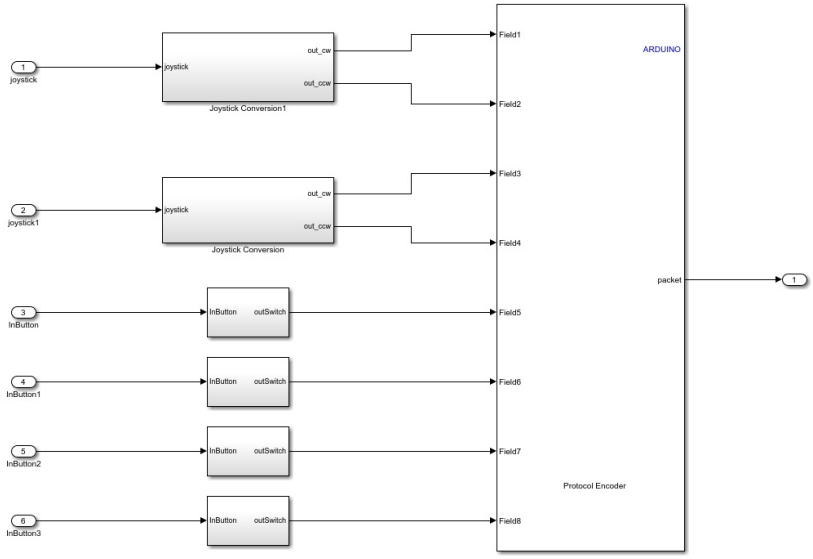

Figure 4.26: Joystick and buttons blocks

Joystick details

Since the joystick is consists of two potentiometers, the input signal is the voltage value detected by the Arduino ADC, from this digital signal it is possible, by making adjustments, obtaining an output between -255 and 255. The process below describes in detail the development made on one axis, as far as the other axis is concerned, the procedure is the same.

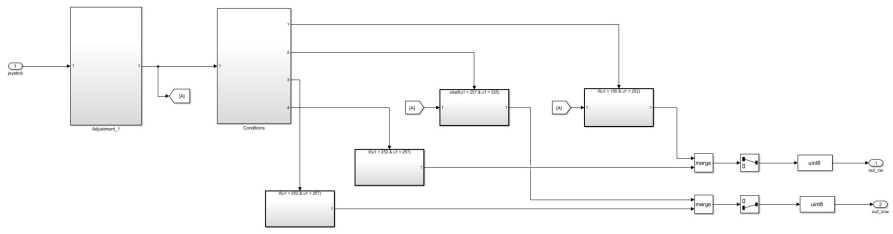

Figure 4.27: Joystick signals interpretation

The first step is to divide the output by a coefficient of two, this occurs in the first subsystem. As shown in the next figure, all conditions are included that translate into code form if the joystick handle is moved in one direction or the other, their outputs are connected to four different subsystems.

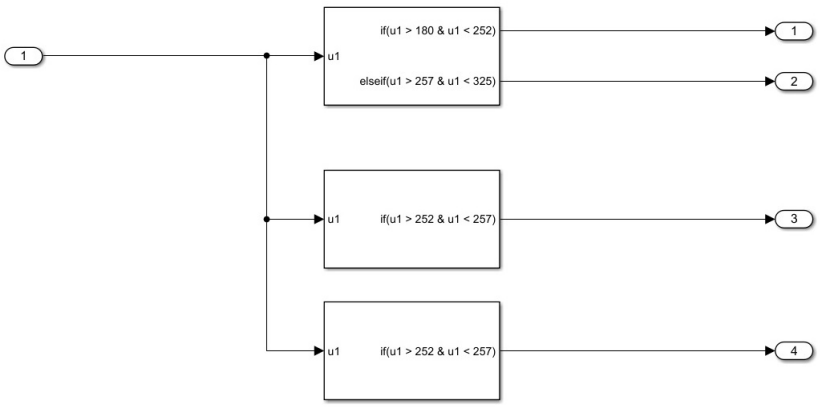

Figure 4.28: Joystick conditions

The two at the bottom left both contain two null constants, while the two on the right show the transformations used to obtain the desired output. The signal is not rescaled between -255 and 255 but is interpreted as 255-0 in one direction and 0-255 in the opposite. For the first condition, the signal is subtracted from a constant of 255 to obtain the maximum value when the lever is tilted to the left, for example, and a null value when it is in the center. In the rightmost subsystem, the exact opposite happens, a constant is subtracted from the signal to have the opposite behavior. A multiplication factor of four is added to both, this is because the potentiometers do not use their entire range of values.

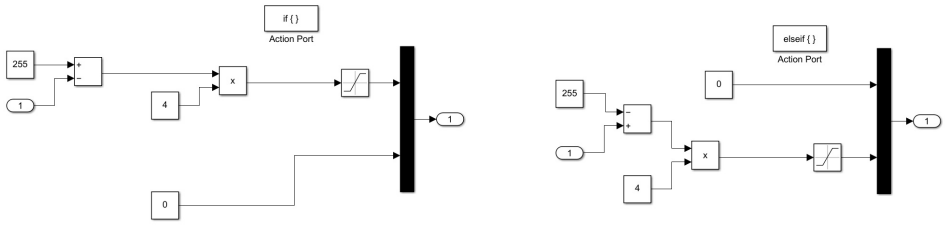

Figure 4.29: Joystick data adjustment

Button details

The buttons used cannot maintain their state, they need a logic that makes them behave like switches. It starts from the output that produces the Digital output block of a single button, the digital signal can only have two states, High or Low. However, being connected between ground and a digital pin, the logic of functioning is inverse, they produce zero if they are pressed.
For simplicity, a "NOT" block is positioned at the beginning to bring everything back into the classic logic. The operation is quite simple when the signal has a rising edge the rightmost subsystem forces the output to 1, and since the signal returns to zero when the button returns to the rest position, a subsystem takes the output state and through a memory block it brings the actual output to the high state. The last subsystem at the bottom compares the output with the input signal, if they are all two 1s for an instant it means that the button has been pressed again and the output must reset, this simulates the behavior of a switch.

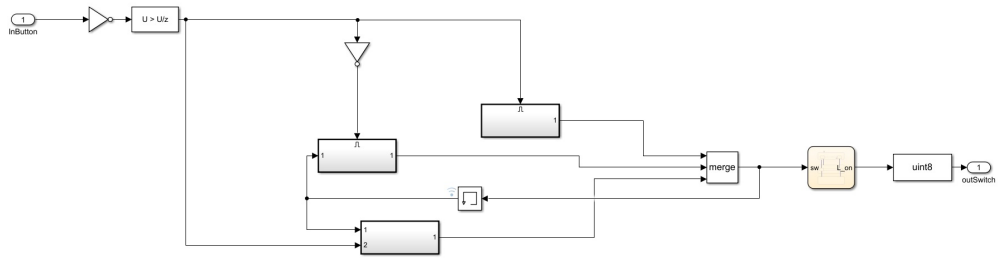

Figure 4.30: Single button model

Buttons often generate spurious open/close transitions when pressed, due to mechanical and physical problems: these transitions can be read as multiple presses in a very short time, detected by the microcontroller, which can test the state of a button up to a few million times per second. The debounce means checking the status in a short period to make sure that the button is pressed definitively. Without debouncing, pressing the button once can cause unpredictable results.
The simplest hardware solution, which works most of the time, is the hardware one. It is enough to implement a resistor and a capacitor. It is a space-saving solution that uses low-cost components [18].

A second solution, the one adopted in this project is software development, the Stateflow below shows how this is done. The Debouncer chart contains an intermediate state called Debounce. This state isolates transient inputs by checking if the signal SW remains positive or negative, or if it fluctuates between zero crossings over a prescribed period.
If SW has been positive for longer than 0.07 seconds, the switch moves to state On, but if SW has been negative for longer than 0.07 seconds, the switch moves to state Off. Moreover, when SW fluctuates between zero crossings for longer than 0.4 seconds, the switch moves to the state off. Fault, isolating SW as a transient signal and giving it time to recover [19]

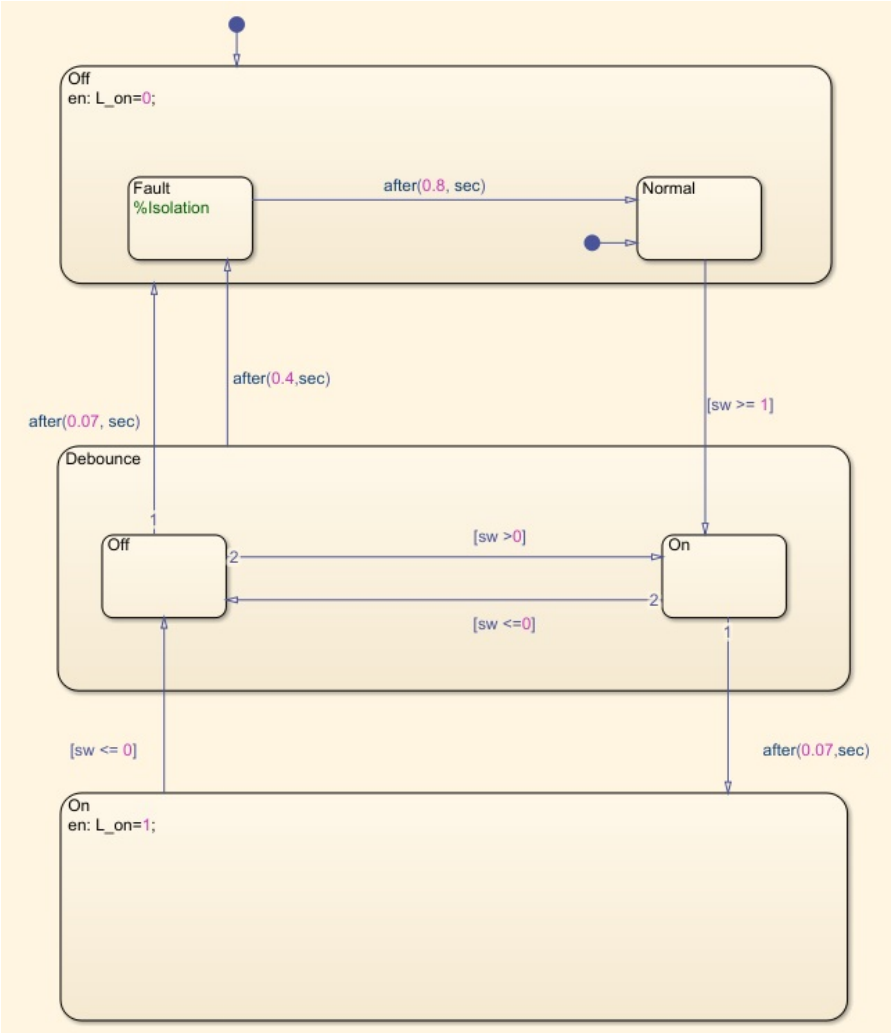

Figure 4.31: Stateflow of debounce

4.3.2 MATLAB-Arduino Display interface

The incoming data enters the Protocol Encoder block, which, as already mentioned, decrypts the packet based on the start and end communication characters. The isValid signal is again used as an additional check.

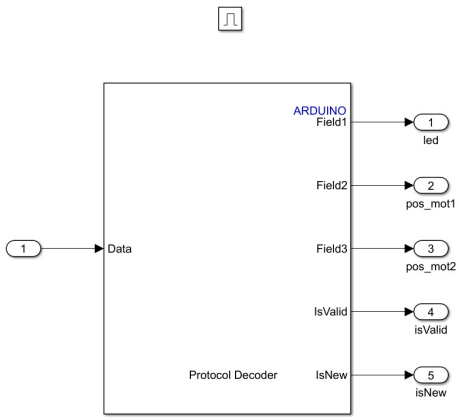

Figure 4.32: Serial signal receiver

The first output, referred to as the LED, has been separated as it is simply connected to an Arduino digital pin, the other two are those that represent the angles of PAN and TILT. They are first divided each by a factor of ten, following the logic mentioned before, thus obtaining a floating-point number to be passed to the interface function with the display.

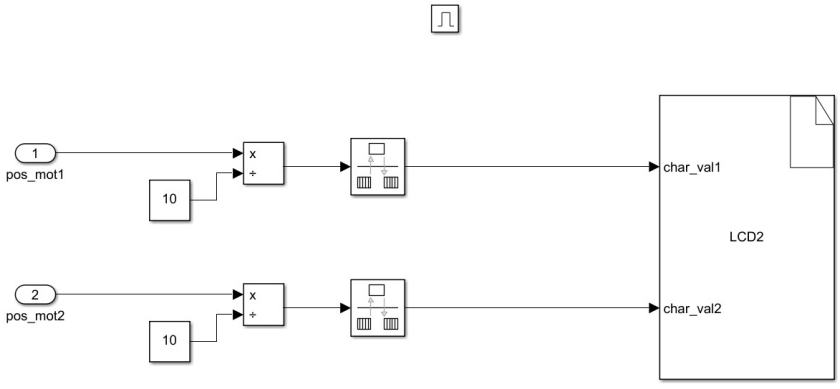

Figure 4.33: Display interface

The interface between Arduino and Simulink, as regards the display, takes place through the use of an S-Function Builder that allows writing the C code by creating an environment similar to that of the Arduino IDE. Here there are two default functions which are called "void setup" and "void loop", the first contains the definition of the variables, the second contains the program that will be repeated in a loop.
Initially, the useful libraries are defined, inside a MATLAB_MEX_FILE, subsequently, in the UPDATE section the starting points of the display cursors are defined. The numeric input data is transformed into strings in the OUTPUT section and finally printed. Thanks to this technique it is possible to update the whole string avoiding the flickering phenomenon.

```
/* Includes_BEGIN */
#include <math.h>
#include <string.h>
#include <stdlib.h>
#ifndef MATLAB_MEX_FILE
#include "LiquidCrystal.h"
#include "LiquidCrystal.cpp"

LiquidCrystal lcd(25,27, 33,35,37,39);

#endif
/* Includes_END */

void LCD2_Start_wrapper(real_T *xD,
                        SimStruct *S)
{
/* Start_BEGIN */
/* Custom Start code goes here. */
/* Start_END */
}

void LCD2_Outputs_wrapper(const real_T *char_val1,
                          const real_T *char_val2,
                          const real_T *xD,
                          SimStruct *S)
{
/* Output_BEGIN */

 if (xD[0] ==1)
 {
     #ifndef MATLAB_MEX_FILE
     char str1[10];
     lcd.setCursor(5, 0);
     dtostrf(char_val1[0],6,1,str1);
```

```
35        lcd.print(str1);
36
37        char str2[10];
38        lcd.setCursor(5, 1);
39        dtostrf(char_val2[0],6,1,str2);
40        lcd.print(str2);
41
42        delay(400);
43       #endif
44    }
45  /* Output_END */
46  }
47
48  void LCD2_Update_wrapper(const real_T *char_val1,
49                           const real_T *char_val2,
50                           real_T *xD,
51                           SimStruct *S)
52  {
53  /* Update_BEGIN */
54  if(xD[0] != 1)
55      {
56          #ifndef MATLAB_MEX_FILE
57          lcd.begin(16,2);
58           lcd.setCursor(0, 0);
59           lcd.print("M1: ");
60
61           lcd.setCursor(0, 1);
62           lcd.print("M2: ");
63          #endif
64          xD[0] =1;
65      }
66  /* Update_END */
67  }
68
69  void LCD2_Terminate_wrapper(real_T *xD,
70                              SimStruct *S)
71  {
72  /* Terminate_BEGIN */
73  /* Custom Terminate code goes here. */
74  /* Terminate_END */
75  }
```

Chapter 5

Test bench environment

5.1 Global hardware configuration

Each component used for the realization of this project has been individually tested on a test bench to study both its functioning and its interface with the MATLAB software. Once the compatibility with the other components was also verified, it was possible to wire up the system and perform more complex tests.
The figure below shows the complete wiring diagram with all components and related wiring.

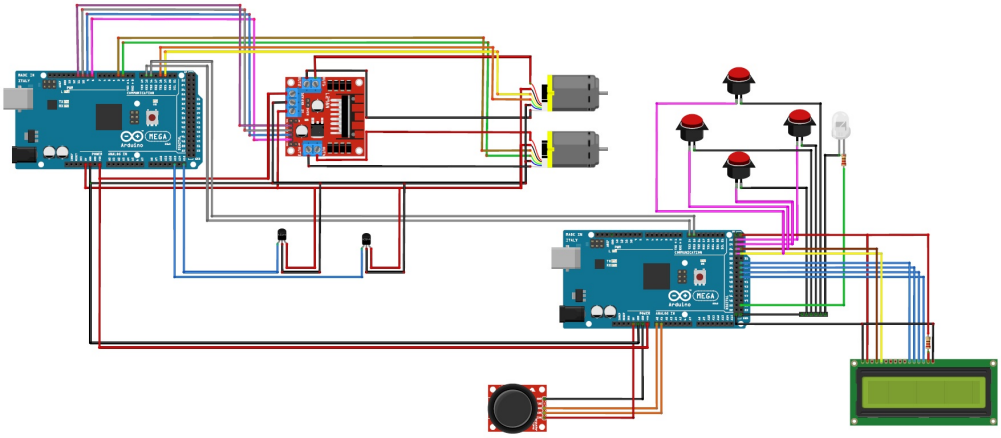

Figure 5.1: Complete connections scheme

Being composed of two parts connected, for a factor of clarity, the diagrams related to the implementation of the actuation system and the console are shown separately in the following sections.

A wooden structure was created as a support for the gimbal shell to study its behavior in a static environment, also making sure it is in the same position as when it will be positioned under the wing of the airplane.

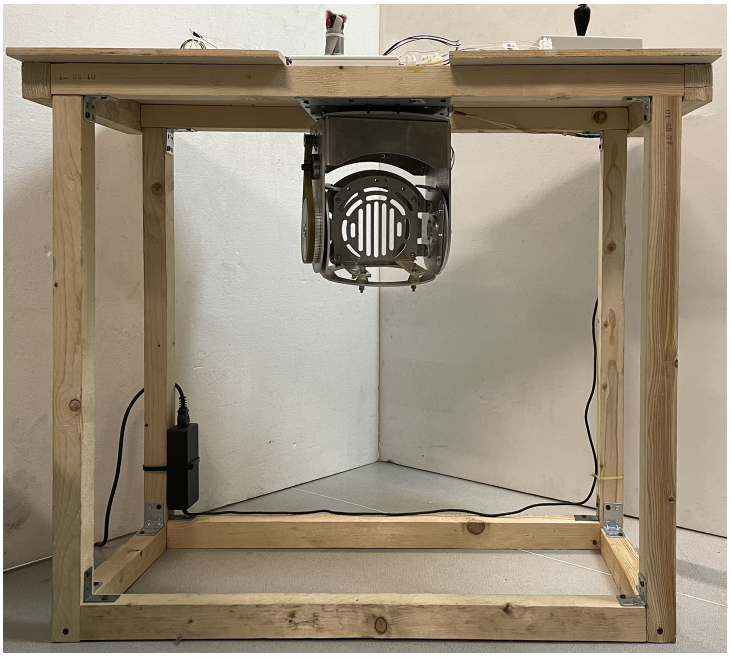

Figure 5.2: Side view of test bench

The top view shows the console on the left side and the various connections made on the right side. The concept is to separate the test circuit and the console just as it is.

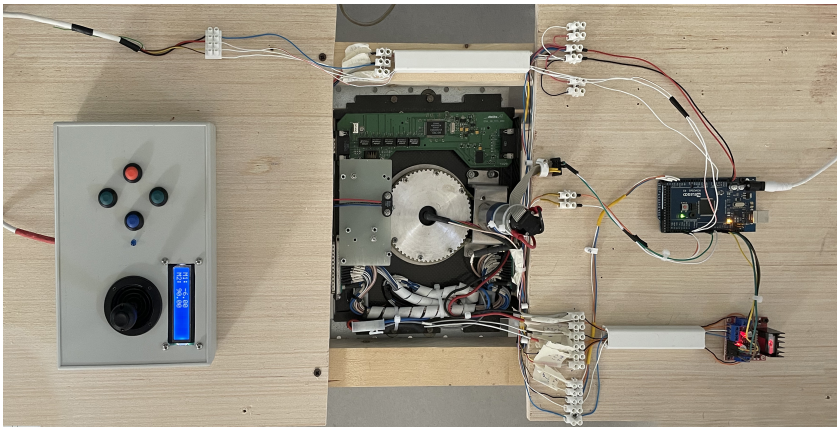

Figure 5.3: Top view of test bench

5.2 Gimbal hardware configuration

The figure below is an overview of the test bench regarding the interim connections of the actuation system.

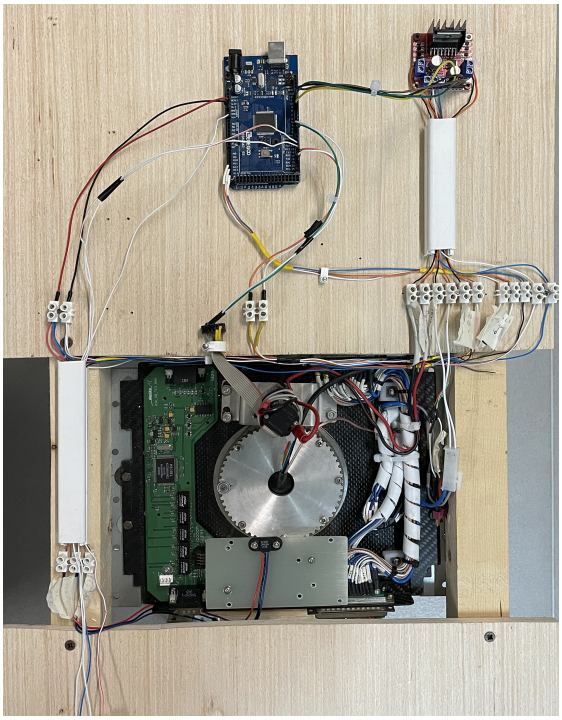

Figure 5.4: Detailed motors connections

The circuit below is a representation of the gimbal motion control connections, this drawing allows to detail clarifying the wiring system.

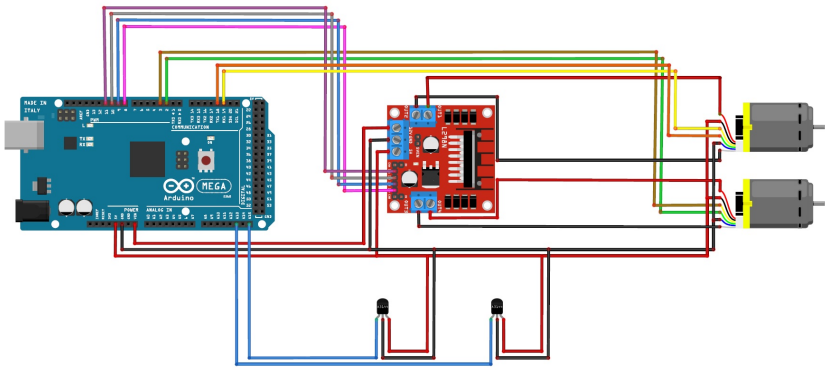

Figure 5.5: Motors connections scheme

5.3 Console hardware configuration

The console had been assembled previously but did not have the display, this component was added in this project together with the strengthening of the welds and the fixing of any moving parts concerning internal connections, in this way the internal components should be resistant to vibrations.

Figure 5.6: Remote console

Since cable management was done inside the console, the diagram below clearly shows all the connections between the components.

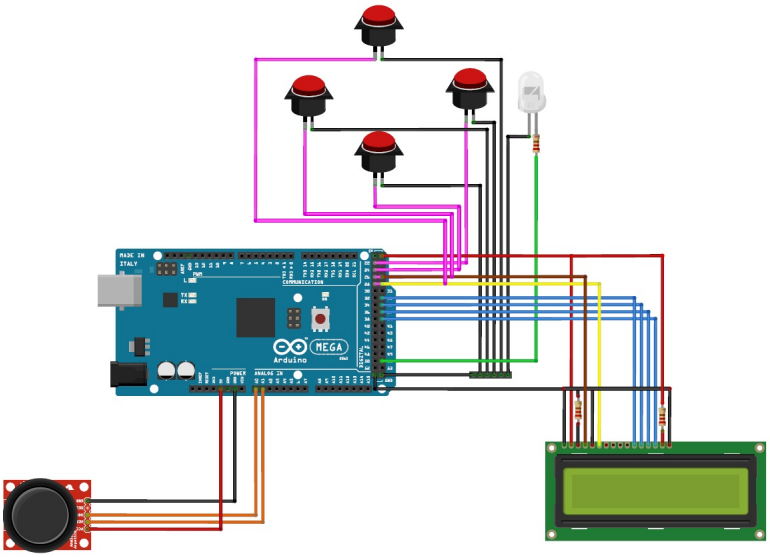

Figure 5.7: Console connections scheme

Chapter 6

Development of a PCB

6.1 Preliminary concepts

To complete the work, it was decided to create a printed circuit, passing from a test bench solution to a complete solution that would become plug and play and could be allocated within the SmartBay platform to also carry out in-flight tests. The request was to design a board in which the Arduino and the motor driver could be plugged, and which was able to contain adequate connectors to ensure that the connections between the components resist the vibrations of the aircraft.

For this purpose, it was decided to operate Fusion 360, a program of the Autodesk suite. This is a versatile CAD package with all the needed features to develop products, from concept to design verification to manufacturing with both traditional and digital tools, such as 3D printing [20].
Autodesk also included Eagle, the graphic editor for schematic and circuit board design, in Fusion 360. This integration made it possible to combine the functionalities of mechanical CAD and electronics. Eagle is designed software for creating electrical diagrams and PCBs. Inside, it includes some easy-to-use features, including schematic editor, layout editor, auto-router, accessible library content. In particular, the schematic editor turns ideas into reality with the schematic capture functionality, while the layout editor brings projects to life with intuitive tools for creating the printed circuit board (PCB) layout. The software has an intuitive interface and, in addition, also includes the complete suite of SPICE simulation methods, which allows validating the performance of the circuits created [21].

6.2 Project sizing

The calculations for creating the working area, that is the actual PCB size, were made based on the top view of the gimbal system. With this approach it was possible to choose the actual shape it had to have, positioning the fixing points correctly so as not to clutter the parts already present and also making sure to have the right space for the allocation of the components.
As a further test, a model was also made with the 3D printer before sending the prototype of the board to be built.

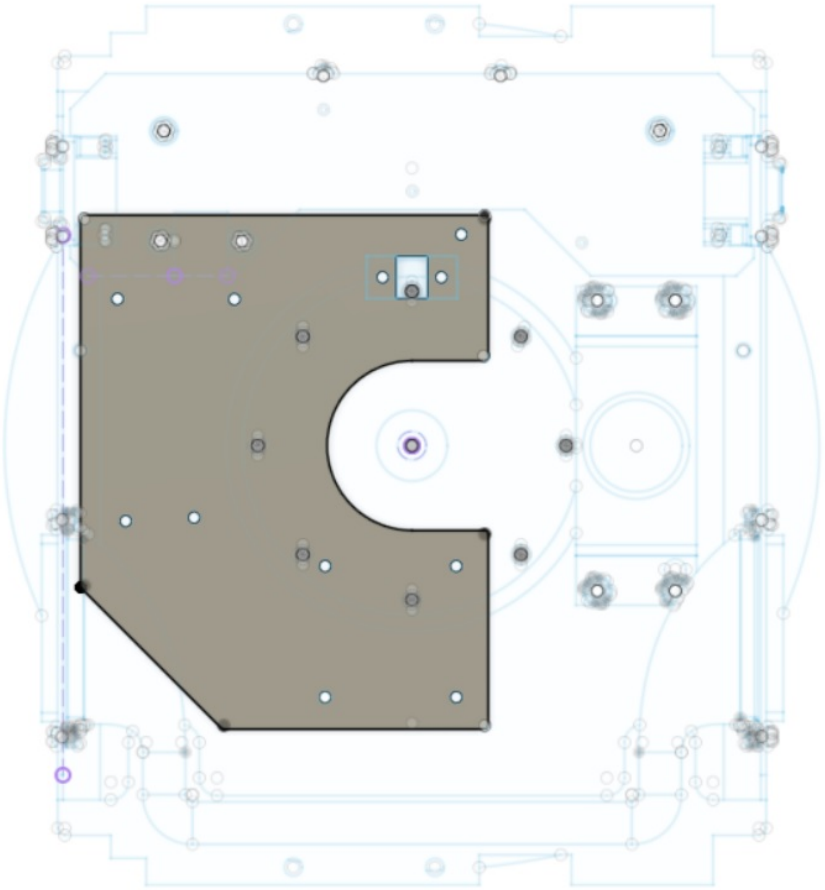

Figure 6.1: Workspace model

Electric scheme

The wiring diagram highlights all the connections between the various components chosen and used, in this section two important libraries have been added, the Arduino footprint and that of the connectors. The connectors were a very delicate matter because it was necessary to choose correctly sized elements in terms of amperage, easy to install on the board but resistant above all. Therefore, the most suitable connectors for this task were searched in the Molex company catalog.
The connectors that have been chosen are the Mini-fit series by Molex, equipped with a hook closure to ensure that they resist vibrations. The following figure shows an example of a 12-way 2-row connector, used for the wire to board connection of the slip-ring.

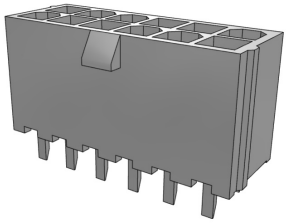

Figure 6.2: Molex header connector

An important feature is that of being able to choose the width of the tracks from the wiring diagram, in this way all the connections to the power supply circuit have wider tracks. The following figure shows the entire wired electrical circuit.

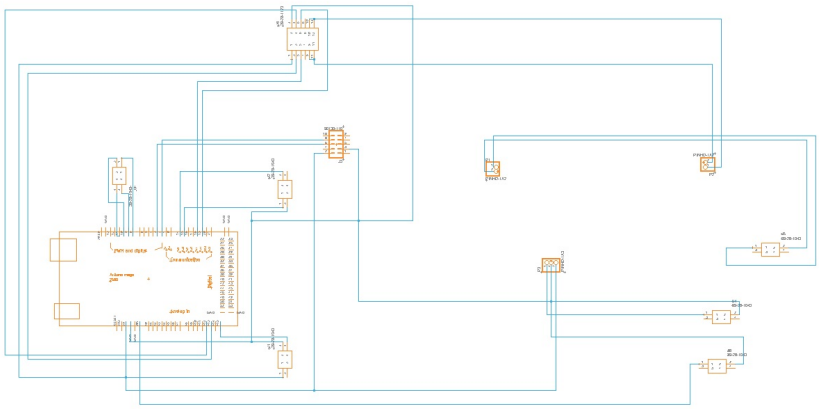

Figure 6.3: Electrical connection scheme

Wires routing

The auto-routing function can automatically generate the tracks, based on the connections made in the previous wiring diagram. By making an effective connection between the two files, the one with the PCB and the one containing the circuit, it is possible to manually position the components in the designed workspace.
The figure below shows the product of this function with small manual changes to avoid artifacts, the red traces represent those in the upper part of the board, while the blue ones are found in the lower part.

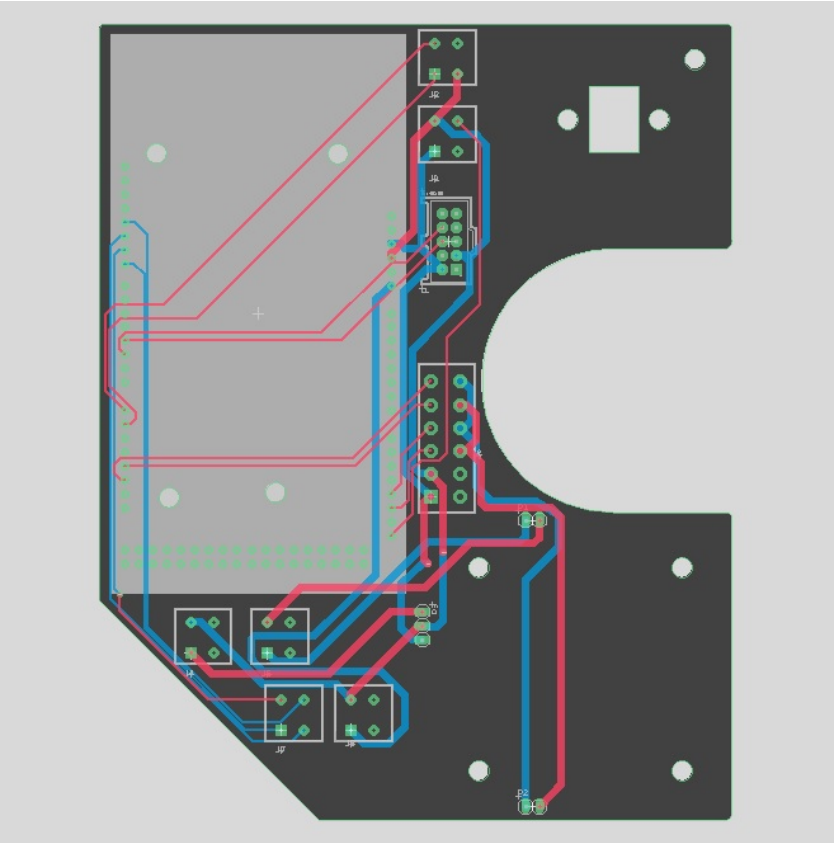

Figure 6.4: Auto-routing product

Realization

Once all the operations described above have been completed, it is possible to generate the Gerber file from the PCB routing, this file contains all the information of the board and is what will be given to the company that will take care of the molding. Finally, the following figure shows the preview of what will return after printing.

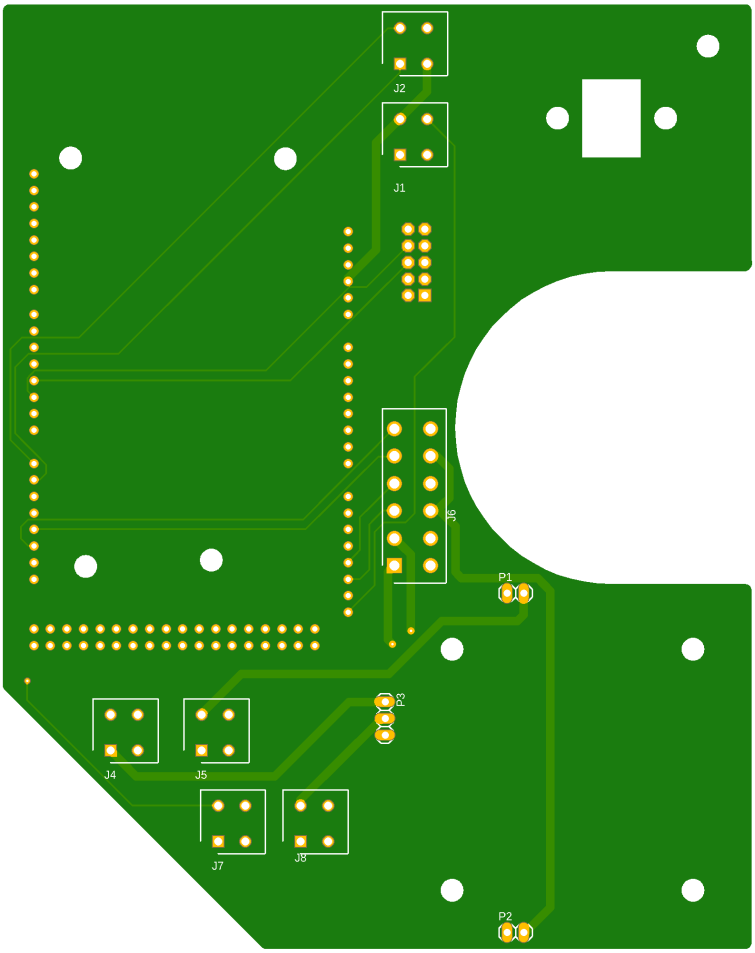

Figure 6.5: Print preview

Chapter 7

Camera implementation

7.1 Video stream management

Blackmagic's Micro Studio Camera 4K is a miniature Super 16mm professional digital film camera that is perfect in any location, the body of the camera is slightly larger than the micro 4/3 lens mount, keeping the design compact as possible, but with maximum professionalism. The basic structure has been made of magnesium alloys, which make the camera light, strong, and durable for any purpose.
There is no SD memory, so it does not have the possibility to record internally and has no built-in module to preview what is being filmed. It is a sensor with a lens mount and a few outputs, but that makes it qualified for this application [22].

Figure 7.1: Blackmagic Micro Studio Camera 4K

The sensor offers 1080 HD resolution and an incredibly dynamic range of up to 13 stops. Unlike DSLRs and sports cameras, whose limited dynamic range tends to dampen light and shadow, the Blackmagic Micro Studio Camera 4K delivers crisp, clear images without compromising shadow detail. With a sensitivity of ISO 1600, quality is always flawless even in poorly lit conditions.
On one side it has a full-size HDMI port and a DB15 connector to which an extension cable can be connected for remote control functions. On the other side of the camera, two Micro SDI ports were placed, the small size of the camera allowed these connections compared to the standard ones [23].

To enable video transmission, the camera output is connected to the slip ring coaxial cable. In this way, the video stream passes through a single cable with SDI convention, instead of the classic HDMI standard. To have greater communication versatility between the system and any type of display device, an important component is added to this chain: the SDI-HDMI converter, which allows transforming the coaxial cable signal into the classic monitor standard.

Figure 7.2: Camera mounted on the gimbal structure

Once the video connection has been tested, the camera is fixed on a cushioned surface, which allows to further reduce the vibrations to obtain more stable images, and finally powered at 12V by the expansion port, avoiding the presence of the battery.

7.2 Remote control methods

The Blackmagic Micro Studio Camera 4K cannot control functions directly from the camera body. It was designed to be remotely controlled as the name "studio" suggests. The primary method of controlling camera exposure, zoom, and lens focus is using the HD-SDI return video feed, which allows a connection between the ATEM switcher and the camera. The first component sends the program video, program audio, intercom audio, tally light, and camera control data at the second. All ATEM products have a Camera Control Unit (CCU) panel in their control software, to adjust the white balance, color, and exposure of the camera and control the zoom and focus on supported goals. The switcher also encodes and embeds camera and lens adjustments in the return video feed sent to the camera.
Therefore, if connected to an ATEM video switcher, there are no problems, as this device manages the control of all the camera parameters. Unfortunately, for reasons of space inside the cockpit and the simplicity of the user of the gimbal, the presence of this device should be avoided [24].

Figure 7.3: Blackmagic ATEM 2 M/E Production Studio 4K

The model shown in the previous figure represents the flagship product presented at NAB 2015 together with the Micro Studio Camera 4k. Nowadays there are cheaper and more compact solutions, with an HDMI connection standard suitable for small multi-camera productions.

Control using HD-SDI return video feed

The first method is realized by Blackmagic and it is a shield that supports sending a subset of data to the camera. The shield has an HD-SDI video input, an HD-SDI video output, and a serial communication interface with Arduino. This is a pluggable solution on a microcontroller board and can be used to send camera and lens control commands over the I2C protocol. This is an interesting control scheme that allows users to control everything from an embedded system.

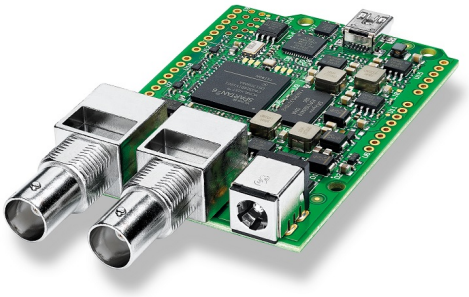

Figure 7.4: Blackmagic 3G-SDI Arduino shield

The LANC Port

The second way to control the exposure and the lens is using the LANC protocol on the camera's expansion port. This is a Sony invention, are consists of a connection between a wired remote controller and the camera by a 2.5mm jack. Most LANC controllers are designed to mount to a tripod and permit the camera operator to control the lens without having to reach the front of the camera.

Figure 7.5: Blackmagic LANC zoom demand

The S.BUS Protocol

The last method to control the camera parameters is using the Futaba S.BUS, a digital protocol used by radio-controlled cars, trucks, and aircraft. The protocol links radio receivers to one or more servo motors in the model.
The Blackmagic Micro Studio Camera 4K is small enough that one target market is drone photography and videography. Using the S.BUS protocol, the camera could connect to the existing radio receiver on the drone and permit an operator on the ground to adjust the camera and lens using their existing radio equipment.

Figure 7.6: Futaba S.Bus system

The S.BUS protocol is proprietary, but it's been reverse engineered. It is basically an inverted TTL-level serial data at 100 kbps with even parity and two stop bits. 16 channels of 11 bits are mapped into 25 data bytes that are sent over the serial link. The only complicated part is mapping the 16 channels of 11 bits into 25 bytes of 8 bits correctly [25]. This protocol could be an excellent approach to apply to this project. By structuring the communication between console and camera in this way, it will be possible to create a very practical one-wire solution, also avoiding the allocation of additional components.

Chapter 8

Hardware adopted

8.1 Arduino Mega 2560

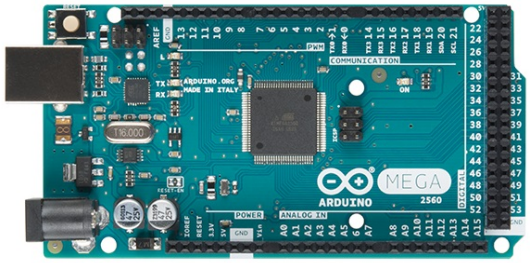

Figure 8.1: Arduino Mega 2560

Microcontroller	**ATmega2560**
Operating voltage	5V
Input voltage (recommended)	7-12V
Input voltage (limit)	6-20V
Digital I/O pins	54 (15 provide PWM output)
Analog input pins	16
DC current per I/O pin	20 mA
DC current for 3.3V pin	50 mA
Flash memory	256 kB (8 kB used by bootloader)
SRAM	8 kB
EEPROM	4 kB
Clock speed	16 MHz

Table 8.1: Arduino specifications

8.2 Motor Driver

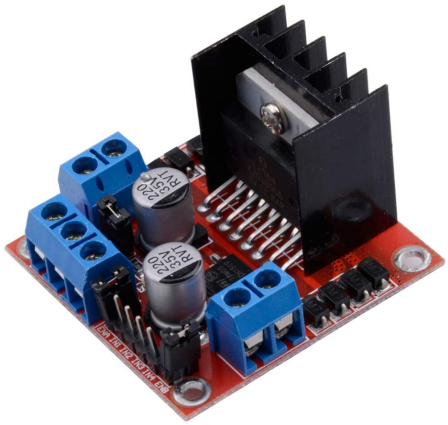

Figure 8.2: H-Bridge motor driver L298N

H-Bridge motor driver	**L298N**
Power supply	5 - 35 V
Input voltage	3.2 ~40 V
Peak current	2 A
Operating current range	0 36 mA
Control signal input voltage range	Low: -0.3V $\leq$ Vin $\leq$ 1.5V High: 2.3V $\leq$ Vin $\leq$ Vss
Enable signal input voltage range	Low: -0.3 $\leq$ Vin $\leq$ 1.5V (invalid) High: 2.3V $\leq$ Vin $\leq$ Vss (active).
Maximum power consumption	20W (@ T = 75 ° C).
Storage temperature	-25 ~130 °C

Table 8.2: L298N motor driver specifications

8.3 DC motors

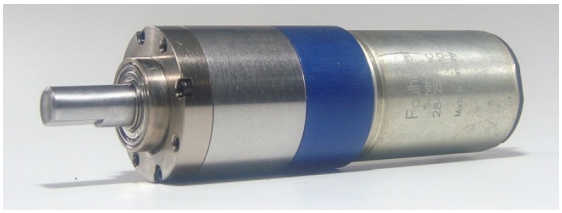

Figure 8.3: Faulhaber DC motor

DC motor	Faulhaber 2642-012 CXR
Nominal voltage	12V
Terminal resistance	1.46 Ω
Output power	22.1W
Efficiency, max.	76 %
No-load speed	5800 min^{-1}
No load current, typ	0.092A
Stall torque	144.6 mNm
Friction torque	1.7 mNm
Speed constant	514 min^{-1}/V
Back-EMF constant	1.9545 mV/ min^{-1}
Torque constant	18.57 mNm/A
Current constant	0.054 A/mNm
Slope of n-M curve	40.4 min^{-1}/mNm
Rotor inductance	135 μH
Mechanical time constant	5.1 ms
Rotor inertia	12 g cm^2
Angular acceleration	121 $\times 10^3$ rad/ s^2
Speed up to	7000 min^{-1}
Shaft diameter	4mm
Mass	114g

Table 8.3: DC motor specifications

8.4 Encoder

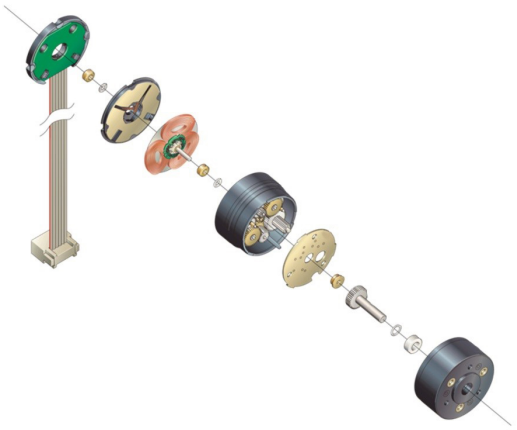

Figure 8.4: Faulhaber encoder

DC motor	Faulhaber IE3-256L
Lines per revolution	256
Frequency range, up to	120 kHz
Supply voltage	4.5 - 5.5 V
Current consumption, typ	17 (max 25) mA
Index pulse width	90 ±45
Phase shift, channel A to B	90 ±45
Inertia of code disc	0.08
Operating temperature range	-40 ~85 °C

Table 8.4: Encoder specifications

8.5 Hall sensor

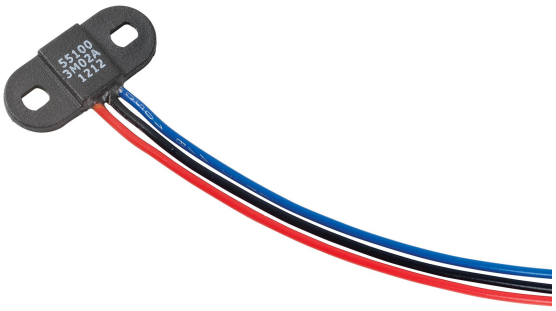

Figure 8.5: Hall sensor

Hall sensor	**Littlefuse 55100-3M02A**
Supply voltage	Absolute ratings -18 ~18 V
	Operate 2.7-24 V
	Overvoltage protection 32 V
Output High voltage	Sinking output
Output Low voltage	0.4 @20mA
Output current	25 mA (Max)
Current consuption	1.1-2.4 mA
Switching speed	12 kHz
Operating temperature	-40 ~100 °C
Distance to magnet	13 mm

Table 8.5: Hall sensor specifications

8.6 Joystick

Figure 8.6: Joystick controller

Joystick	**CH Products 2-Axis Joystick**
Potentiometers resistance	5kΩ ±20 %
Potentiometers power	0.5W
Potentiometers linearity	±5 %
Operating temperature ranges	-25 ~85°C
Number of Axes	2
Joystick Travel	±30°

Table 8.6: Joystick specifications

8.7 Arduino display LCD

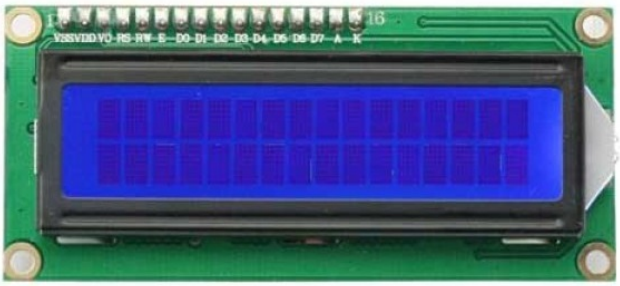

Figure 8.7: Arduino LCD Display 16x2

Display Pin configuration		
PIN	Name	Function
1	GND-VSS	Ground pin connected
2	VDD	Powers the LCD with +5V
3	Vo	Decides the contrast level of display
4	RS	Register Select
5	R/W	Read, Write
6	E	Enable
7-14	DB0-DB7	Data Bus
15	LED+	Backlight LED +5V
16	LED-	Backlight LED ground

Table 8.7: Display pin configuration

8.8 Video camera

Figure 8.8: Blackmagic video camera

Camera	BMMSC 4K
Sensor Size	13.056mm x 7.344mm
Shooting Resolutions	3840 x 2160, 1920 x 1080
Frame Rates	HD 1080p23.98, 24, 25, 29.97, 30, 50, 59.97, 60, 1080i50, 59.94, Ultra HD 2160p23.98, 24, 25, 29.97, 30
Focus	Remote focus control via expansion connector or ATEM Switcher CCU protocols via SDI
Iris Control	Iris control via Up and Down buttons, remote control via expansion connector or ATEM Switcher CCU protocols via SDI
Lens Mount	Active MFT mount
Controls	5 control buttons including Set, Up, Down, Menu, and Power
Microphone	Integrated stereo microphone

Table 8.8: Blackmagic specifications

Camera connections	**BMMSC 4K ports**
SDI Video Output	1 x 6G-SDI 10-bit 4:2:2 via DIN 1.0/2.3
SDI Video Input	1 x 6G-SDI 10-bit 4:2:2 via DIN 1.0/2.3
SDI Audio Output	2 channel embedded audio support in SDI stream
HDMI Video Output	1 x HDMI Type A output
HDMI output	always in HD 1080p.
HDMI Audio Output	2 channels 48 kHz and 24-bit
Expansion Port DB-HD15	Power input
	LANC input
	1 x S.Bus channel input
	PTZ output
	B4 lens control output
	Genlock input
Audio Input	1 x 3.5mm stereo audio mic or line level
Audio Output	1 x 3.5mm stereo connector for headphones
Remote Control	Remote control over SDI
	via ATEM Switcher CCU protocols
	or via expansion port using LANC or S.Bus
Computer Interface	USB Mini-B port
	(for software updates and configuration)

Table 8.9: Blackmagic connection specifications

8.9 Video ports converter

Figure 8.9: Blackmagic Mini Converter

Converter	SDI to HDMI 6G
Video Input	1 x SD, HD or 6G-SDI.
	1 x ALT SDI
Video Output	1 x HDMI type A
	1 x 6G-SDI
Analog Audio Outputs	2 channels of balanced analog audio
Digital Audio Outputs	4 channels of AES/EBU digital audio
Multi Rate Support	Auto detection of SD, HD or 6G-SDI.
Updates and Configuration	USB
Reclocking	Yes
Power Supply	12 V

Table 8.10: SDI to HDMI converter specifications

Chapter 9

Conclusions and future works

The development of this project is based on the realization of the software in the MATLAB/Simulink environment, achieving a system that works according to the given requirements composed of cheap and easily available hardware components, thus ensuring simplicity in replacement in case of damage. This system can self-calibrate, rotate 360 degrees on the PAN angle, and limit the TILT angle movement to a range of -110 to +96 degrees. The management of these functions has been entrusted to a console which will be placed in the control cabin and controlled by an operator. The communication between these elements takes place via a serial protocol. After having tested the correct functioning of these parts, a PCB has been designed on which the Arduino, the driver, and the connectors will be plugged to transform all the connections into tracks. This solution will lead to more compact hardware that can be quickly installed on the SmartBay platform. This feature makes it suitable for in-flight tests with real physical conditions. Finally, the last topic addressed is the interface with the camera, this has been installed and tests have been carried out on the functioning of the video transmission.
The work described in this document may be complemented by further developments and extensions, which would allow for the addition of components or the implementation of additional functions. Since SmartGimbal has been designed not only for aerial monitoring but also for tracking a target, an image processing system based on machine learning algorithms can be added to the system, so that the control process recognizes the target when it appears in the frame and changes the trajectory by following it. In this way, it would be possible to expand the tracking to a multitude of targets related to different classes of objects by recognizing them based on specific details.

Bibliography

[1] LinkedIn Digisky Srl. URL: https://www.linkedin.com/company/digisky/mycompany/. (accessed: 06.11.2021) (cit. on p. 1).

[2] Digisky Srl Website. URL: https://www.digisky.it/. (accessed: 06.11.2021) (cit. on p. 2).

[3] Annachiara Greco. *Development of the SmartGimbal Control System for the SmartBay Platform*. URL: https://webthesis.biblio.polito.it/13135/. (accessed: 06.11.2021) (cit. on pp. 3, 7).

[4] Salman Abdul Moiz Mohammed Rizwanullah N Md Jubair Basha. *Model Based Software Development: Issues & Challenges*. URL: https://arxiv.org/pdf/1203.1314.pdf. (accessed: 07.11.2021) (cit. on p. 5).

[5] MathWorks Support Team. URL: https://it.mathworks.com/matlabcentral/answers/440277-what-are-mil-sil-pil-and-hil-and-how-do-they-integrate-with-the-model-based-design-approach. (accessed: 07.11.2021) (cit. on p. 6).

[6] Radim Hýl and Renata Wagnerová. «Fast development of controllers with Simulink Coder». In: *2017 18th International Carpathian Control Conference (ICCC)*. (accessed: 07.11.2021) (cit. on p. 8).

[7] ADMET. URL: https://www.admet.com/open-loop-vs-closed-loop-systems-materials-testing-industry/. (accessed: 15.11.2021) (cit. on p. 11).

[8] Glen Alleman. *Is There an Underlying Theory of Software Project Management? (A critique of the transformational and normative views of project management)*. URL: https://www.researchgate.net/publication/2537163_Is_There_an_Underlying_Theory_of_Software_Project_Management_A_critique_of_the_transformational_and_normative_views_of_project_management. (accessed: 08.11.2021) (cit. on p. 12).

[9] Scott Zhuge. *PID Control Theory*. URL: https://www.crystalinstruments.com/blog/2020/8/23/pid-control-theory. (accessed: 09.11.2021) (cit. on p. 13).

[10] Giovanni Testolin. *Controllori PID e tecniche "anti wind-up"*. URL: http://tesi.cab.unipd.it/44095/1/Testolin.Giovanni.1007244.pdf. (accessed: 09.11.2021) (cit. on p. 15).

[11] Titus Rotich. *Dynamic Model of a DC Motor-Gear-Alternator (MGA) System*. URL: https://www.researchgate.net/publication/308319754_Dynamic_Model_of_a_DC_Motor-Gear-Alternator_MGA_System. (accessed: 10.11.2021) (cit. on p. 16).

[12] Pratyusha Biswas Deb & Oindrila Saha & Sajan Saha & Shaon Paul. *Dynamic Model Analysis of a DC Motor in MATLAB*. URL: https://www.ijser.org/researchpaper/Dynamic-Model-Analysis-of-a-DC-Motor-in-MATLAB.pdf. (accessed: 10.11.2021) (cit. on p. 17).

[13] *Esperimentanda*. URL: https://www.esperimentanda.com/come-creare-un-controller-pwm-con-arduino-per-luci-motori-carichi-dc-ac-controllati-via-software-computer/. (accessed: 10.11.2021) (cit. on p. 18).

[14] *How To Mechatronics*. URL: https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/. (accessed: 10.11.2021) (cit. on p. 18).

[15] *Electronics Tutorial*. URL: https://www.electronics-tutorials.ws/electromagnetism/hall-effect.html. (accessed: 12.11.2021) (cit. on p. 19).

[16] *Eltran*. URL: https://www.eltra.it/encoderpedia-glossario-tecnico/cose-lencoder-magnetico/. (accessed: 11.11.2021) (cit. on p. 20).

[17] Fabio Nelli. *La comunicazione seriale*. URL: https://www.meccanismocomplesso.org/la-comunicazione-seriale/. (accessed: 16.11.2021) (cit. on p. 23).

[18] *Arduino website*. URL: https://www.arduino.cc/en/Tutorial/BuiltInExamples/Debounce. (accessed: 19.11.2021) (cit. on p. 43).

[19] *MathWorks Help Center*. URL: https://it.mathworks.com/help/stateflow/ug/debouncing-signals.html. (accessed: 19.11.2021) (cit. on p. 44).

[20] *FormLabs*. URL: https://formlabs.com/it/blog/tutorial-fusion-360-nozioni-di-base-e-consigli-per-la-stampa-3d/. (accessed: 15.11.2021) (cit. on p. 52).

[21] *SmeUp*. URL: https://www.smeup.com/magazine/blog/autodesk-eagle-editor-grafico/. (accessed: 15.11.2021) (cit. on p. 52).

[22] *Blackmagic Website*. URL: https://www.blackmagicdesign.com/products/blackmagicmicrocinemacamera. (accessed: 27.11.2021) (cit. on p. 57).

[23] Chris Monlux. *Blackmagic Design Micro Studio Camera 4K Review*. URL: https://www.videomaker.com/article/r02/18632-blackmagic-design-micro-studio-camera-4k-review. (accessed: 27.11.2021) (cit. on p. 58).

[24] *Blackmagic Camera Control*. URL: https://documents.blackmagicdesign.com/DeveloperManuals/BlackmagicCameraControl.pdf. (accessed: 27.11.2021) (cit. on p. 59).

[25] Glen. *USB Knob Box Doubles as a Blackmagic Designs Camera Remote*. URL: https://bikerglen.com/blog/usb-knobs-that-double-as-a-blackmagic-remote/. (accessed: 27.11.2021) (cit. on p. 61).

Ringraziamenti

A conclusione di questo elaborato, vorrei dedicare uno spazio a chi, con dedizione e pazienza, ha sempre creduto in me supportandomi sempre. A loro va un sentito ringraziamento. Senza queste persone, questo lavoro, frutto del mio percorso universitario non esisterebbe nemmeno.

Alla mia relatrice Corpino Sabrina, che ha saputo guidarmi, con suggerimenti pratici nella stesura dell'elaborato, donandomi fiducia e reputandomi pronto per la conclusione del mio cammino.

Al mio tutor Paolo, che mi ha fatto conoscere ed entrare a far parte di quella realtà che è Digisky. Sempre disponibile e pronto a darmi le giuste indicazioni in ogni fase della realizzazione di questo progetto. Insieme a lui, un ringraziamento va anche ai miei colleghi Fabio, Matteo, Luca e Gianluca, i quali hanno saputo integrarmi e con cui ho condiviso momenti più o meno felici, e hanno sempre fornito tutto l'aiuto di cui avessi bisogno.

Ai miei amici e ai colleghi di corso, le persone con cui ho condiviso attimi di gioia e di tristezza, ma che nonostante tutto sono rimasti accanto a me in questi anni. Senza di loro, sarebbe stato tutto più cupo: grazie per avermi trasmesso entusiasmo e coraggio.

Ai miei genitori Antonio e Irene, il pilastro fondamentale nella mia vita e mia nonna Laura che ha fatto parte delle fondamenta su cui si posa tutto quello che ho costruito finora. Infinitamente grazie a voi, senza i quali non sarei arrivato dove sono ora, mi avete sempre dato il giusto sostegno, appoggiando ogni mia decisione e dandomi il supporto morale nei momenti di sconforto. Questa tesi è per voi e a voi dedico la gioia che ho nel mio cuore nel tagliare questo traguardo importante.

Ad Arianna, che con la sua dolcezza, la sua intelligenza e il suo amore puro, mi ha sopportato, supportato, calmato, spronato e incoraggiato sempre. Sei sempre stata accanto a me non lasciandomi mai solo e facendomi sempre sentire quanto tu credessi in me, giorno dopo giorno. Dal profondo del cuore, grazie oggi e sempre.