UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

Modeling, testing and validation of viscoelasticity measurement system for VesEvo

Control Lab Report

submitted by students:
Salvatore Granata - P38000219
Carlo Cianflone - P38000165
Francesco De Maio - P38000172
Davide Rubinacci - P38000182

supervised by:
prof. **Raffaele Iervolino**

Naples, 2023

Acknowledgements

The authors of the report express sincere gratitude to Professor Eng. Raffaele Iervolino for his invaluable guidance and support throughout the duration of this project. His dedication to provide us with the necessary tools and knowledge to develop this cutting-edge project has been truly helping and inspiring, while his availability for clarification and mentorship significantly contributed to the success of this project.

Additionally, we extend thanks to the Vesevo Company for their generous contribution. Their provision of articles exploiting the system's functionality and detailed insights into the implementation of the actuation system greatly enriched our understanding. Moreover, their enthusiastic reception and keen interest in our developmental ideas were truly inspiring.

Abstract

As part of a fruitful collaboration between the company and the academic environment in order to equip tomorrow's engineers with useful experience and skills, the group worked with VESevo in order to:

- provide a useful service to the company by helping them solving issues proper of the field of study of the team members;
- put into practice the skills acquired during the Control Lab course in the field of Model Based Design;
- acquire new skills by being in a work environment which is probably not far to the one in which the team members may be working in the future.

The company asked the team to work on a device used for data retrieving. The team worked on modernising the system model in the Matlab-Simulink environment, also modelling previously ignored dynamics, after which they focused on finding multiple solutions that would allow the automatic implementation of the device. In simulation, the choice fell on:

- a linear actuator based on a solenoid with magnetic effect and return spring;
- a DC linear actuator, consisting of a DC motor, assisted by an H-bridge, and a gear box to transform the motion from rotary to linear with an appropriate gear ratio.

Subsequently, following the development of control laws for the two solutions, the team worked on the control of the actual device using an actuator similar to the second option developed in the design phase. However, given the impossibility of characterising the actuator with parameters that could be used in simulation, the control law for the final actuator was developed and implemented via the Arduino IDE. To close on a high note, a script was implemented in Python to control the actuator and read sensor values, bypassing the microcontroller mounted on the device

Acronyms

During the reading of the report, you might find these terms written as acronyms. Here are the meanings:

DC Direct Current

SIL Software In the Loop

PIL Processor In the Loop

HIL Hardware In the Loop

IDE Integrated Development Environemnt

PID Proportional-Integral-Derivative action

PWM Pulse Width Modulation

Contents

Chapter 1

Introduction

VESevo stands for “Viscoelasticity Evaluation System evolved”. The company’s main work revolves around their flagship product, which is the device that will be described in detail in section 2.2, used for viscoelastic materials characterization.

The company is actively working on developing solutions that automate the data collection process for this device.

For this reason, as students of ”Control Lab” course at the Federico II University, we collaborated with VESevo to implement one of the proposed solutions.

As it will be shown, our work started by incorporating a moderation phase using Simulink software. Subsequently, we progressed to develop solutions employing embedded microcontrollers such as SIL (Software-in-the-Loop), PIL (Processor-in-the-Loop), and HIL (Hardware-in-the-Loop), completing the V-model development cycle. Finally, we validated the achieved results by integrating and testing our work on their VESevo system.

Our goal was to create a solution enabling the device’s use by both a human user on the track and a robotic arm, thereby paving the way for its deployment in industrial environments. This collaborative effort aimed to enhance the versatility and applicability of VESevo’s device across diverse operational settings.

Chapter 2

Preliminar analysis

2.1 The problem

The company faced 2 main problems. Firstly, their Simulink-based digital twin model operated on an older version of MATLAB, rendering it incompatible with the current iteration due to issues with toolbox compatibility. This outdated digital twin was confined to local validity and lacked the interface capabilities required for external microcontroller command and interaction.

Secondly, their measurement system posed operational inefficiencies as it remained a manual, step-by-step process devoid of automation. This lack of automation meant that various phases within the measurement system were handled sequentially, leading to potential bottlenecks and reduced efficiency in data collection and processing.

In essence, the hurdles stemmed from both technological constraints—pertaining to the outdated Simulink model and its limitations in interfacing with external microcontrollers—and procedural inefficiencies due to the manual nature of their measurement system. These challenges hindered seamless integration, real-time control, and the holistic automation essential for optimizing their viscoelasticity assessment device's performance and applicability across diverse operational landscapes. For this reason, the main requests made by the company to the working group can be broken down into three points:

- Design of the digital twin model of the entire system, including the electromagnet;
- Design and implementation of an actuation system for the automated surveys;
- Performing tests on the physical system, as required by the Hardware in the loop logic.

2.2 The Device

The VESevo (Viscoelasticity Evaluation System Evolved) is an innovative device capable of characterizing the mechanical properties of the compounds within specific ranges of temperature and frequency.

Figure 2.1: VesEvo Measurement System

To achieve this purpose, the device has been developed following these focal points:

- Ensure the analysis of phenomena characterized by high frequency dynamics;
- Correctly correlate the sensor displacement measurement with the temperature of the tested material;
- Ensure high repeatability of the measurements.

The main components of the device are 6:

1. **Rod-Spring Mechanism**: the main part of the device consists of a rod-spring mechanism The mechanism is composed of a metal rod that slides in a metal guide allowing it to rebound freely after hitting the material with the indenter. The rod is characterized by having extremities of different geometry:

   - The upper end is characterized by the presence of a flat head that allows the rod to be hooked by the loading system and the micro vibrometer laser to measure its displacement;
   - The lower part is characterized by the presence of a spherical indenter that will go into contact with the material to be tested.

   An important aspect of the design was the choice of the materials for the mechanism in order to reduce the friction phenomena, which could affect the rebound between the guide and the rod. The main feature of the mass-spring mechanism is that it is all contained inside a cartridge, it is easily removable and replaceable both in case of breakage of the mechanism or of wanting to change the configuration of the device.

2. **Vibrometer Laser**: the displacement of the rod is measured by micro vibrometer laser that uses the physical principle of Doppler effect: it consists in the apparent change,

respect to the original value, of the frequency or wavelength perceived by an observer reached by a wave emitted by a source that is moving respect to the observer himself. It was chosen for its compact dimensions and very high-frequency response.

3. **Temperature sensor**: the temperature of the compound during a single test must be acquired together with the displacement data because the viscoelastic properties are influenced by the temperature. The sensor used is an infrared pyrometer that was chosen for the following reasons:
   - Compact size;
   - It can measure the temperature without touching the sample so as not to alter its dynamics. This is made possible because the heat, or more precisely thermal radiation, is an electromagnetic radiation emitted from the surface of an object and is caused by its temperature. During the dispersion it is possible to measure the energy carried by these electromagnetic waves thus allowing us to measure the temperature without touching the body concerned.

4. **Support bas**e: the base of the device, the only part in contact with the material to be tested, has the shape of a cylinder with a lateral hole; this geometry was chosen for the following reasons:
   - The wide base allows the device to rest on the sample in a stable way;
   - The cavity of the cylinder allows the indenter to get in touch with the sample;
   - The side hole allows the mounting of the temperature sensor, which was not possible to insert inside due to its size and working.

5. **Loading and Unhooking System**: the high repeatability of the measurements can be guaranteed only if during each test the motion of the rod starting from the same initial position, in order to comply this condition, the Vehicle Dynamics research group of the Industrial Engineering department of the University of Naples Federico II has patented a magnetized system, a magnet, that is mounted on a suitable slider inside the case, allows the lifting of the rod until the magnetic force is greater than that exerted by the spring contained in the rod-spring mechanism.

6. **Handle**: the handle is used by the operator to bring the mass to the starting point.

2.3 Pre-existing digital twin

As a starting point from which realise the new modern digital twin, the company provided us with the old version of it, realised in Matlab 2018b. In this version, the whole system was simulated through the use of a launcher and a Simscape model. The launcher can be used to easily execute many different operations:

- Initialization of simulation parameters, like sampling frequency, total time required for the simulation and parameters regarding the starting configuration of the system, like initial vertical position and velocity of the rod;

- Initialization of constructive parameters, like the mass of the rod, stiffness and damping coefficients and gravity acceleration;
- Initialization of material's stiffness and damping coefficients in order to characterize the contact dynamics;
- Execution of the simulation and data retrieving;
- Plotting of the mass' position, velocity and acceleration (figure 2.5).

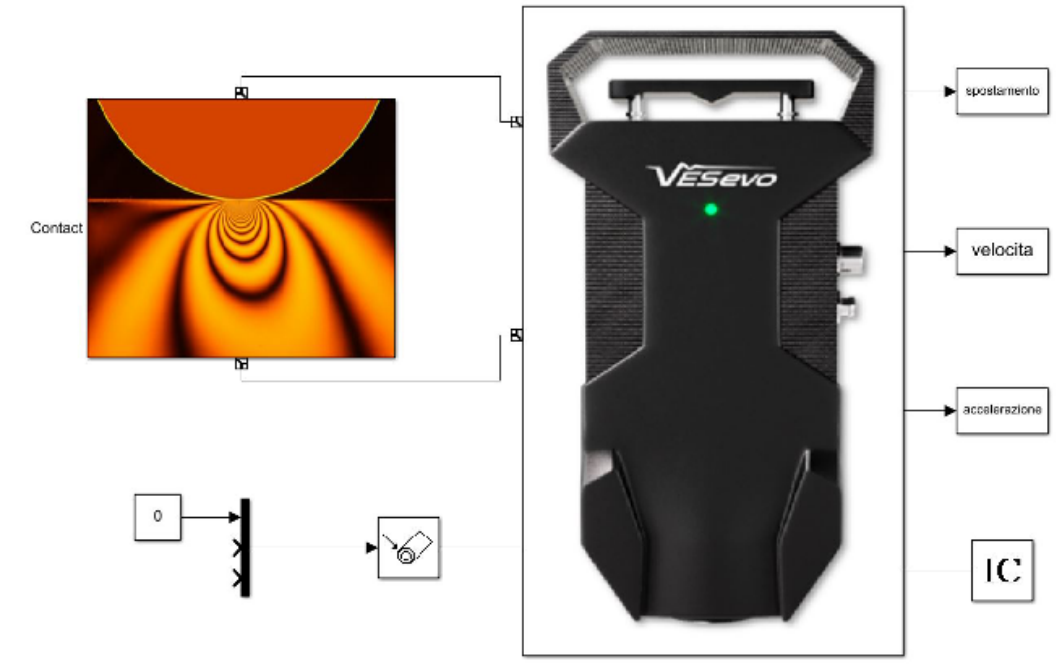

Figure 2.2: Overall view of the Matlab 2018b digital twin

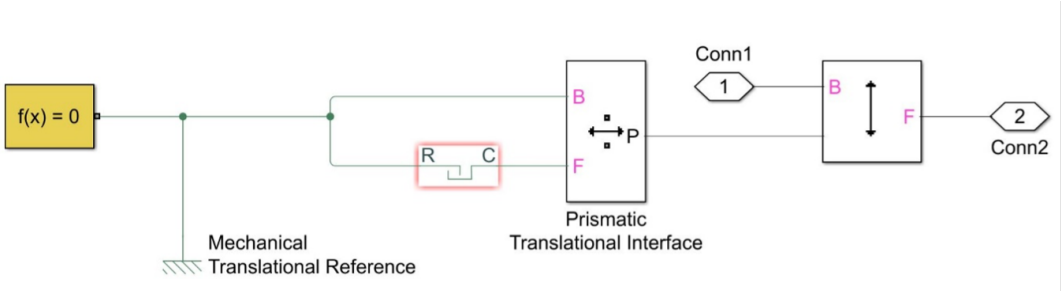

Figure 2.3: Modelling of the contact dynamics in the Matlab 2018b digital twin

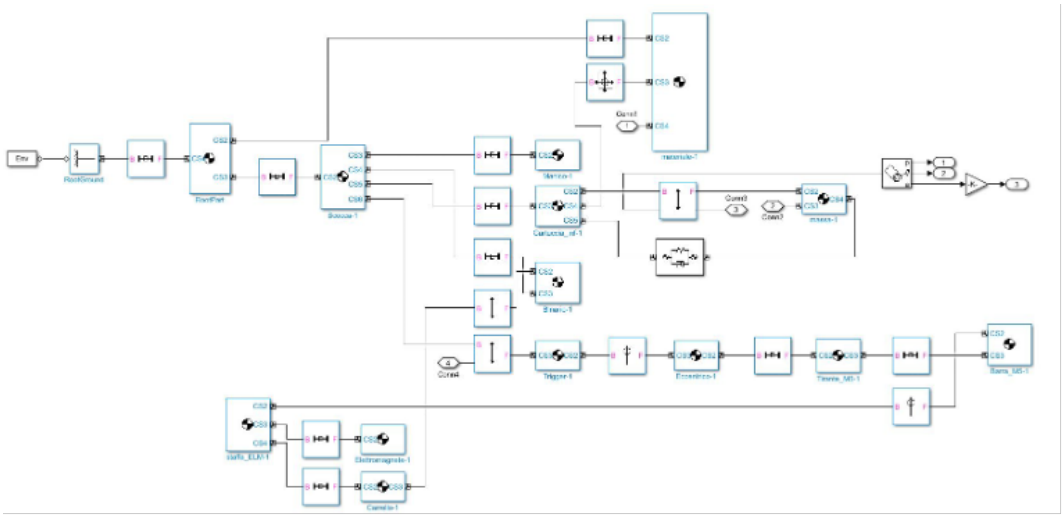

Figure 2.4: Modelling of the device in the Matlab 2018b digital twin

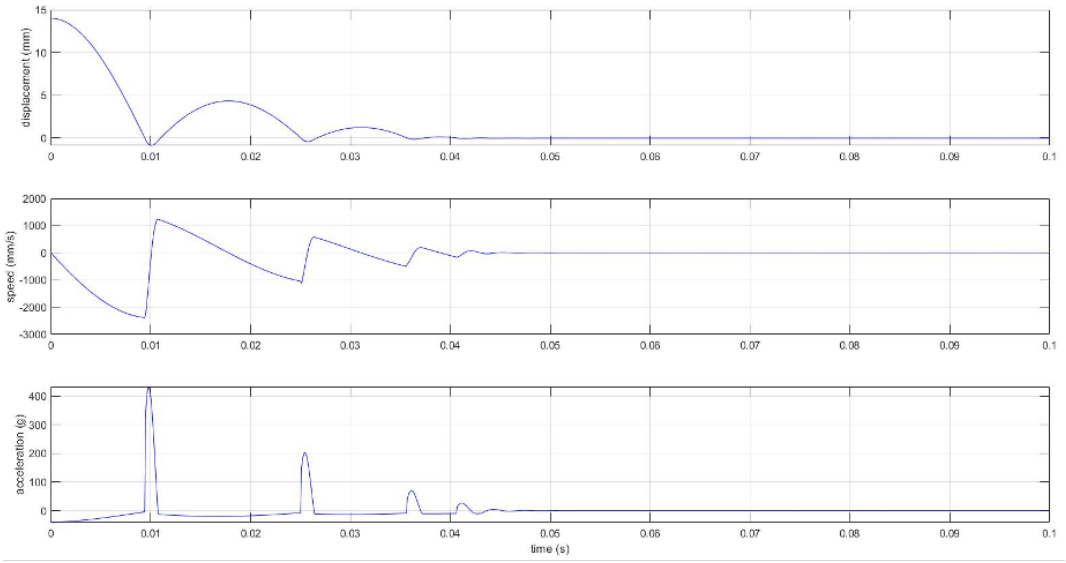

Figure 2.5: Plots of vertical position, velocity and acceleration in time for the rod

Focusing on the Simscape model, we can see as the system is modelled as two block interacting with each other (figure 2.2), one to simulate the contact dynamics between the rod tip and the material under examination (figure 2.3) and the other to model the whole device (figure 2.4). The latter is structured as many body blocks interconnected by different kinds of joint. Each body block has its contact points' pose expressed in world frame as:

- For the position, the components of the displacement's vector from the origin of the world frame;
- For the orientation, the rotational matrix.

Regarding the mass properties, each body block is endowed with the mass of the modelled piece and its inertia tensor. The model simulates the free fall of the rod from a starting vertical position of 14 mm. No dynamics between the rod and the electromagnet nor the uplifting of the rod to this desired starting point is modelled. Due to the many Matlab's updates occurred,

many blocks are no longer available. Moreover, the proposed solution for modelling of the contact dynamics is simplistic thus not completely reliable. All these reasons justified the need of a modern digital twin.

Chapter 3

Model based design

The model based design is a typical approach upon which it is possible to design complex controls for a physical system starting from a simulation environment. It provides an efficient approach for establishing a common framework for communication throughout the design process while supporting the development cycle (figure 3.1).

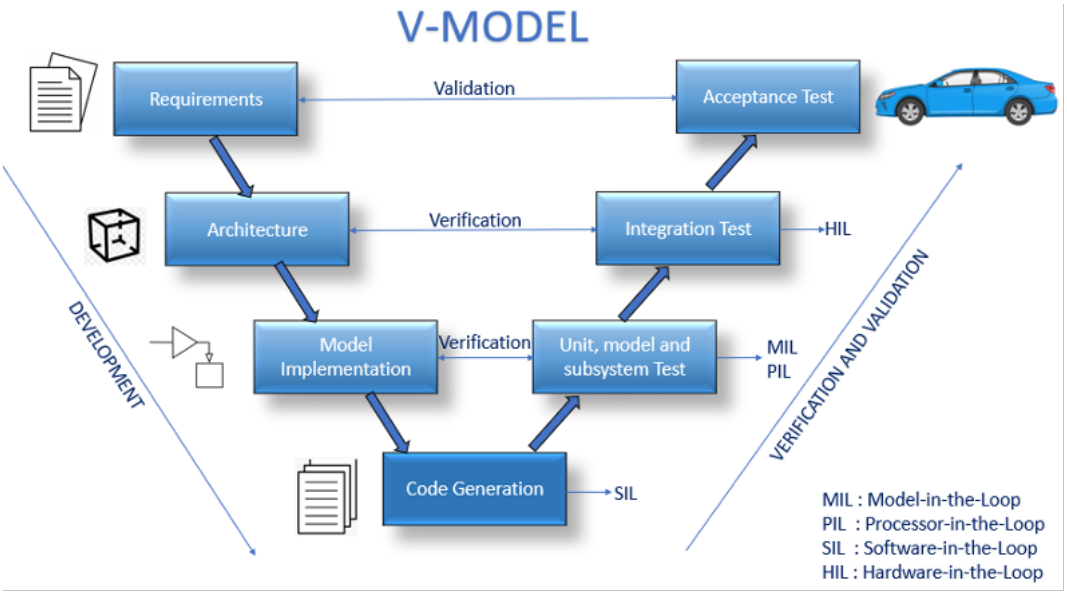

Figure 3.1: Systems development lifecycle (V-model)

In model based design of control systems, development is manifested in these four steps:

1. Modelling of the plant to control;
2. Analyzing and synthesizing a controller for the plant;
3. Simulating the plant and controller;
4. Integrating all these phases by deploying the controller.

In this chapter, we'll focus on three stages of the model based design:

- **Model in the loop (MIL)**: the modern digital twin is realized and two solutions of actuation are implemented in simulation;
- **Software in the loop (SIL)**: the code executing the control law is generated from the Matlab-Simulink environment and is applied to control the plant;
- **Processor in the loop (PIL)**: the code is loaded on a board, which communicates with the Matlab-Simulink environment in order to control the simulated system.

Regarding the **Hardware in the loop (HIL)** stage, due to the impossibility of retrieving the needed parameters in order to simulate the real actuator, it couldn't be executed for the designed controllers. The code for the real actuator has thus been written through Arduino IDE and will be analyzed in chapter 4.

3.1 Model in the loop

In this section we'll focus on the process which led to the design of a new digital twin for the system in Matlab 2023a, using modern blocks and a brand new way of modelling the dynamics between the rod and both the material and the electromagnet. Then we'll move onto two proposals of actuation solution and the related control law.

3.1.1 Modern digital twin

In order to correctly design the modern digital twin the team looked to the old version, trying to recreate all the connections between the blocks correctly. Since the Body block is not available in Matlab 2023a, upon the reception of all the components stl from the company it has been possible to use the File Solid block in order to integrate all pieces in Simscape. Due to the format of the 3D models given by the company, it hasn't been possible to set new frames with respect to the single component as wished. Moreover, only the homogeneous rotational matrices with respect to world frame are known for each component and joint. As a consequence, it has been necessary to exploit the known matrices as showed in figures 3.2 and 3.3 in order to correctly replicate the connections.

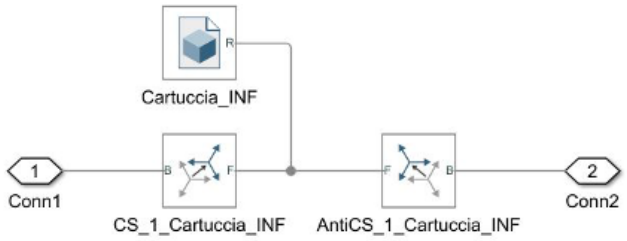

Figure 3.2: Example of connection of a component to the Simscape model

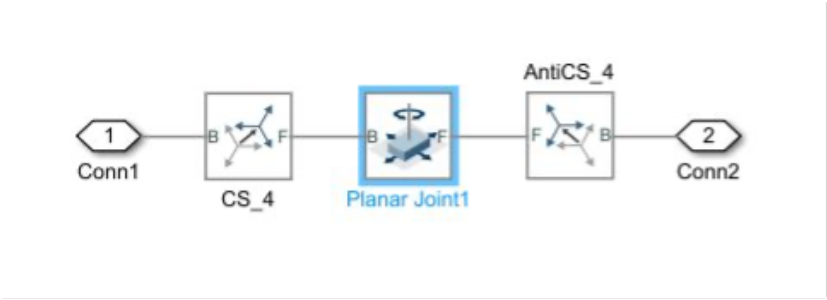

Figure 3.3: Example of connection of a joint to the Simscape model

The File Solid block (figure 3.4) allows the user to visualize the component 3D model upon selection of the desired stl file and to set parameters regarding the component inertia. It doesn't allow the use of the known inertia tensors but, by knowing the mass of each piece, it was possible to exploit the knowledge about the component geometry to automatically compute all inertia parameters of each piece. The automatic computation of these parameters was proven trustable by computing the inertia tensors from the obtained parameters and comparing them to the ones used in the old digital twin. The block allows also the export of the component convex hull, which has been used to realize the contact between rod and material.

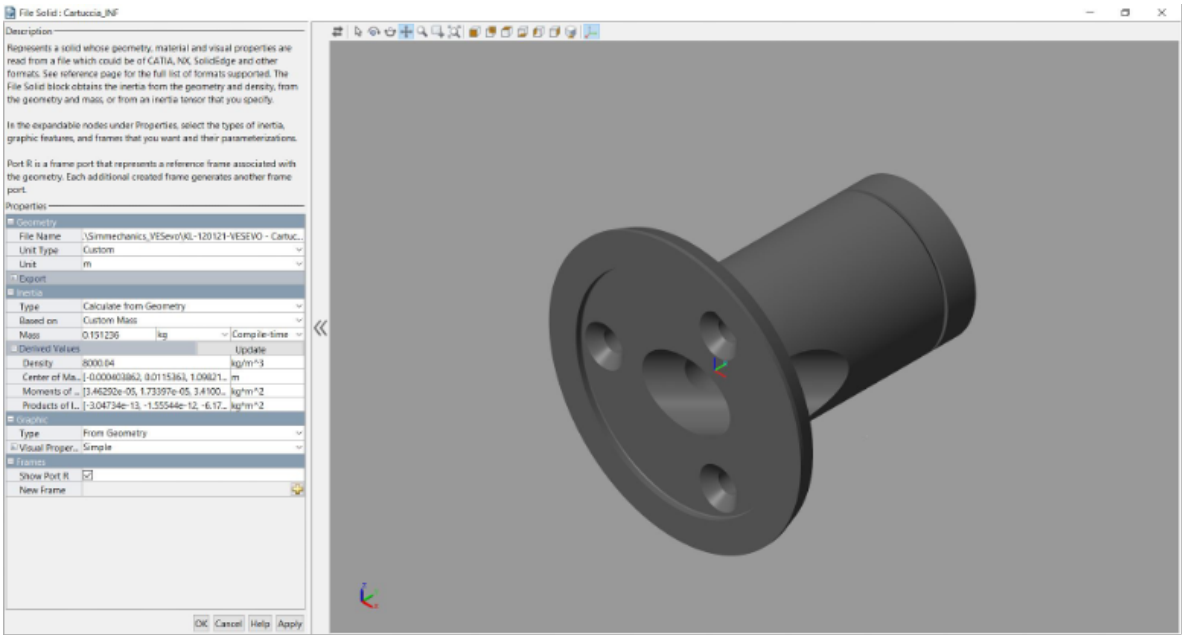

Figure 3.4: Layout of the File Solid block

In order to model the latter a dedicated Simscape block called Contact Force has been used. The block, upon being connected to the convex hulls of the two rigid bodies, simulates the contact forces between them using the same parameters previously used by the translational hard stop in Matlab 2018b.

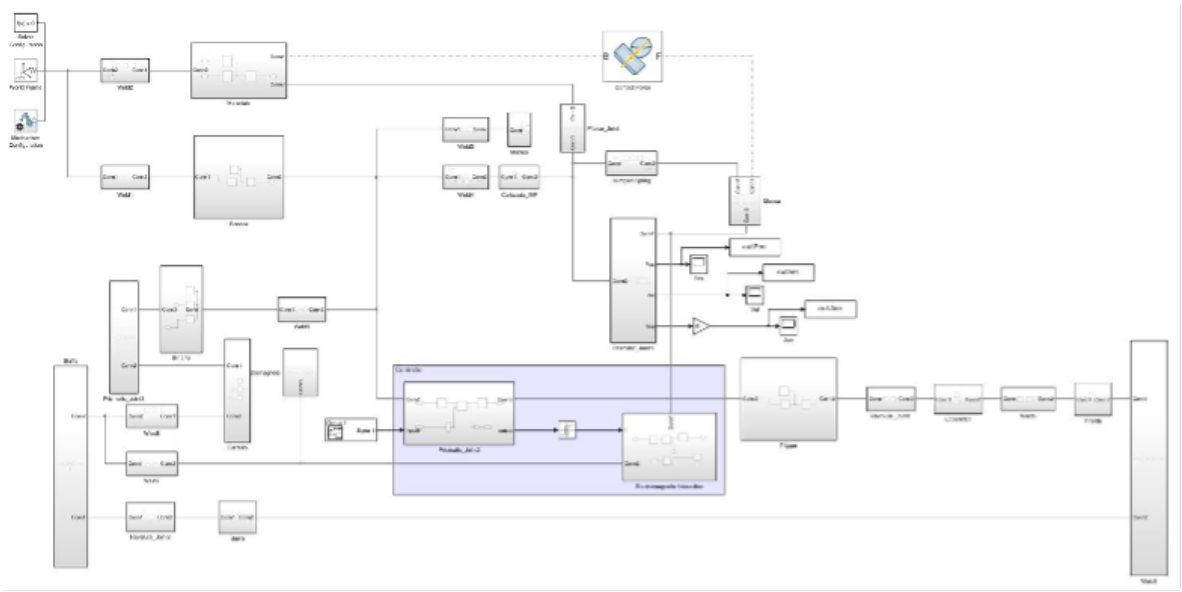

Figure 3.5: Overall overview of the Matlab 2023a digital twin

As requested by the company, the dynamics of the electromagnet have been modelled. The interaction between the electromagnet and the rod has been modelled as a weld joint. This choice is due to the fact that among the block parameters there's a mode configuration section: the mode of the joint can assume two value between "normal" and "disengaged" or being provided by input. This way it has been possible to disengage the weld joint when desired, which it has been valued sufficient to model the electromagnet interaction with the rod. In particular, a sort of passive control has been implemented for which the electromagnet shuts down after 0.75 seconds (delay suggested by the company in order to avoid perturbation of the following surveys) upon the trigger reaching the height of 14 mm, and turns on again with the same delay after the same component return at height zero. This behaviour has been modelled through a relay. In order to verify that the system behaves as desired, the position of the trigger has been varied according to a reference signal and the retrieved values of position, velocity and acceleration of the rod have been compared with the ones in figure 2.5, showing the oscillations' characteristics to be perfectly equivalent:

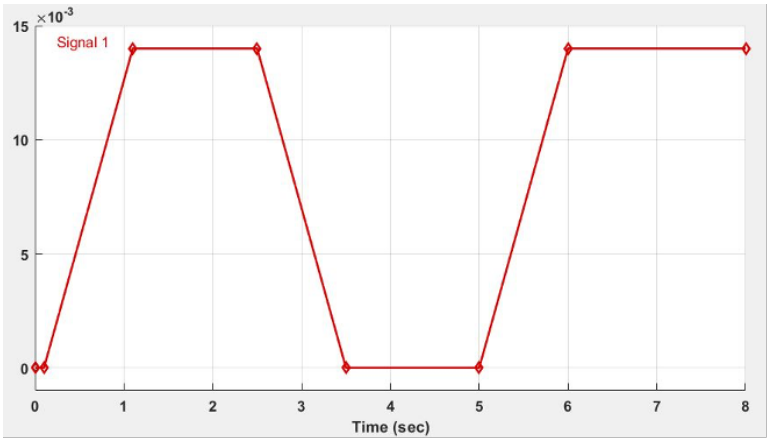

Figure 3.6: Input reference for the position of the trigger

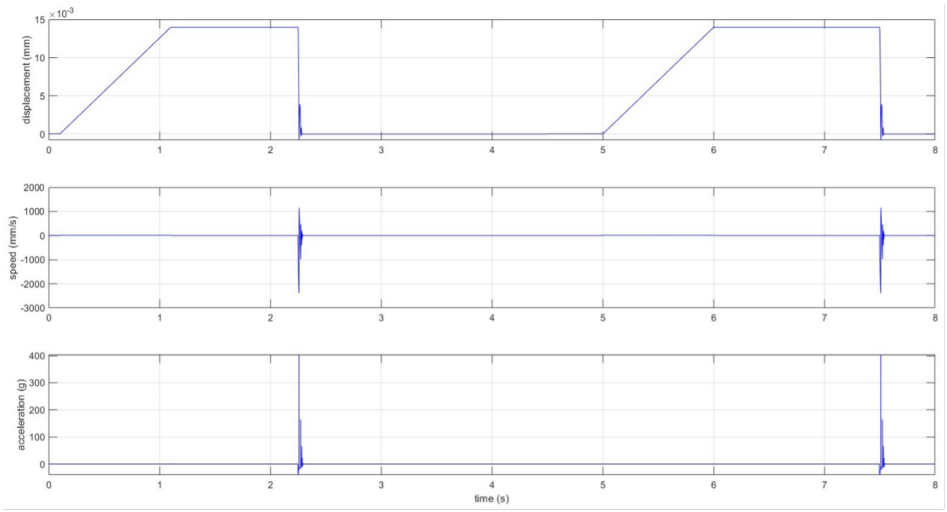

Figure 3.7: Plots of vertical position, velocity and acceleration in time for the rod in Matlab 2023a

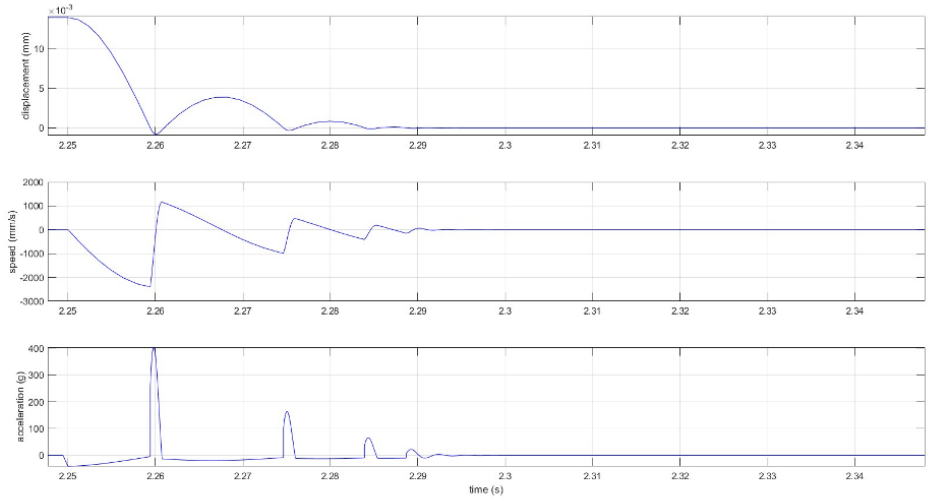

Figure 3.8: Plots of vertical position, velocity and acceleration for the rod in Matlab 2023a, detail for better comparison with previous model

In order to be able to control the trigger position to bring the rod in the desired position an actuator must be used. In this case two different solutions where found and their performances where evaluated in simulation.

3.2 Solenoidal actuator

The first solution shown for linear actuation is a 24 V solenoid with a spring return. The solenoid is modelled using magnetic blocks (figure 3.11). The current through the solenoid produces a magnetomotive force (mmf) which drives a flux through the magnetic core of the solenoid. A reluctance force is produced which drives the plunger to close the air gap, initially 15mm in length. The flux in the magnetic core increases as the air gap length decreases. When unpowered, the spring pulls the plunger to the resting position.

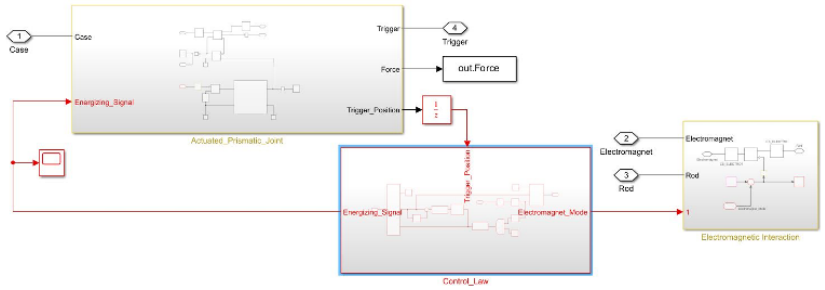

Figure 3.9: Simulink subsystem realizing the interaction between trigger and electromagnet when the solenoid is used, inside view

In order to allow the electrical system of the solenoid to interact with the trigger joint, a translational multibody interface block has been used (figure 3.10).

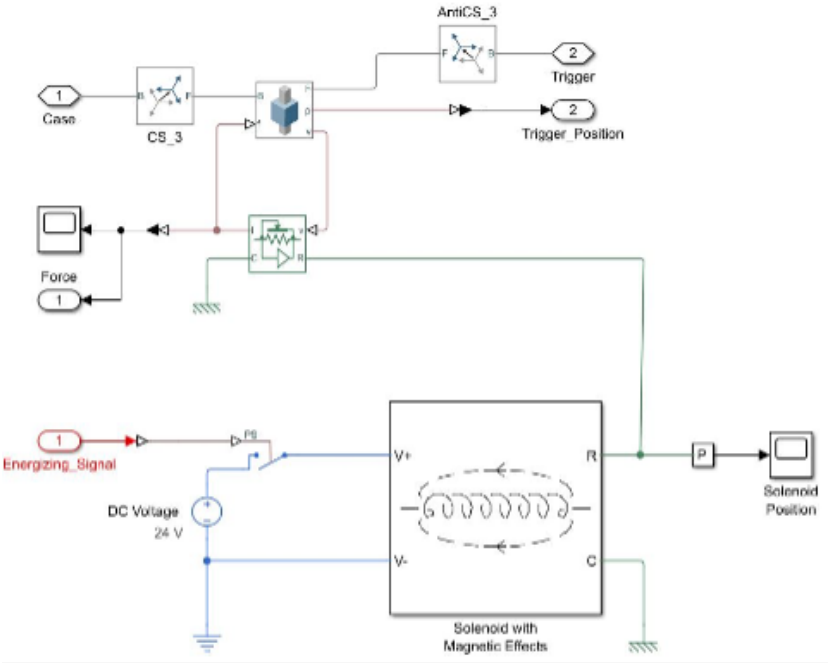

Figure 3.10: Actuated prismatic joint, inside view. The solenoid and its connection to the joint can be appreciated

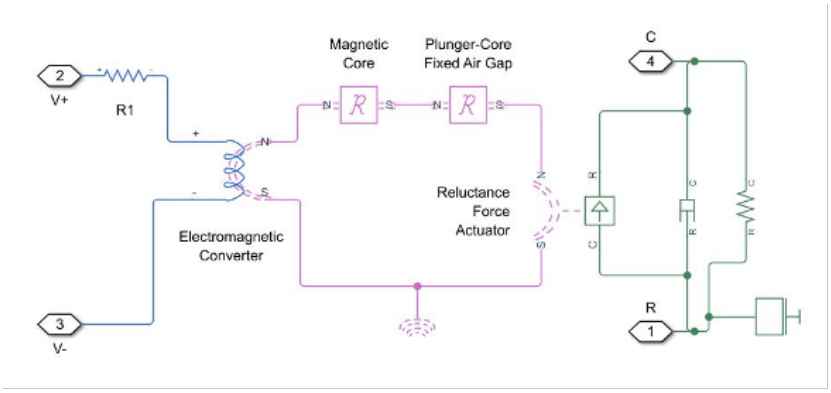

Figure 3.11: Solenoid with magnetic effects, inside view

The mode of the weld joint can assume value equal to 0 when normal and -1 when disengaged, but in practice we desire to control the electromagnet status through a 0-1 signal. As a consequence, in order to control the simulated model as the real one, the subsystem has been realized as shown in figure 3.12:

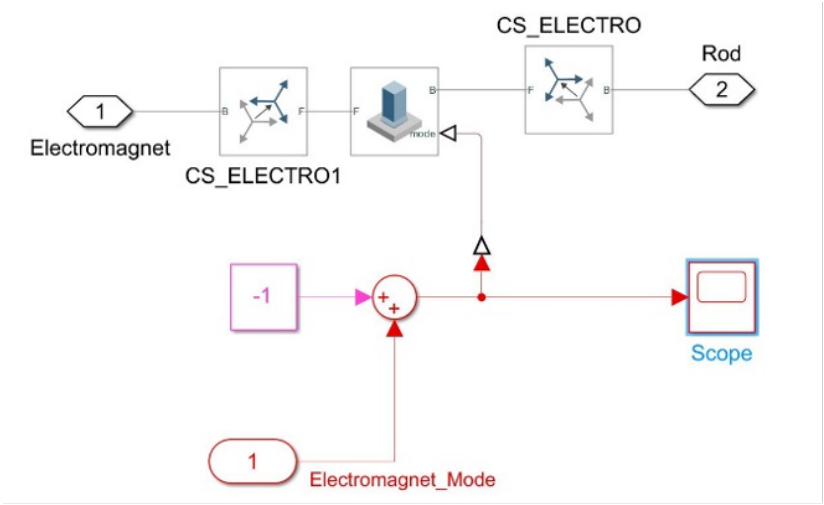

Figure 3.12: Electromagnetic interaction subsystem, inside view

In order to control the system and execute a defined number of consecutive surveys, a control law has been made such that:

1. The electromagnet turns on;
2. The solenoid is energized with a delay of 0.5 seconds;
3. Once the rod is in the desired position of 14 mm, after a delay of 0.75 seconds the electromagnet turns off and the rod falls against the material;
4. After a delay of 0.5 seconds the solenoid turns off and the electromagnet returns in contact with the rod;
5. After 0.75 seconds the electromagnet turns on again and the cycle restarts;
6. After a number of rising edges in the electromagnet mode signal greater than the desired number of the iterations, the energizing signal is put to zero indefinitely;
7. After a 0.5 seconds delay the electromagnet turns off.

A sample time of 1 ms has been used. The block scheme of this control law is shown in figure 3.13.

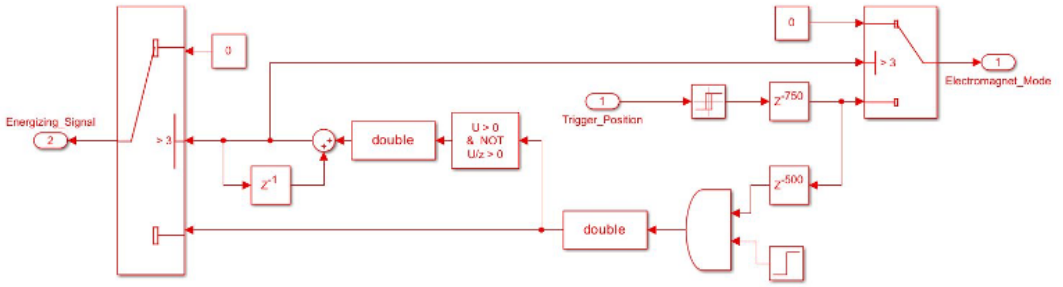

Figure 3.13: Control law subsystem for the solenoid actuator and the electromagnet, inside view

As an example, a simulation has been executed with desired number of consecutive measurements equal to three.

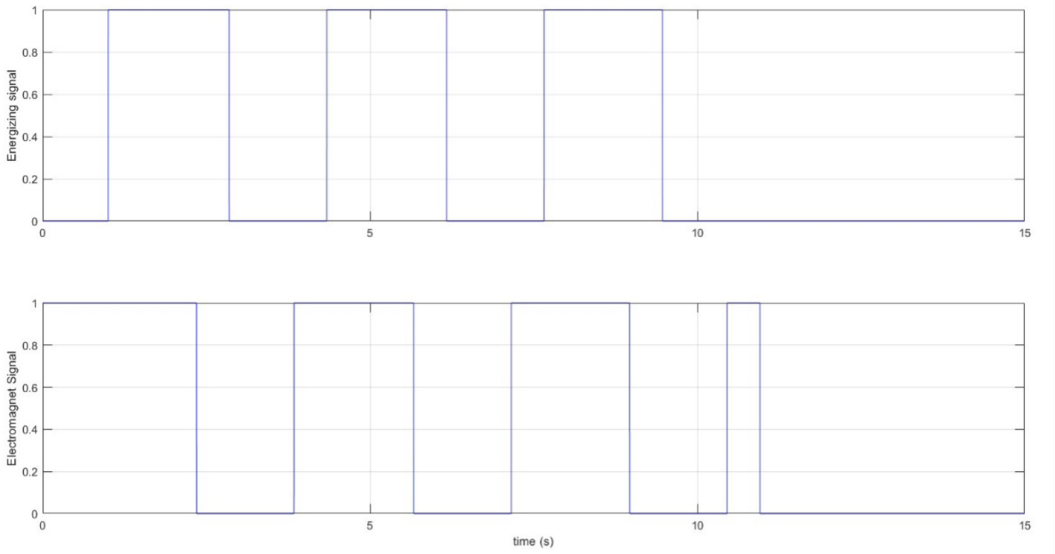

Figure 3.14: Plots for the solenoid energizing signal and the electromagnet state signal

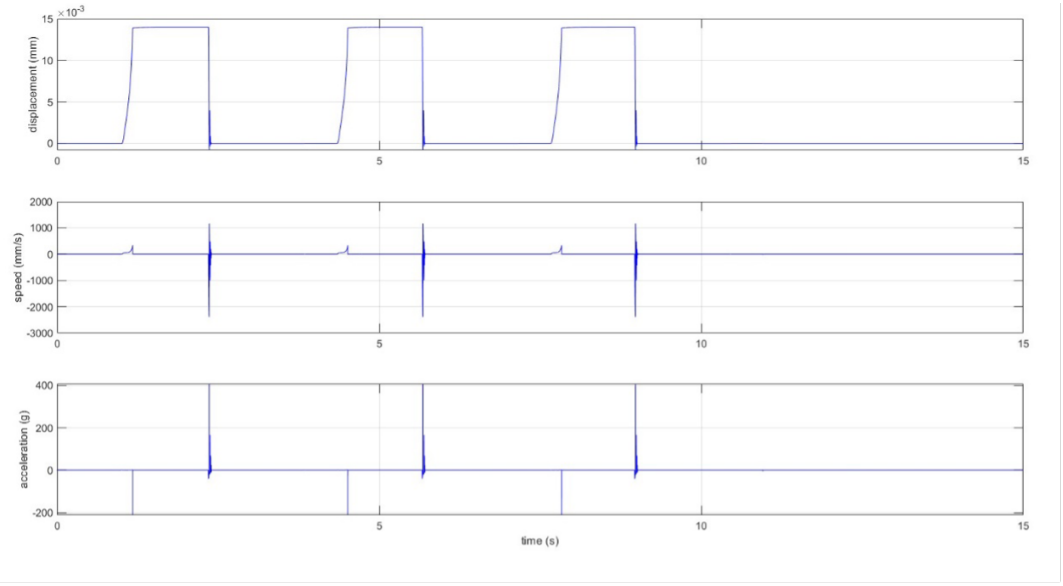

Figure 3.15: Plots of vertical position, velocity and acceleration for the rod using the solenoidal actuator

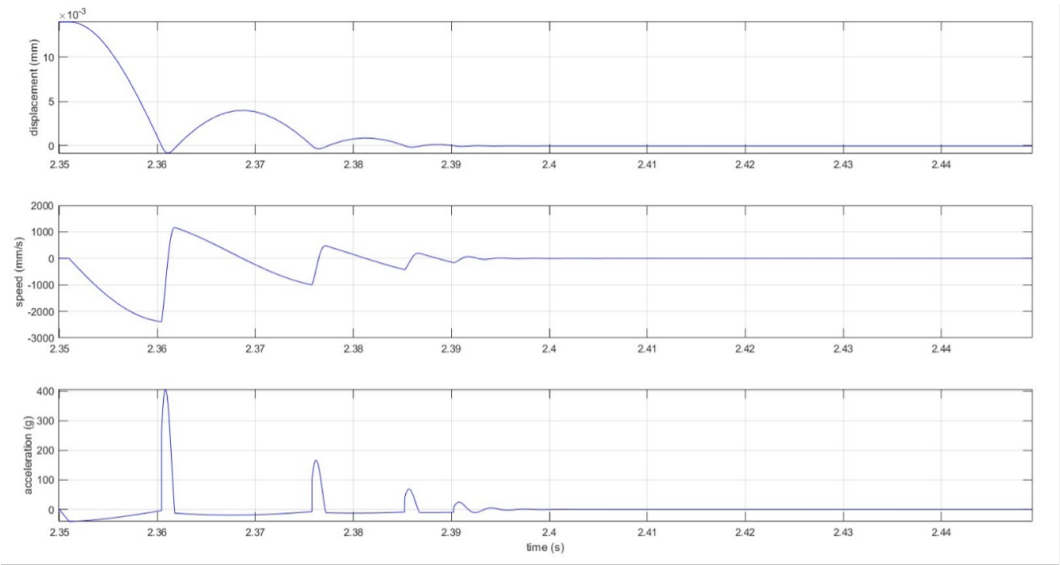

Figure 3.16: Plots of vertical position, velocity and acceleration in time for the rod using the solenoidal actuator, detail

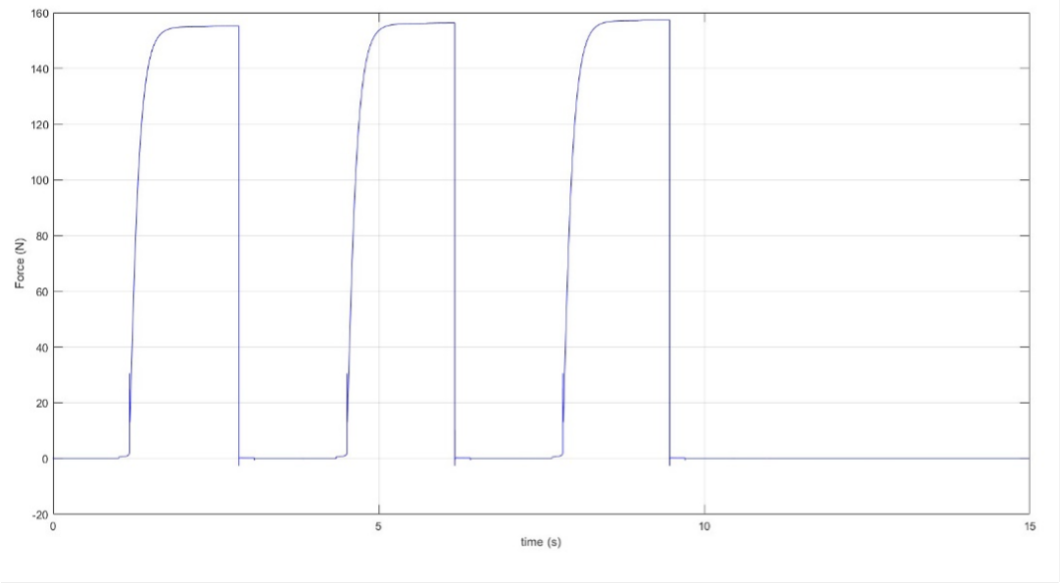

Figure 3.17: Plots of the force actuating the trigger's prismatic joint using the solenoidal actuator

3.3 DC linear actuator

The second solution shown for linear actuation is a 24 V DC linear actuator consisting of a DC motor driving a worm gear, which in turn drives a lead screw to produce linear motion. Similarly to the solenoid case, in order to allow the electro-mechanical system of the actuator to interact with the trigger joint, a translational multibody interface block has been used.

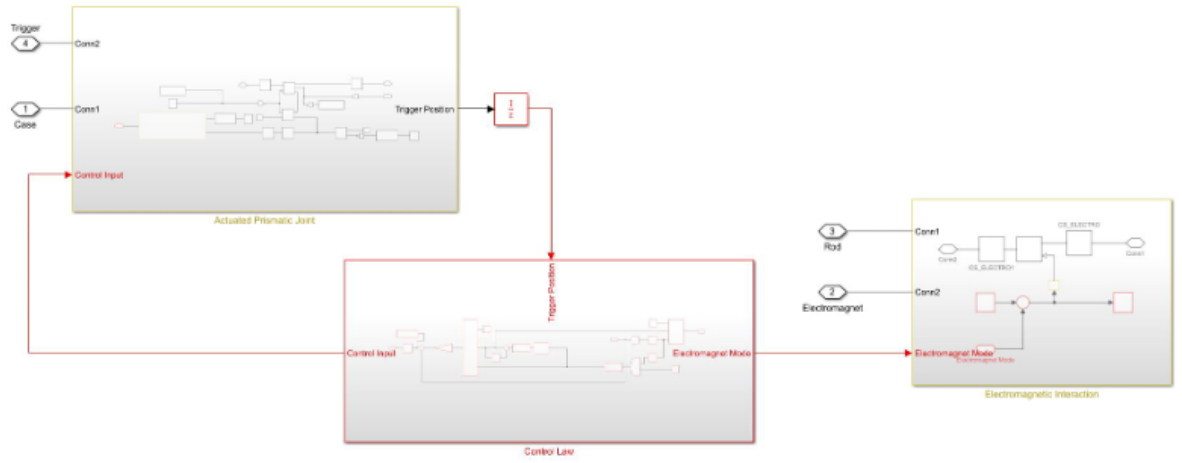

Figure 3.18: Simulink subsystem realizing the interaction between trigger and electromagnet when the DC linear actuator is used, inside view

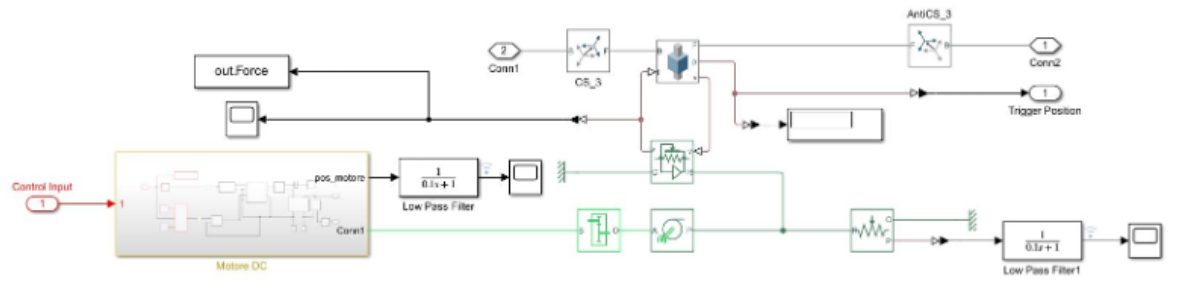

Figure 3.19: Actuated prismatic joint, inside view. The interaction between the DC linear actuator and the prismatic joint can be appreciated

The control law, shown in figure 3.20, is extremely similar to the one used for the solenoidal actuator, but in addition what was previously the energizing signal is now multiplied by the desired height in order to obtain the trigger position reference. This is compared with the effective trigger position and an error is computed. Through a PID action we are able to compute the desired duty cycle. The latter is given as input to a controlled PWM voltage block which drives the H-bridge motor drive, which in turn drives the DC motor (figure 3.21).

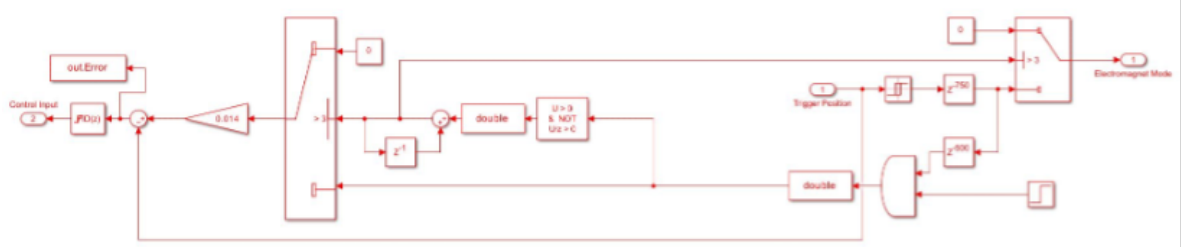

Figure 3.20: Control law subsystem for the DC linear actuator and the electromagnet, inside view

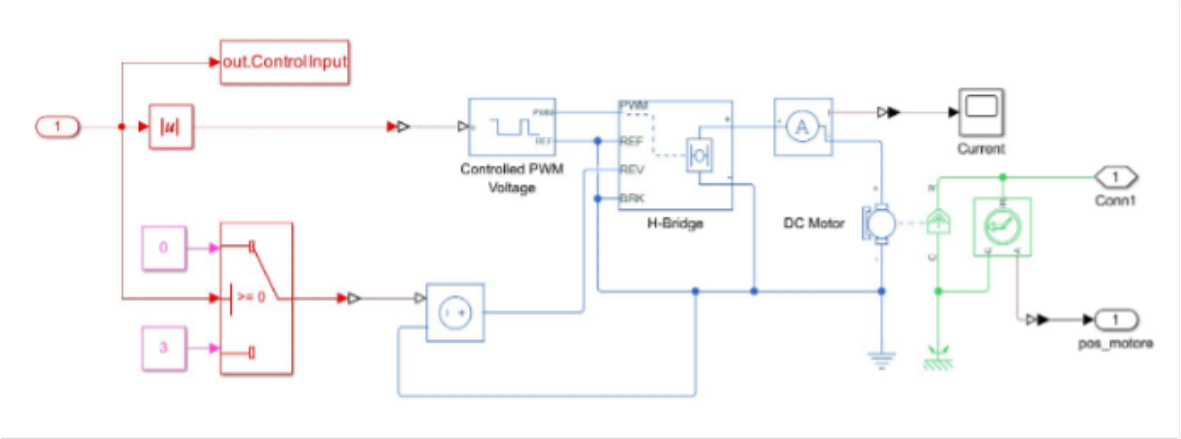

Figure 3.21: Controlled DC motor block scheme

The PID has been manually tuned and the final chosen gains are P=50,I=100,D=7. As an example, a simulation has been executed with desired number of consecutive measurements equal to three

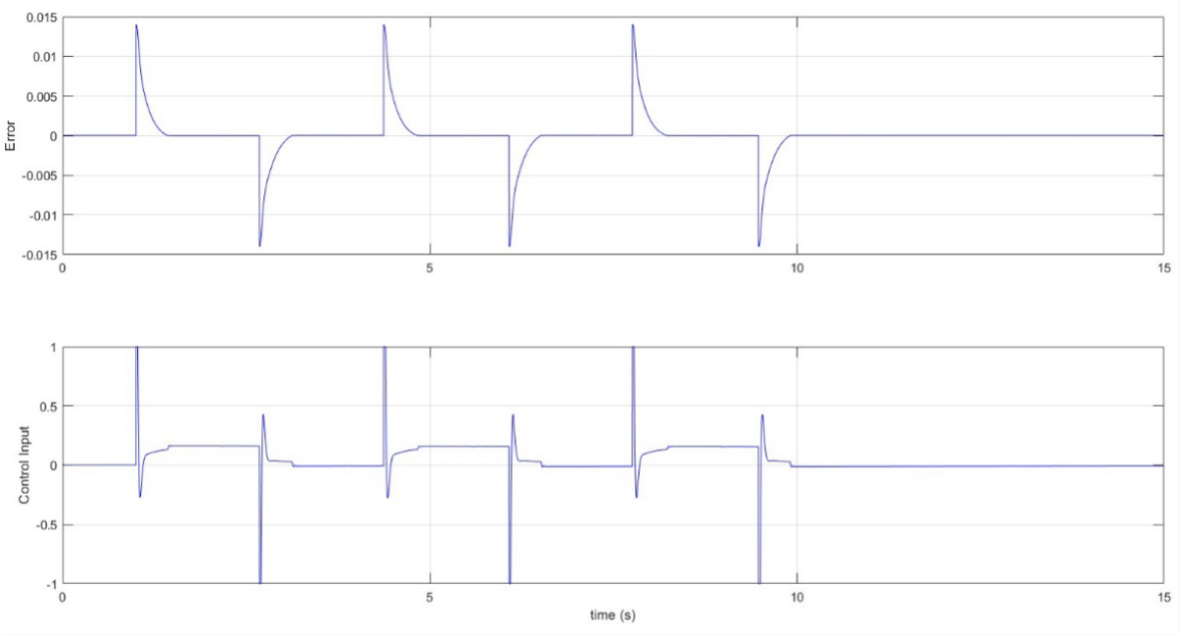

Figure 3.22: Error for the trigger position and control input computed by the PID action

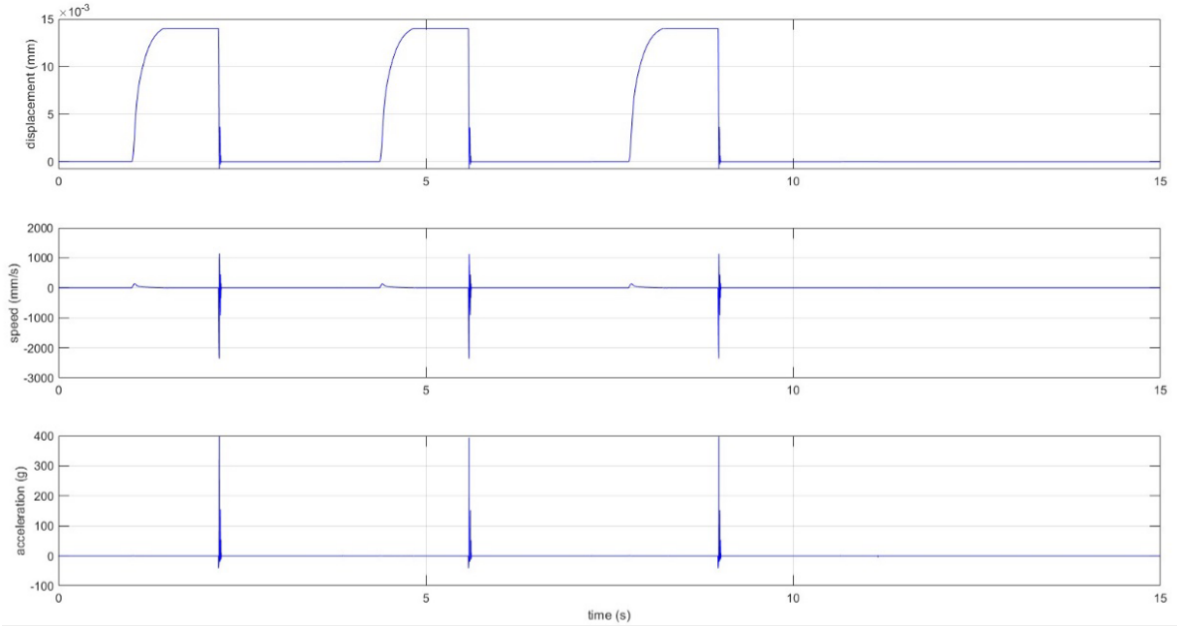

Figure 3.23: Plots of vertical position, velocity and acceleration for the rod using the DC linear actuator

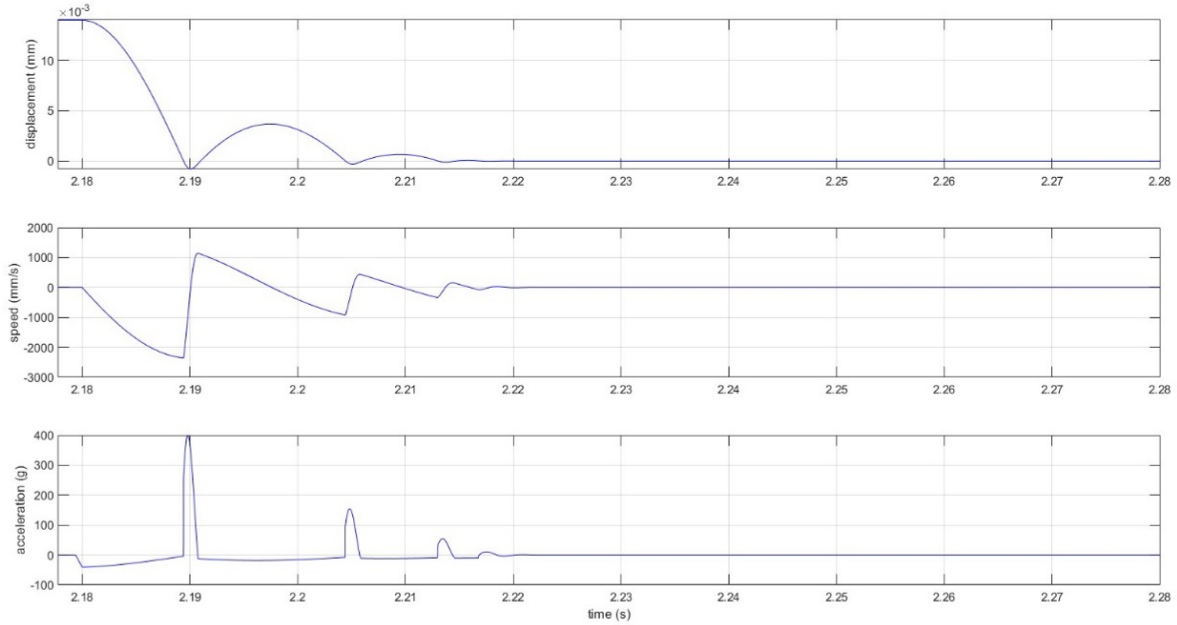

Figure 3.24: Plots of vertical position, velocity and acceleration in time for the rod using the DC linear actuator, detail

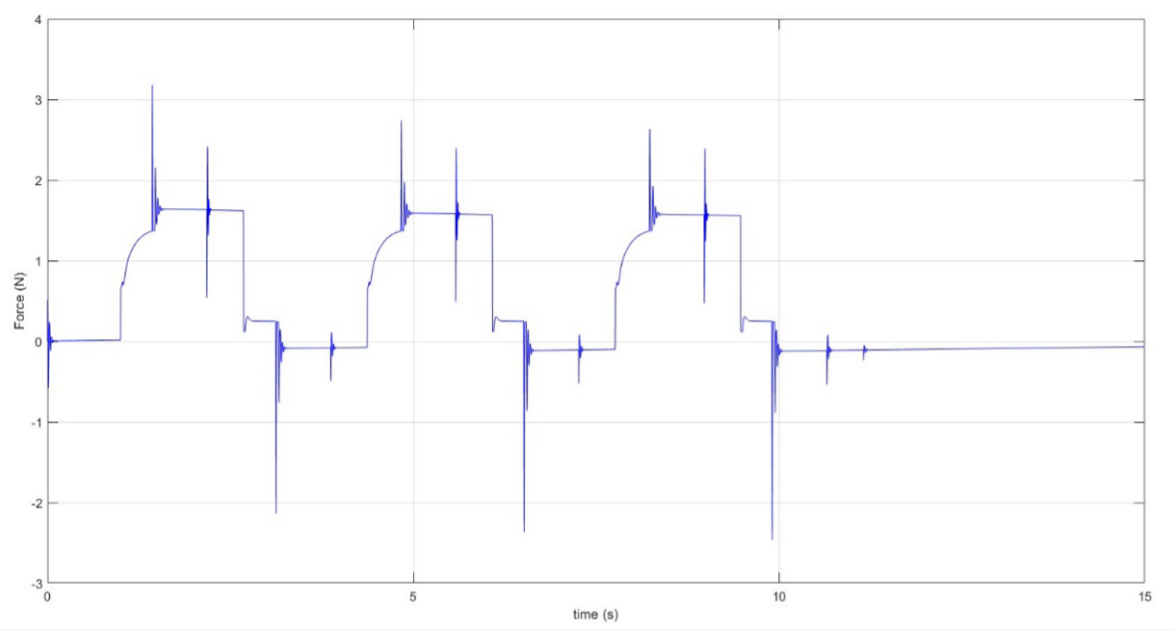

Figure 3.25: Plots of the force actuating the trigger's prismatic joint using the DC linear actuator

3.4 Software in the loop

After verifying that both the actuation solutions are feasible and able to guarantee the desired behaviour for the system, the next step in the development of the controller for the physical system is the Software in the loop stage. The modelled controller is replaced with the code in order to verify its ability to control the simulated plant. Through the Matlab-Simulink environment functionalities the team has been able to automatically generate the code able to control the system for both the modelled actuator. Then simulations have been executed in order to verify that the code is able to control the system as desired. Since no noticeable difference has been detected between the modelled controller and the generated code performances, we'll focus only on the plots regarding the outputs of the SIL blocks for both the actuators (figure 3.26 and 3.27).

3.4.1 Solenoidal Actuator SIL

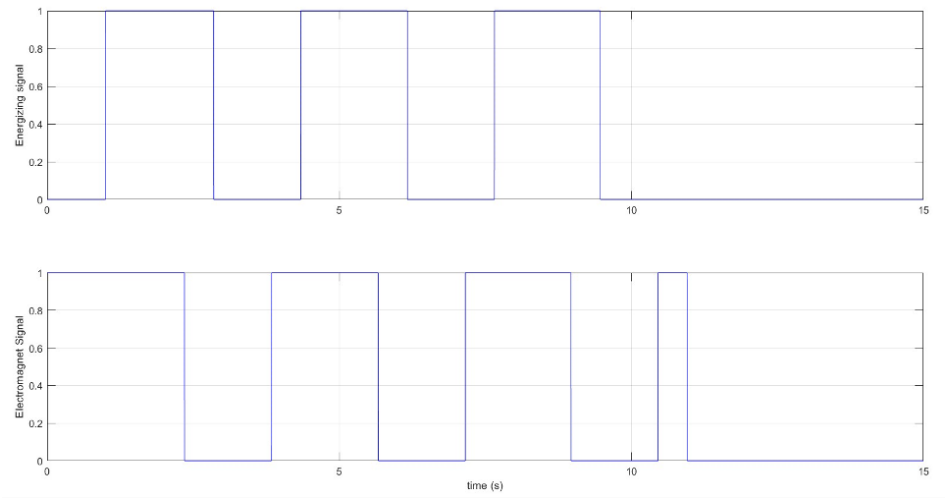

Figure 3.26: Plots for the solenoid energizing signal and the electromagnet state signal when the SIL control block is used

3.4.2 DC linear actuator SIL

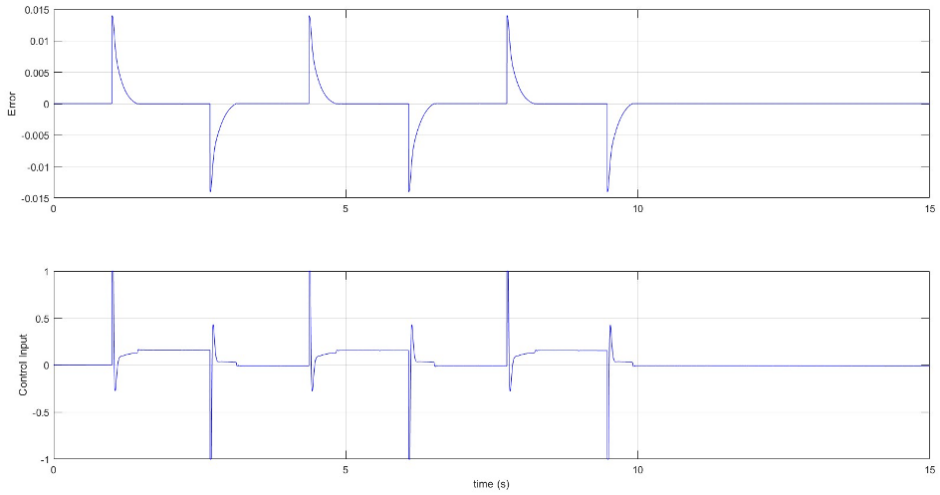

Figure 3.27: Error for the trigger position and control input computed by the PID action when the SIL control block is used

3.5 Processor in the loop

After assessing the code to be correctly controlling the system, the next step of development required to verify that the hardware designed to run the controller onto the real plant is able to correctly execute the code and control the system. This is done by executing Processor in the loop simulations in which the controller code runs onto the chosen board while the latter interacts with the simulated plant. In this case the chosen board is an STM32 Nucleo F401RE (figure 3.28).

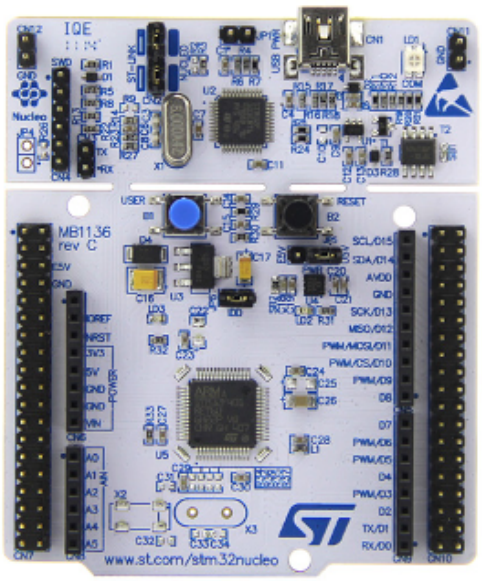

Figure 3.28: STM32 Nucleo F401RE

In order to execute a simulation introducing a PIL block in Simulink it is necessary to adopt a fixed-step solver. In the executed tests, the solver chosen has been ode14x (extrapolation) with an extrapolation degree of 4 for better results and a fixed-step size of 1 ms. Once a fixed-step solver is used the data regarding the simulated mechanical system start to differ from the real one due to Simulink interpolation. In order to verify that the controller is working correctly, we'll focus on the outputs of the controllers only. By comparing them with the ones previously obtained in MIL and SIL stages no noticeable differences are detected.

3.5.1 Solenoidal Actuator PIL

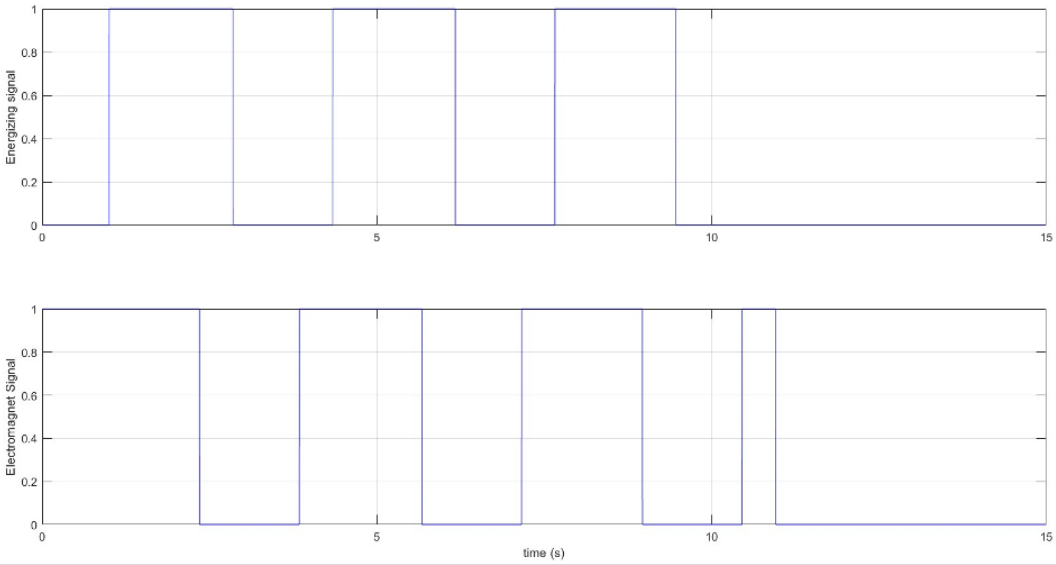

Figure 3.29: Plots for the solenoid energizing signal and the electromagnet state signal when the PIL control block is used

3.5.2 DC linear actuator PIL

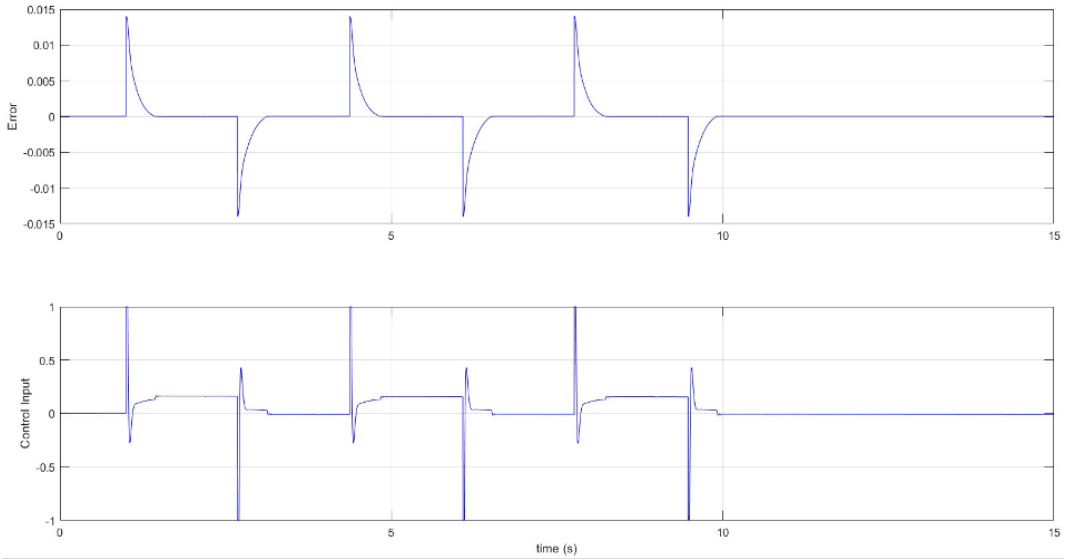

Figure 3.30: Error for the trigger position and control input computed by the PID action when the PIL control block is used

Chapter 4

Hardware in the loop

Hardware-in-the-loop (HIL) is a simulation technique used primarily in the development and testing of complex real-time systems. It involves connecting physical hardware components, like controllers or devices, to a simulation environment to validate their behavior under various conditions. An Hardware in the loop simulation, tipycally contains:

- Real-Time Simulation Environment: A computer-based simulation system is set up to replicate the behavior of a larger system. This simulation environment typically includes models that mimic the behavior of the actual system components.
- Physical Hardware Connection: The real hardware components, such as controllers, sensors, actuators, or any other devices that interact with the system, are connected to the simulation environment.
- Feedback Loop: The simulation environment interacts with the hardware in real-time, providing inputs and receiving outputs. The hardware responds to the simulation just as it would in the real world.
- Testing and Validation: Engineers can test and validate the performance of the hardware under various simulated scenarios. They can evaluate how the actual hardware behaves in response to different conditions, disturbances, or faults that might be difficult or risky to replicate in a purely physical environment.

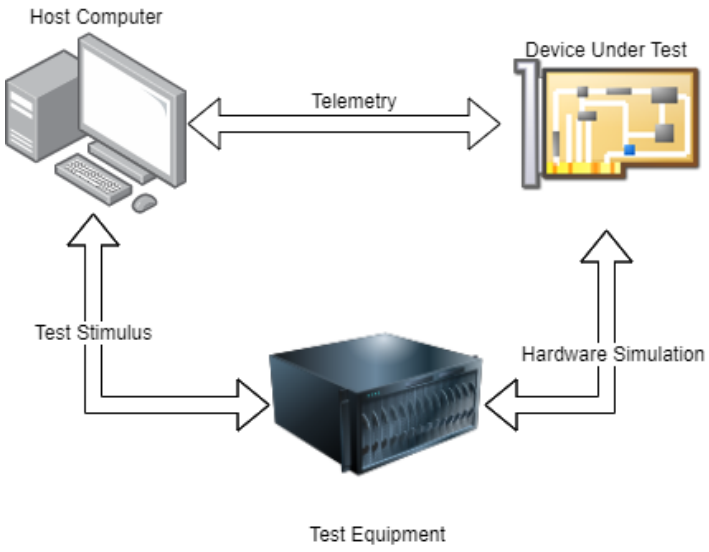

Figure 4.1: Hardware in the loop scheme

4.1 Controls of actuators and involved electronics

Linear DC actuators are devices that convert rotary motion into linear motion. They're commonly used in various applications like robotics, automation, automotive, and more. These actuators consist of a DC motor combined with a lead screw, worm gear, or other mechanisms to convert the motor's rotational movement into linear motion.

The heart of the linear actuator is a DC motor. When electricity flows through the motor, it generates rotational movement. This rotational movement of the motor is then translated into linear motion using mechanisms like lead screws or gears. This converts the spinning motion of the motor shaft into a back-and-forth linear movement of a rod or shaft.

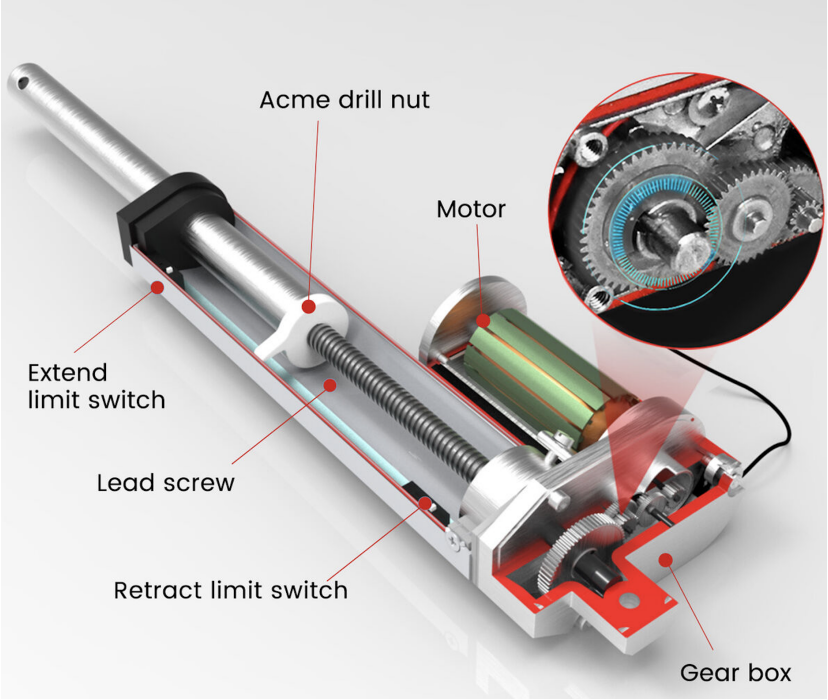

Figure 4.2: Linear DC Actuator

A problem which has been faced regards the fact that embedded boards like Arduino might not provide sufficient current to directly power and control high-powered DC motors used in larger linear actuators. For this purpose a Motor shield or H-Bridge controllers (like the L298N component) act as interfaces between the low-power output of embedded boards and high-power DC motors.
The L298 is a dual H-Bridge motor driver IC that allows bidirectional control of two DC motors or a single stepper motor. It can control the direction and speed of the motor by controlling the voltage and polarity of the applied power.

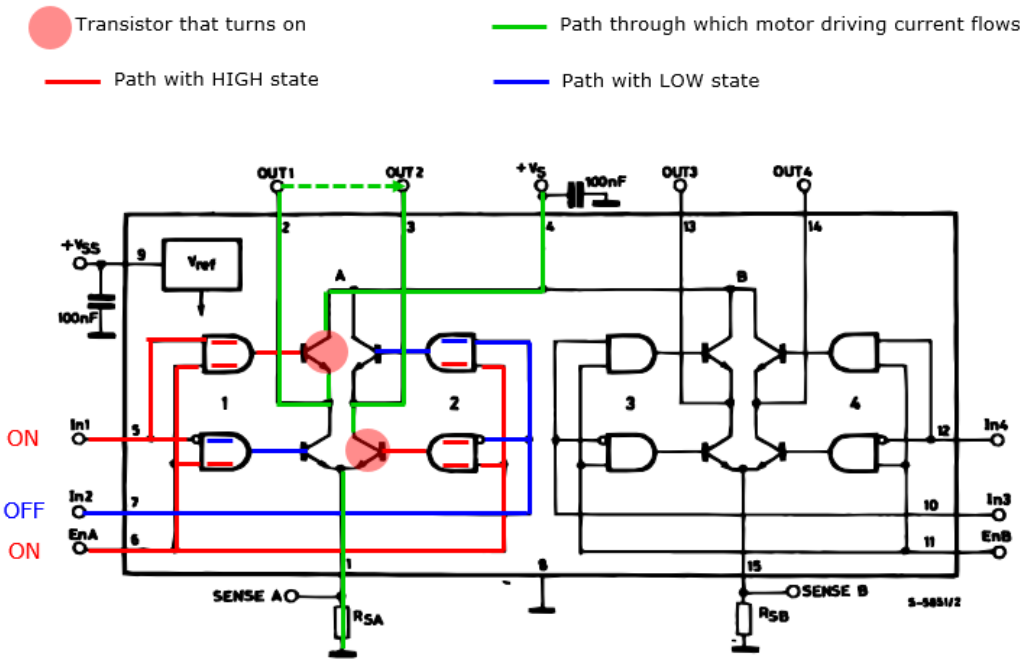

Figure 4.3: L298 IC Scheme and principle of working

By using motor shields or H-Bridge controllers like the L298N, embedded boards with limited current output can effectively control larger linear DC actuators, regulating their speed and direction while ensuring the necessary power is delivered to the motors. The L298N is connected to the DC motor of the linear actuator. It receives commands (control signals) from the embedded board (like an Arduino) through digital pins and an external power source of 12V In order to control the speed of the motor connected to the shield board, PWM signals from the Arduino can be used, where by adjusting the duty cycle of the PWM signal, it's possible to control the average voltage applied to the motor, hence controlling its speed.

4.2 Arduino development board

Arduino boards are versatile microcontroller platforms which comes with its programming IDE (Integrated Development Environment), C/C++. An Arduino board typically comprises a microcontroller chip, input/output pins, voltage regulators, and a USB interface for programming and power. The heart of an Arduino is its microcontroller, which executes the code uploaded via the IDE.

Algoritmo_Motorio_VesEvo | Arduino 1.8.19

Algoritmo_Motorio_VesEvo §

```
/*
 ********* VESEVO ACTUATING SYSTEM ************
 *
 * A project of: Salvatore Granata, Francesco De Maio, Carlo Cianflone, Davide Rubinacci
 * Professor: Eng. Raffaele Iervolino
 * University of Naples Federico II
 *
 */
#include <Adafruit_MotorShield.h> //Libreria

#define pinPulsante 7
#define pinLed 2
#define pinInferiore 3

int previousState = HIGH;   //Stato precedente del pin
int buttonWasPressed=false; //Flag per tenere traccia dello stato di pressione del pulsante

int avvio=0;
// Create the motor shield object with the default I2C address
Adafruit_MotorShield AFMS = Adafruit_MotorShield();

// Select which 'port' M1, M2, M3 or M4. In this case, M1
Adafruit_DCMotor *myMotor = AFMS.getMotor(1);

void setup(){
 pinMode(pinPulsante,INPUT);
 pinMode(pinLed,OUTPUT);
 pinMode(pinInferiore,INPUT);
 Serial.begin(9600);
 Serial.println("Avvio Programma");

  if (!AFMS.begin()) {         // create with the default frequency 1.6KHz
  // if (!AFMS.begin(1000)) {  // OR with a different frequency, say 1KHz
    Serial.println("Could not find Motor Shield. Check wiring.");
    while (1);
  }
  Serial.println("Motor Shield found.");

  // Set the speed to start, from 0 (off) to 255 (max speed)
  myMotor->setSpeed(150);
  myMotor->run(FORWARD);
  // turn on motor
  myMotor->run(RELEASE);
  Serial.println("Programma pronto per esecuzione");
}

```

Figure 4.4: Arduino IDE

In this projects, Arduino boards serve as the central processing unit, interfacing with various sensors, actuators, and shields. Shields are add-on boards that expand the capabilities of the Arduino. They might offer functionalities like wireless communication (Wi-Fi, Bluetooth), motor control, GPS, or additional input/output ports. Connecting shields to Arduino boards is straightforward: they stack on top of the Arduino, utilizing the standardized pin layout and connectors. This plug-and-play design simplifies the process, allowing users to easily add or remove shields based on project requirements. Each shield usually interfaces with the Arduino through specific pins or communication protocols, expanding the board's capabilities without complex wiring or soldering. In our case, in order to control the motion of the motor, it's been useful to use an Arduino Motor Shield, made by Adafruit, which is directly mounted on arduino and offers a library with some APIs that allows to control motor loads in terms of both direction and speed (by means of PWM).

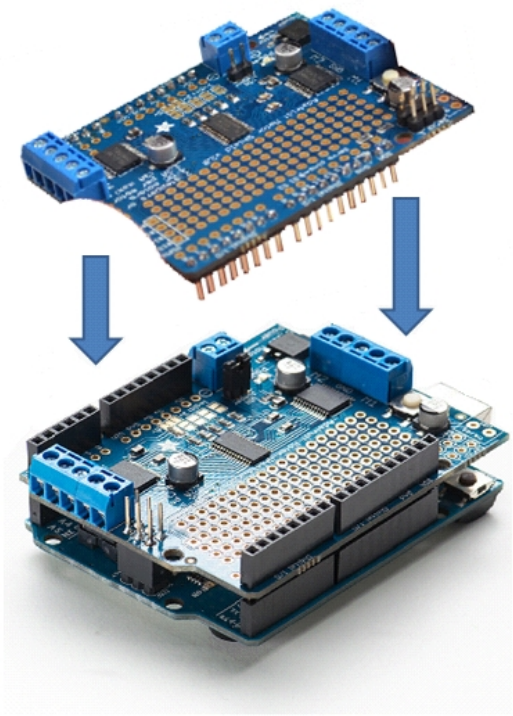

Figure 4.5: Motor Shield Connection

4.3 Testing and validation

Once the shield was mounted on the main board, the testing phase began, where the circuit that would be mounted on the Vesevo system was first created, simulated separately, and then validated by testing it on the original system.

The connection scheme of the circuit to be used for controlling and actuating the Vesevo measurement system is as follows:

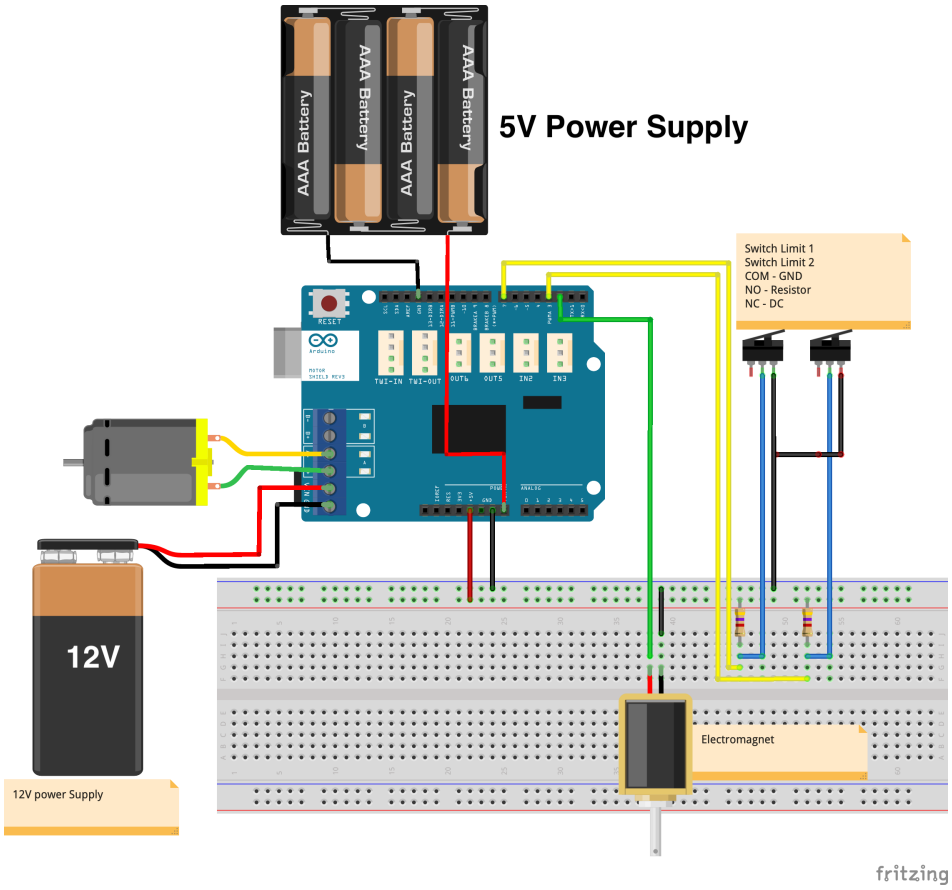

Figure 4.6: Connection Scheme of the Actuation System

It's important to highlight the addition of signal debouncing resistors, useful to make the system more robust and ensure correct measurement of the switch state in the presence of noise. Once the connections shown in the diagram were made, we moved to the programming phase, where the logic implemented in the program is as follows:

1. Initialization of the hardware peripherals used (initiating communication with the shield and serial monitor).

2. Waiting for the measurement start signal (simulated in the code by sending the number 1 but replicable through buttons or sensors).

3. Starting the motor in the rising phase to bring the bouncing mass to the correct height.

4. When the trigger enables the upper microswitch, the linear actuator stops, and as a result:

   - 1 second is waited to stabilize the mass after the ascent.
   - The mass is released by de-energizing the electromagnet.
   - The measurement phase is indicated by turning on an LED.

5. Once the measurement is taken (thus estimated to be completed after a temporal period), the linear actuator lowers, stopping when the lower microswitch is reached.

```
void loop(){
 avvio=Serial.read();
 if (avvio == '1') {
  bool finito=false;
  Serial.println("AVVIO PROGRAMMA");
  while(!finito){
     myMotor->run(FORWARD);
     myMotor->setSpeed(100);
     delay(10);
     int currentState=digitalRead(pinPulsante);   //Leggo lo stato attuale del pin
     //Verifica se c'è stato un fronte di salita e se il pulsante era stato precedentemente lasciato
     if(currentState==HIGH && previousState==LOW && !buttonWasPressed){
        Serial.println("Fronte di Salita rilevato (premuto)");
        //Esegui qui le azioni desiderate quando viene rilevato il fronte di salita (premuto)
        myMotor->setSpeed(0);
        digitalWrite(pinLed,HIGH);
        Serial.println("Motore fermo, attendo 1 secondo");
        delay(1000);
        Serial.println("Elettromagnete disalimentato");
        digitalWrite(pinLed,LOW);
        buttonWasPressed=true;  //Imposto il flag a true quando il pulsante è stato premuto
        bool sotto=false;
        while(!sotto)
        {
           myMotor->setSpeed(100);
           myMotor->run(BACKWARD);
           if(digitalRead(pinInferiore)==LOW){
              Serial.println("Switch Inferiore Raggiunto");
              sotto=true;
           }
        }
        myMotor->setSpeed(0);
        myMotor->run(RELEASE);
        Serial.println("Campioni acquisiti");
        finito=true;
     }
      else Serial.println("Non è successo nulla");
     //Verifico che il pulsante sia stato rilasciato
     if(currentState==LOW && previousState==HIGH){
      buttonWasPressed = false; //Resetta il flag quando il pulsante viene rilasciato
     }

     //Aggiorna lo stato precedente con lo stato attuale
     previousState=currentState;

   }
  }
}
```

Figure 4.7: C++ Implementation of the actuation steps

The advantage of this automated actuation system is easily highlighted in the repeatability of the measurements, as initiating a new measurement only requires repeating step 1, after which the others occur automatically. As evident, human intervention is minimal, favoring this system to be mounted on robots or autonomous systems.

Once the actuation system was built and tested using the HIL methodology, we proceeded with a validation of the result by testing the control hardware on the VesEvo system.

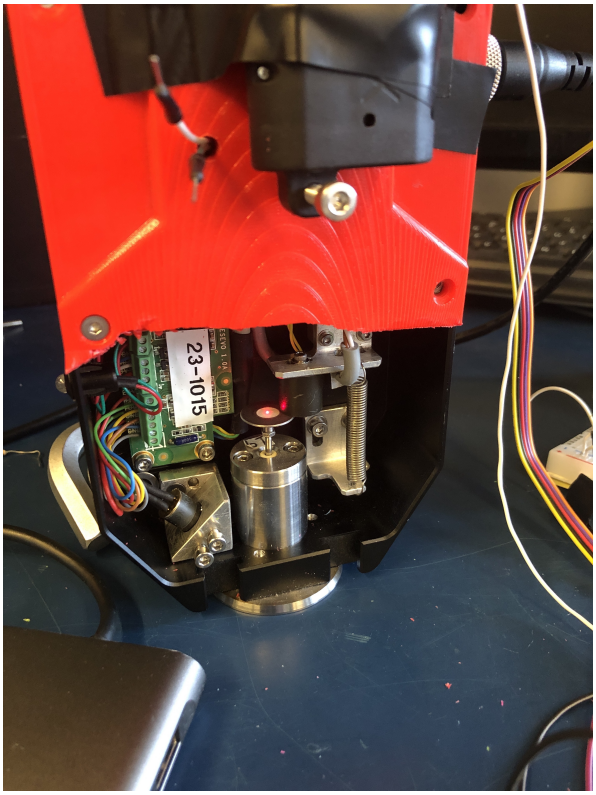

Figure 4.8: Vesevo Magnet and laser system to measure mass displacements

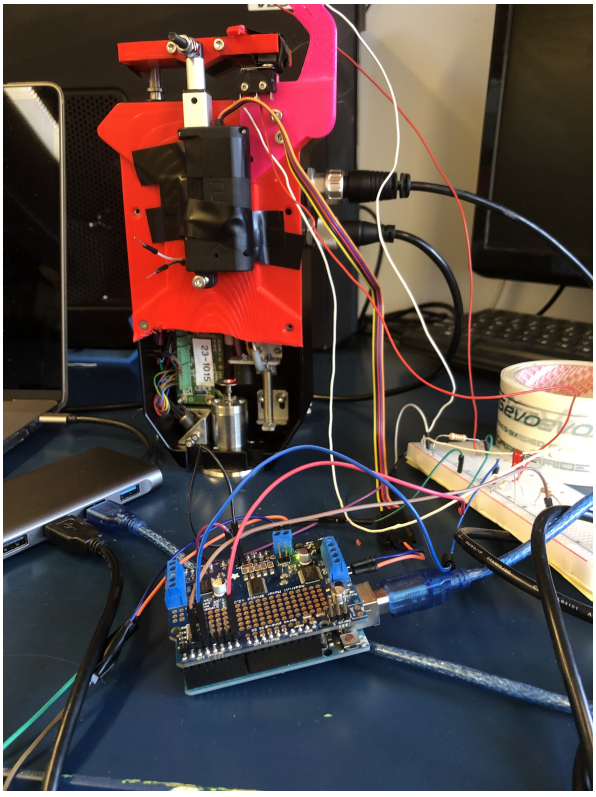

Figure 4.9: Vesevo system actuation commanded by Arduino (HIL)

4.3.1 Sensor status reading and magnet command

One of the problem we faced was the fact that the system was "closed" from outside and there were no chance to program it directly.
Without losing faith, in order to read the operational status of the board, control the magnets, and access the sensor without direct access to the custom microcontroller embedded within the system, it was necessary to bypass it by using the USB serial communication. However, given that Arduino supports UART serial communication rather than USB serial, a Man-in-the-Middle system was devised using a Python script to establish communication between the two microcontrollers. The overall impression at the conclusion was that Arduino was commanding the Vesevo system for measurement purposes.

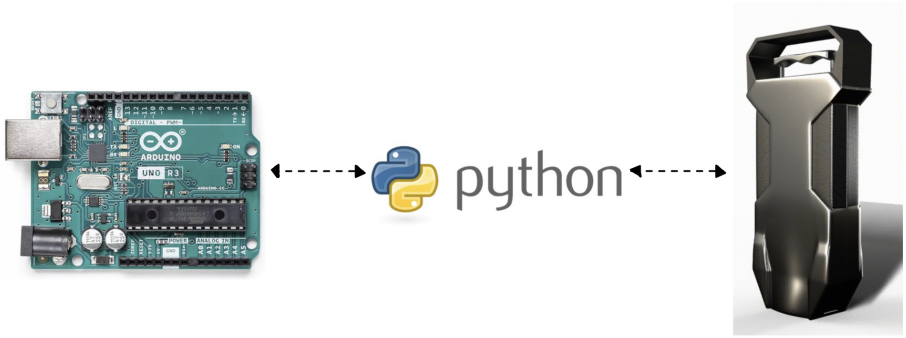

Figure 4.10: man-in-the-middle logic

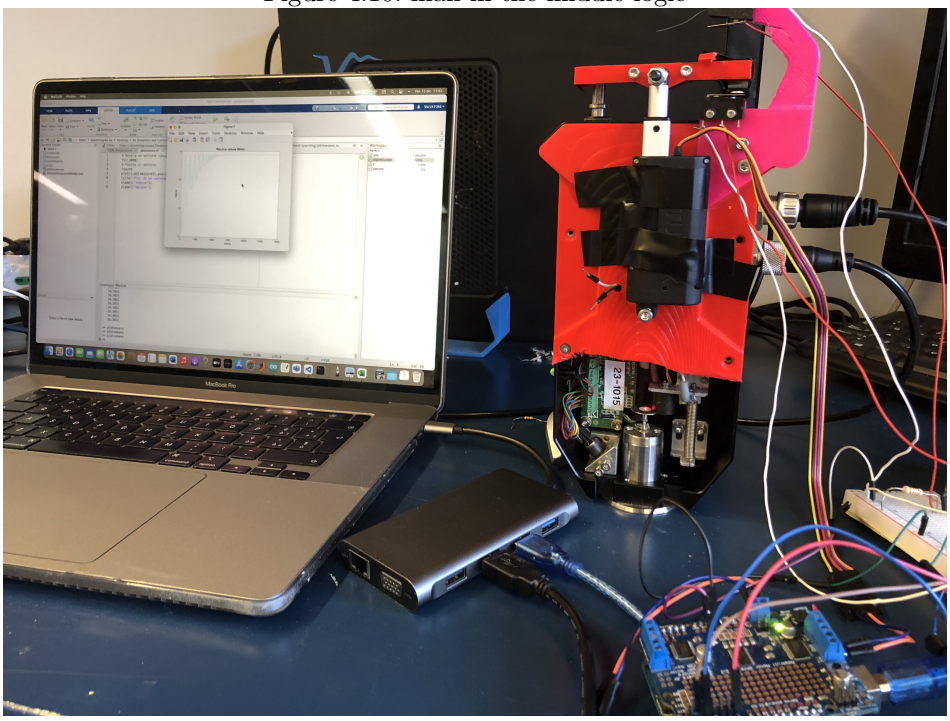

Figure 4.11: Real representation of the man-in-the-middle

This was achieved by employing a computer running a python script, which received and exchanged data with both devices, initiating the electromagnetic coil excitation procedure

precisely when the trigger was activated. At the end stop, the Python script instructed the Vesevo system to check the trigger status, and upon a positive result, to de-energize the magnet, consequently releasing the bouncing mass while simultaneously acquiring a specified number of samples.

Users > salvatoregranata > Desktop > import serial.py > ...

```
        # Verifica se la risposta contiene "OK" e termina con "#"
    if "<release>" in response:
        time.sleep(10)  # Estende il tempo di acquisizione a 2 secondi per raccogliere i dati

        # Legge e salva tutte le righe disponibili
        release_responses = []
        while ser.in_waiting:
            line = ser.readline().decode().strip()
            if line:
                release_responses.append(line)

        print("Risposta al comando release*3000:")
        print('\n'.join(release_responses))  # Mostra tutte le righe ricevute

        # Salva tutte le righe nel file CSV sul desktop
        with open(csv_file_path, mode='a') as file:
            for release_response in release_responses:
                file.write(release_response + '\n')

        # Invio del numero 3 ad Arduino
        arduino.write(b'3\n')
        print("Inviato '3' ad Arduino")
        misuraVesevo = True
    else:
        print("Non inviato '3' ad Arduino")

def save_to_csv(data):
    # Salva i dati nel file CSV
    with open('dati_misurati.csv', mode='a') as file:
        csv_writer = csv.writer(file)
        csv_writer.writerow(data)

while exit==False:
    # Input iniziale dall'utente
    misuraVesevo=False
    start_command = input("Digita 'avvio' per iniziare: ").strip()

    if start_command.lower() == 'avvio':
        #abilito l'acquisitionmode
        ser.write(('acquisitionmode\r\n').encode('utf-8'))
        ser.write(('acquisitionmode\r\n').encode('utf-8'))
        # Invio del numero 1 a Arduino
        arduino.write(b'1\n')
        time.sleep(0.1)
        arduino.write(b'1\n')
        read_response()  # Legge la risposta dalla seriale principale


        # Chiede all'utente di inserire il comando da inviare
        while misuraVesevo==False:
```

Figure 4.12: Part of the Python Implementation code of the man-in-the-middle system

The most interesting thing in this is that all these steps were previously executed 'manually' by sending commands to the vesevo custom board from a computer, while with this system, they are entirely automated. Upon completion of the measurement, a .csv file containing all the measurement data will be generated. To conclude, upon receiving a positive signal from the Python program, Arduino will command the motor to descend to the lower end stop, thereby concluding the measurement cycle. These measurements saved in a CSV file are then processed by a MATLAB script responsible for plotting the rebounds.

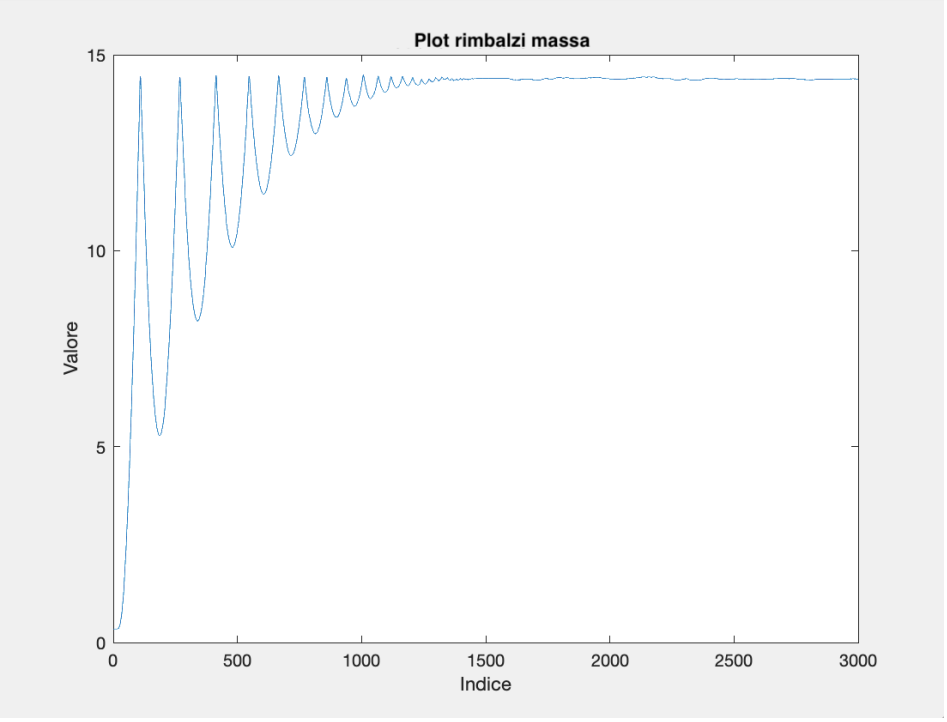

Figure 4.13: Output of the laser distance sensor (bouncing results)

It is noteworthy that the plot displayed is inverted concerning that of Simulink because here, the reference is no longer the mass but rather the zero point of the sensor. The resulting outcome was excellent, as the system responded well to commands and integrated well with the measurement system.

Chapter 5

Conclusions and future developments

As a conclusion chapter of the report conducted for Vesevo, it's crucial to highlight the satisfaction with the work accomplished and the achieved outcomes. The essence lies in the simplification of modeling, simulation, and control operations concerning the measurement device.

As future developments, a significant step involves the abolition of the spring system, favoring a design entirely driven by an electric motor. This transition not only saves space but also necessitates a restructuring of the measurement device's setup. This anticipated change stands as a testament to the adaptability and scalability of the current system, showcasing its potential for evolution and improvement.

Moreover, an intriguing aspect of future development involves the integration of the Vesevo system onto a robotic arm. This direction has prompted a focus on motor-driven solutions for triggering and moving the mass from the plate. The prospect of adapting the Vesevo system to a robotic arm presents exciting possibilities, expanding its applications beyond conventional static setups. The ability to witness and contribute to this transition highlights the versatility of the system and its potential to evolve into diverse operational environments.

Overall, the completion of this phase of the project for Vesevo marks a significant milestone we've been glad to set, affirming not only the successful achievements but also laying the groundwork for future advancements and expansions of the system's capabilities in diverse contexts.