POLITECNICO DI TORINO
Master of science in Mechatronic Engineering

Master Degree Thesis

Feasibility Study and Development of an Anti-Pinch Application with Machine Learning

Supervisor

Prof. Massimo Violante

Candidate

Giovanni Santeramo

Academic year 2018-2019

Abstract

An anti-pinch is a safety system that senses if a motor is running against an obstacle and prevents any injuries to people or damages to the obstacle or the motor itself. The anti-pinch sensing is usually performed by current monitoring, hall sensors or a combination of both. The state-of-the-art anti-pinch strategy is derivative based. It is simple to implement, and it check continuously the variation of the current and the variation of the velocity. Both are compared with a threshold and if they simultaneously report the presence of an obstacle, the motor is blocked. Although this task may look simple at first sight, many are the factor that add complexity and uncertainty to a standard algorithm: varying system conditions, mechanical and electrical uncertainties, mechanical wear of the motor system, etc. Moreover, the implementation of this algorithm requires a lot of run in order to setup the anti-pinch system correctly (e.g. for setting the thresholds). The scope of this project is to address these issues and to propose an innovative strategy: create an automated test-bench and integrate a machine learning approach to automatically learn what is the best condition to trigger the anti-pinch. The project consists to build a motor bench that simulates a situation where a motor is moving a load and where, during the movement, a pinch can occur. In particular, the test-bench could simulate an automatic back-seat system that works as follow: from an initial position, where the back-seat is parallel to the ground, the motor must move the back-seat until a certain position (expressed in degree). If there is an obstacle during the run, the motor must stop the movement of the back-seat. For the project, it is important to recognize the presence of an obstacle. Avoid false negative is more important than avoid false positive, according to the level of safety. The develop of this project is divided in different steps. Creation of a model of the system using Simulink. Mechanically build of the test-bench. Design of an electronic circuit able to allow

Arduino to control the two motors and to acquire measures from them. Develop the Arduino code in order to control the motors, to collect measures from sensors, and to detect pinch. Design an HMI with Mathworks Matlab software, able to communicate with Arduino board, sending it test parameters and receiving the data Arduino measures. Apply machine learning training algorithm (logistic regression is used in this project) using Matlab. Implement on the Arduino board the machine learning algorithm to detect pinch. Finally, after several test, the machine learning anti-pinch algorithm and the classic anti-pinch algorithm are compared in order to evaluate if the innovative method is more convenient or not than the classic one by analyzing advantages and disadvantages and doing considerations about these methods.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The main purpose of this thesis is to design an innovative algorithm, using machine learning, for implementing the anti-pinch functionality on an automated test-bench and comparing this new approach with the classical one. An anti-pinch is a safety system that senses if a motor is running against an obstacle and prevents any injuries to people or damages to the obstacle or the motor itself. The presence of an obstacle is usually detected by monitoring current behavior, velocity behavior or a combination of both. The state-of-the-art anti-pinch strategy is derivative based. It is an approach easy to implement, and it checks continuously the variation of the current and the variation of the velocity. Both measures are compared with a threshold and if they simultaneously report the presence of an obstacle, the system is blocked, in order to avoid any kind of damages. Although this task may look simple at first sight, many are the factor that add complexity and uncertainty to the standard algorithm. In fact, events due to wearing or uncertainty are very difficult to predict. Also variations on the system conditions can cause error in the pinch detection. Moreover, the implementation of this algorithm requires a lot of runs in order to set-up the anti-pinch system correctly (e.g. for setting the thresholds). The scope of this project is to address these issues and to propose an innovative strategy: create an automated test-bench and integrate a machine

learning approach to automatically learn what is the best condition to trigger the anti-pinch. The project consists to build a motor bench that simulates a situation where a motor is moving a load and where, during the movement, a pinch can occur. In particular, the test-bench could simulate an automatic back-seat system that works as follow: from an initial position, where the back-seat is parallel to the ground, the motor must move the back-seat until a certain position (expressed in degree). If there is an obstacle during the run, the motor must stop the movement of the back-seat. It is important to recognize the presence of an obstacle. Avoid false negative is more important than avoid false positive, according to the level of safety requested. The project development is structured into different steps:

- The creation of a model of the system using Simulink, in order to have an idea of what happens to all the physical quantities involved in the system, such as motor absorbed current, torque load generated by the seat, velocity.
- The mechanically build of the test-bench using two identical motors, provided by Hall sensor, which rotors are physically connected. The two motors rotate in opposite direction (one of the two motors acts as a brake, simulating a load).
- The design of an electronic circuit able to allow a control board to command the two motors and to acquire measurements from them. In particular, it must be able to measure, for both motors, the velocity, the number of revolute, the absorbed current, and the supply voltage of the main motor.
- The software development, implemented on the board. The control board tasks are: control the motors, collect measures from sensors, detect pinch. The main motor is controlled by activating or deactivating a relay, the brake motor is current controlled using the current profile that simulate the load plus additive term that simulates the pinch (only after a certain time instant).
- The design of an HMI, to integrate with an automated test-bench, with Mathworks MATLAB software. The HMI must be able to communicate with control

board through serial communication. The HMI can send load parameters (e.g. weight of the back-seat) and pinch simulation parameters (e.g. time instant after that the pinch must be simulated). The HMI receives and shows also all the measures collected and the pinch detection.

- The development of a machine learning algorithm, starting from the collected data, using logistic regression. Logistic regression is a classification method used to predict a binary output, in this project, the output is represented by the presence, or not, of the pinch. The training, validation and test phase are performed using MATLAB software.

- The implementation of the machine learning anti-pinch algorithm using the logistic regression model, which parameters are obtained during the training phase. Since the machine learning algorithm performs a high number of operations, much importance is given to the optimization of the software.

- The comparison, after several test, between the machine learning anti-pinch algorithm and the classic anti-pinch algorithm in order to evaluate if the innovative method is more convenient or not than the classic one. The evaluation is done by analyzing advantages and disadvantages and doing considerations about these methods.

Therefore, the whole system can be modeled as follows:

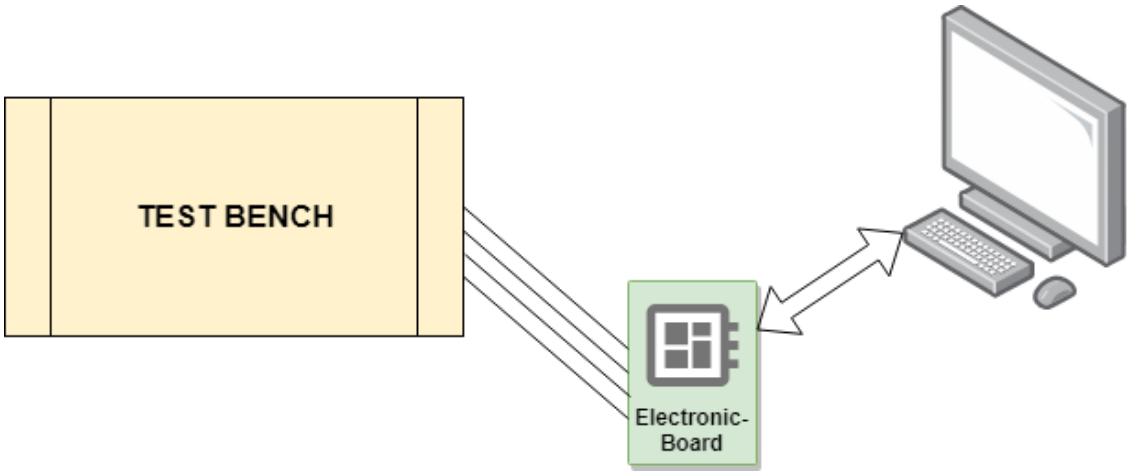

Figure 1.1 – General representation of the system

In the figure, the three main part of the system are highlighted:

1. The testbench. It is formed by the two motors connected one with each other and the electronic interfaces. The electronic interfaces allow the measurements the movement control and the pinch detection of the test bench

2. Electronic board. It is the board which task is to control the testbench. The board used is an Arduino Nano v3.3 equipped with ATmega328P microprocessor.

3. Computer, where it has been implemented the HMI and where the machine learning training has been performed.

1.1 Structure of the thesis

The thesis analyzes the development of the project step by step, after a study phase. In particular, the dissertation has the following structure: the first chapter offers a description of the pinch problem, applied in the automotive field, and an analysis of the advantages and disadvantages of the classical method. In the second chapter, a description of machine learning is provided and there is also illustrated an overview

of the different kind of algorithms. Then, there is a focus on the logistic regression, the machine learning algorithm used in this project. The third chapter is about the test-bench. It begins from the explanation of the simulation model. Then, the hardware used is examined, and the software development phase is described, from the algorithm, implemented on the control board, to the MATLAB HMI. In the fourth chapter, the machine learning method steps are illustrated starting from the training phase, executed on MATLAB, to the anti-pinch algorithm implementation on the control board. Fifth chapter, the conclusive one, examines the results obtained by using the two different approaches and suggests the possible features and improvements that can be implemented on the system in the future.

Chapter 2

Anti-pinch system

This chapter provides a description of the Anti-pinch problem in the automotive field, explaining the objective of an anti-pinch system, the applications field, the general requirements and the model of the system that is considered in this project. After that, the state-of-art implementation is described, focusing on its advantages and its disadvantages.

2.1 Problem description

In the engineering field, and in particular, in the automotive one, safety is an extremely important parameter to take in consideration during the development phase. In a modern vehicle, over the 20% of the purchase price is due to the electronic system on board.

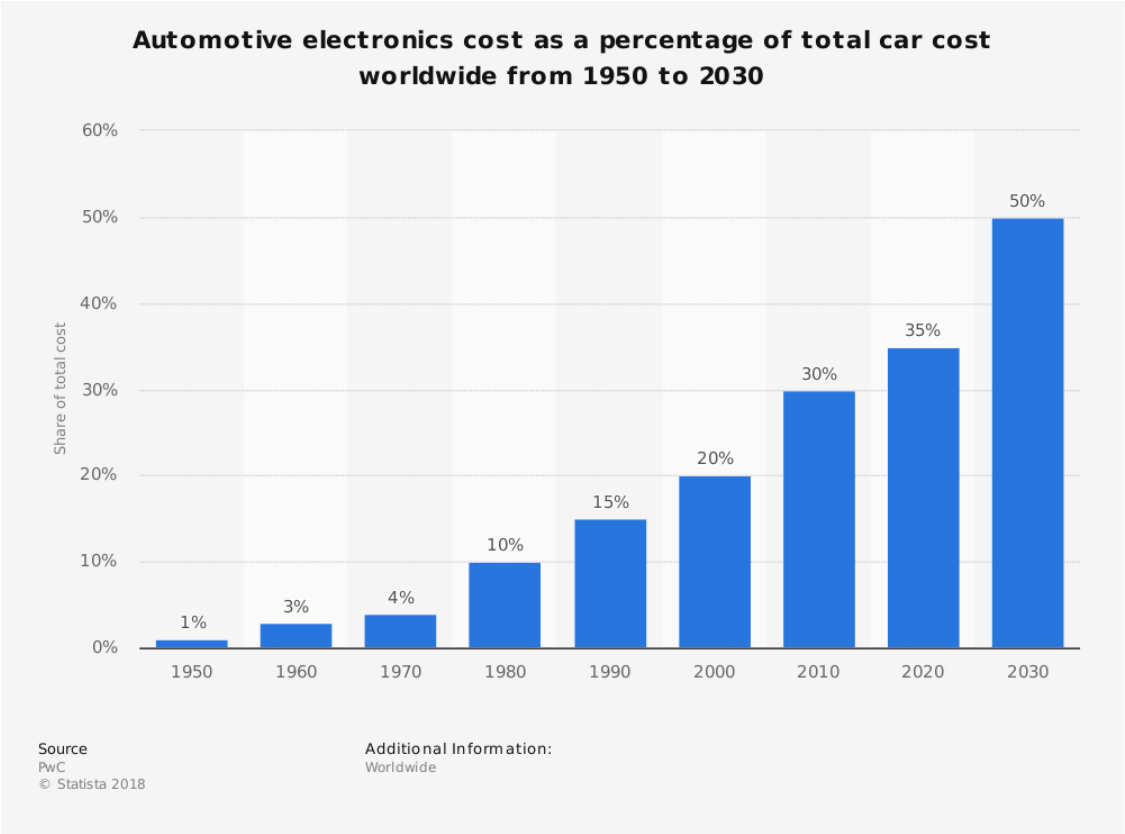

Figure 2.1 – Automotive electronic cost as percentage of total car cost

Electronically controlled systems are involved in a large variety of tasks, in the automotive field. Entertainment system, engine control, ADAS systems, safety systems are only some examples of the electronic in a vehicle.
This evolution brought significant improvements in automotive functionalities and performances, but, at the same time, it has raised the probability of safety issues concerning malfunctions of the electronic system [2][3]. Therefore, safety, in a vehicle, depends on the correct functioning of the electronic system and, consequently, the technologies act to detect fault, or accident protection, assume a primary role [2][4].

An *anti-pinch system* is a safety system that senses if a motor is running against an obstacle and prevents any injuries to people or damages to the obstacle or the

motor itself [25]. The anti-pinch is usually applied in those mechanism where the electronic actuation produces a movement that can be hindered by an external obstacle.

For instances, in a vehicle, it is applied to the automotive motorized window or back-seat apparatus which closes automatically, and which involve risks for trapping, squeezing or injury to people[5]. Indeed, an object that impedes the movement of the back-seat that tries to close, can cause damages for the object itself, or for the seat, or for both. Obviously, the obstacle can be represented not only by an object but also by a person. A control system is required to handle these situations in order to prevent damages. The requirements are usually given in terms of the force that shall trigger the anti-pinch. The criteria to evaluate an anti-pinch implementation can be seen as the number of false-positive and false-negative triggers. The customer may be more sensitive to the number of false-positive or false-negative according to the application and the level of safety and comfort required [25].

In this project, it is considered an automatic back-seat that shall move from an initial position, parallel to the ground, to a final position, corresponding to a certain degree movement.

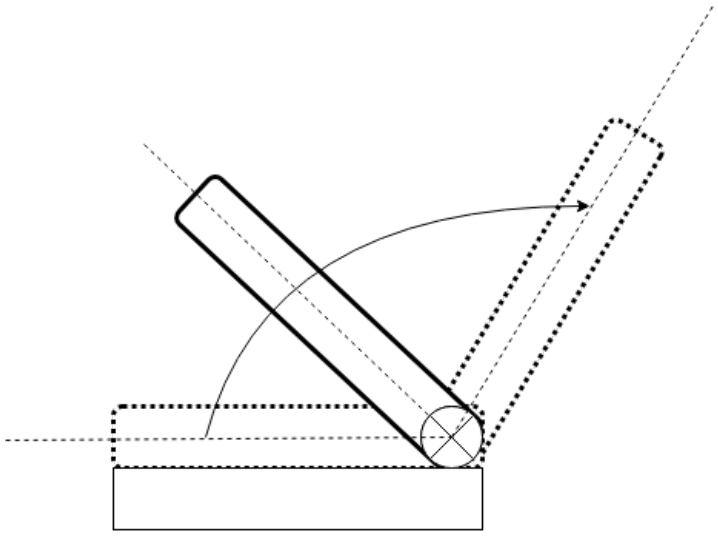

Figure 2.2 – Back-seat representation

The control system of an electric back-seat with anti-pinch function, is composed by a single DC motor. The sensors which the system is equipped with, consist on a Hall sensor and a current sensor. Moreover, the DC motor is connected to the back seat with a gear box, with an elevated gear ratio, that allows to reduce the rotational speed of the back-seat, respect to the motor velocity. The representation of the model is shown in the figure

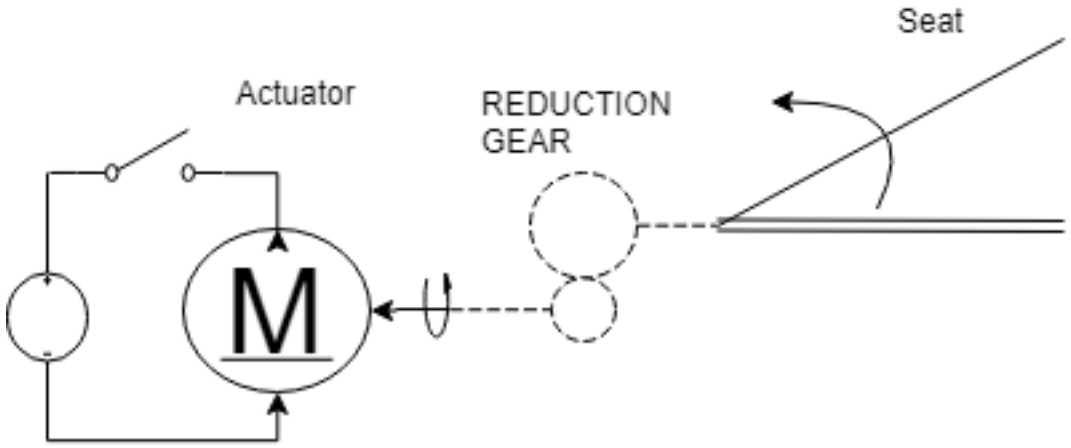

Figure 2.3 – General representation

2.2 Classical methods

Pinch detection methods have been widely studied [2][7]. The anti-pinch sensing is usually performed by current monitoring, hall sensors or a combination of both [23][24]. The state-of-the-art anti-pinch strategy can be split in two categories:

1. History based: more complex and robust. The current measured is compared against a nominal profile. If deviation from this maps occurs, the anti-pinch is triggered [25].

2. Derivative based: simpler to implement. It is based on monitoring the variation of current, or variation of velocity, or both. if the obtained value is higher than a fixed threshold, the anti-pinch is triggered [23][19][25].

In this project, a method belonging to the second category is considered. The anti-pinch algorithm checks continuously the variation of the current and the variation of the velocity [23][24]. Both are compared with a threshold and if they simultaneously report the presence of an obstacle, the motor is blocked [23][20].

2.3 General troubles

Although this task may look simple at first sight, many are the factor that add complexity and uncertainty to a standard algorithm: variations on system conditions, mechanical and electrical uncertainties, mechanical wear of the motor system, etc. Moreover, the implementation of this algorithm requires a lot of run and, consequently, a lot of time, in order to set-up the anti-pinch system correctly (e.g. for setting the thresholds) [25].

The problems of the current strategies are:

- If any part of the system changes, as well as the requirements, there is the need to reiterate the entire process of: performing many runs, manually evaluating the results, choosing the best strategy, implementing a new strategy or adapting the current strategy, evaluating the implementation against the requirements and criteria.
- It is very difficult to implement both strategies in order to have zero false-positive and zero-false negative in every possible condition. There may be time windows in which the anti-pinch is blind [18][19][21][23].
- It is time consuming to analyze the system, implement it, validate it and maintain it [25].

Additionally, many algorithms consider that seat, or window, movement has almost constant velocity, under normal operation conditions. Pinch situations have been determined based on the variation of the object velocity [2][5]. However, it

could happen that these methods can cause false alarms, for example when the velocity decreases, not for due to the presence of an obstacle but due to frictional or load torques [1][5] . Input current, as well as the motor angular velocity, has been considered to address this problem [2][8]. Moreover, measurement noise precludes accurate fault detection [2].

Chapter 3

Machine Learning

In this chapter, an introduction of what machine learning is, and its application fields, are explained. After that, there is a focus on a particular classification method, called logistic regression, that is used in this project.

3.1 Introduction

Nowadays, machine learning is used for a very large field of application: from medical field and engineering field to statistical field and financial field [28]. "Machine learning is the field of study that gives computers the ability to learn, without being explicitly programmed" [9]. In fact, in a traditional program, the developer must consider all the possible scenario that could occur, and he must write code to handle all these scenarios. Machine learning works different, by means of an iterative process. This process is composed by different phases:

- Model. A mathematical model of the problem is defined by the developer
- Train. A large set of data are collected from the model and they are used to get computers to learn or rather to optimize the mathematical model.

- Test and Validation. Another set of data, different from the train dataset, are used to verify the correctness of the method.

According to Tom Mitchell, a definition of well-posed Learning Problem is the follow: "a computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E" [10].
Machine learning can be grouped into three main categories, depending on the nature of the learning signal or response available to a learning system [11]. The three categories are:

1. **Supervised Learning**: when the machine learning algorithm learns from an example dataset and the associated target responses in order to predict the correct response when new examples are posed. Responses can assume a numeric values or they can be string labels, such as classes or tags. This approach mimics the human learning approach. In fact, it is similar to the situation where a student learns under the supervision of a teacher. The teacher provides good examples for the students to memorize, the students then derives the general rules from these examples, and they are able of predict the correct result of new examples [11][22].

2. **Unsupervised Learning**: when the machine learning algorithm learns from simple examples without any associated response, leaving to the algorithm the task to determine the data pattern. This type of algorithm tends to find all kind of unknown patterns in the data, such as new features that may represent a class or a new series of un-correlated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms [11][22]. Finding similarity to a real situations, *unsupervised learning* can be compared to how people figure out that certain objects or events are from the same class, such as by

observing how much they are similar with each other. Some examples of this kind of machine learning algorithm can be found on the web in the form of recommendation systems [11][22].

3. **Reinforcement Learning**: when the machine learning algorithm learns from an examples dataset where the response is not indicated, as in unsupervised learning. However, unlike the unsupervised learning, it is possible to accompany an example with a positive or a negative feedback according to the prediction that the algorithm does. Collecting errors helps the algorithm to learn, it is just like learning by trial and error. Reinforcement learning are widely used in the video games applications by execute actions and learn about the consequences, in order to avoid bad choices in future decisions [11][22].

Focusing only on the supervised learning algorithms, there are also two subcategories:

1. Classification: when the inputs are divided into classes and the learning algorithm must produce a model that will be able to assign the correct class to a new set of inputs. Spam filtering, is an example of classification, where the emails, which are the inputs, are classified into two classes, spam or not-spam [17].

2. Regression: similar to the classification algorithms but the output values are continuous rather than discrete. Estimating life expectance is an example of regression algorithm application [17].

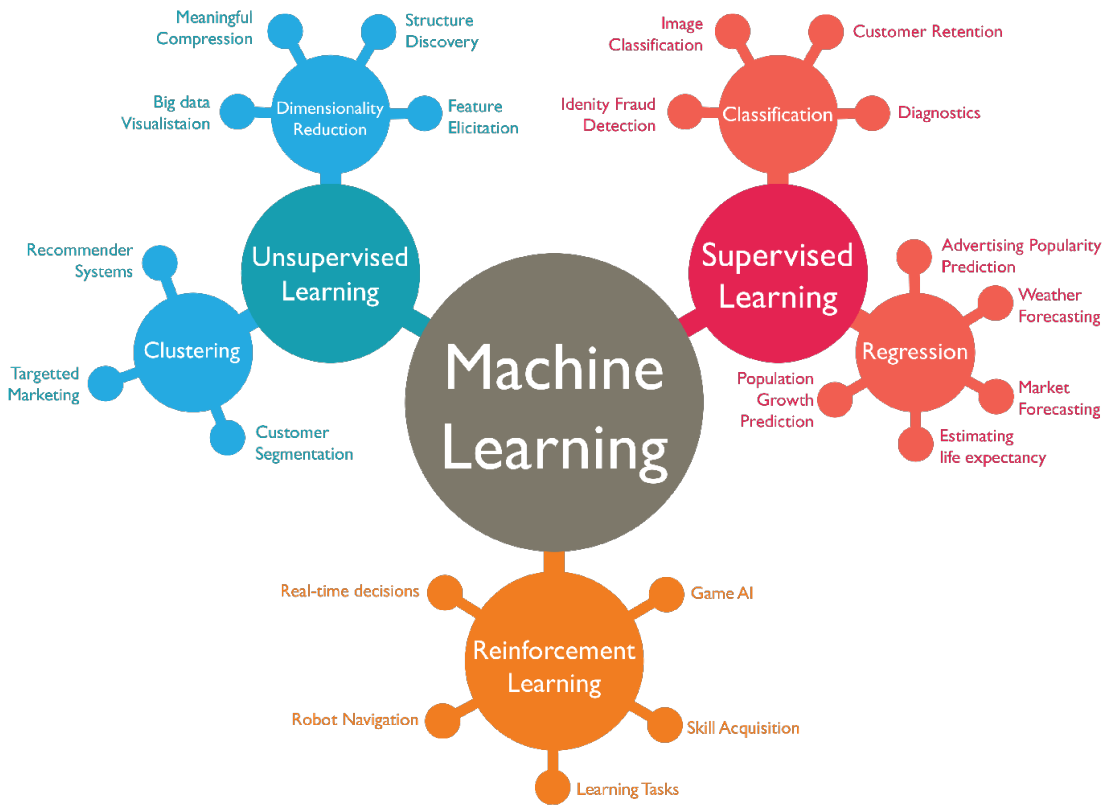

Figure 3.1 – Types of Machine Learning Algorithms

In this project the machine learning method used is the Logistic Regression, that is a supervised learning algorithm, belonging to classification group.

3.2 Logistic regression

Logistic Regression is a Machine Learning algorithm which is used for the binary classification problems, it is a predictive analysis algorithm and it is based on the concept of probability [12]. In a classification problem, input variables are taken, and the algorithm should try to fit the output onto a continuous expected result function. In other words, a function that map the input data "x" into an output data "y", must be used. The possible output values can be only 0 or 1, because of this method admits only binary output, usually labelled as follow:

y = 0	negative class
y = 1	positive class

Table 3.1 – Output values for logistic regression

The logistic regression uses a particular function known as “Sigmoid function”, or “Logistic function”, which has the following equation:

$$h_\theta(z) = \frac{1}{1 + e^{-z}} \tag{3.1}$$

where z is the linear model that represents the dataset.

$$z = \theta^T x \tag{3.2}$$

The following graph represents the behavior of the sigmoid function.

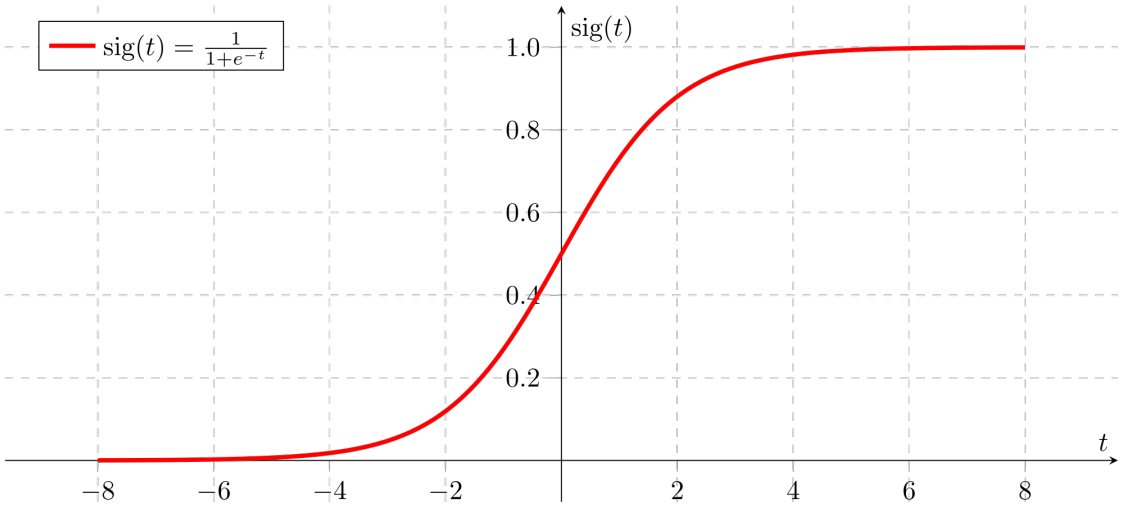

Figure 3.2 – Sigmoid graph

As it is possible to see in the figure, the sigmoid has the following important property:

$$0 \leq h_\theta(x) \leq 1 \tag{3.3}$$

The objective of the logistic regression is to find the parameters θ of the linear model that best fit the training dataset. Then, the logistic function $h_\theta(z)$ estimates the probability that the output is true (equal to 1), or false (equal to 0), by setting a threshold typically set equal to 0.5.

$$h_\theta(x) = P(y = 1 \mid x,\ \theta) \tag{3.4}$$

$$if\ h_\theta(x) \geq 0.5,\ then\ y = 1 \tag{3.5}$$

$$if\ h_\theta(x) < 0.5,\ then\ y = 0 \tag{3.6}$$

3.2.1 Cost function and gradient descent

In order to set the θ values the fit the parameters, it is necessary to have a dataset of training data. The training dataset is composed by "m" examples of input data "x" and known output data "y". Then, it is defined a cost function that represent the error between the output value obtained using the logistic regression model, and the real output data [17].

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}log(h_\theta(x^{(i)}) + (1 - y^{(i)})log(1 - h_\theta(x^{(i)})] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2 \tag{3.7}$$

λ represents the regularization term and it is added in order to prevent over-fitting problem.

By using the gradient descent method, it is possible to find the minimum of the cost function $J(\theta)$. Gradient descent is an iterative algorithm, that $\forall j$ repeats the following equation:

$$\theta_j := \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}\theta_j] \tag{3.8}$$

Only for computing θ_0, the regularization parameter λ must be set equal to zero [17].

3.2.2 Validation and test

Given many models, for example with different number of features, it is possible to use a systematic approach to identify which model is better. From the initial dataset, only a part is used for training, typically the 60% of the entire dataset. The remaining data are divided into "cross-validation" dataset 20% and "test error" dataset 20% [17]. "Cross-validation" is an operation that select the best model based on the error computed as a cost function without the regularization term λ. It is computed on m_{cv} values, by using the same cost function used for the training phase, without the regularization term.

$$J_{cv} = -\frac{1}{m_{cv}}\sum_{i=1}^{m_{cv}}[y_{cv}^{(i)}log(h_\theta(x_{cv}^{(i)}) + (1 - y_{cv}^{(i)})log(1 - h_\theta(x_{cv}^{(i)})] \tag{3.9}$$

"Test error" is another operation that indicates if the model has a good generalization of the problem. Instead of computing the same cost function used for the training phase, it is computed the misclassification error, based on m_{test} values

$$err(h_\theta(x), y) = \begin{cases} 1, & \text{if } h_\theta(x) \geq 0.5 \; and \; y = 0 \; or \; h_\theta(x) < 0.5 \; and \; y = 1 \\ 0, & \text{otherwise} \end{cases} \tag{3.10}$$

$$J_{test} = \frac{1}{m_{test}}\sum_{i=1}^{m_{test}} err(h_\theta(x_{test}^{(i)}), y_{test}^{(i)}) \tag{3.11}$$

3.2.3 Evaluation

From all the data obtained performing test, the results obtained can be classified as:

1. True positive TP
2. False positive FP
3. False negative FN
4. True negative TN

And it is possible to build a table like this:

	Predicted class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

Figure 3.3 – Table precision / recall

In order to evaluate the algorithm, some parameters are defined:

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{3.12}$$

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{3.13}$$

$$Accuracy = \frac{\mathrm{TP} + \mathrm{FP}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN} + \mathrm{TN}} \tag{3.14}$$

The precision and the Recall can be adjusted by changing the threshold when computing the output value. To estimate a good trade-off between Precision and Accuracy, another parameter, known as F_1 score, is defined [17].

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{3.15}$$

Chapter 4

Test Bench

The test-bench overall system can be represented with the following diagram:

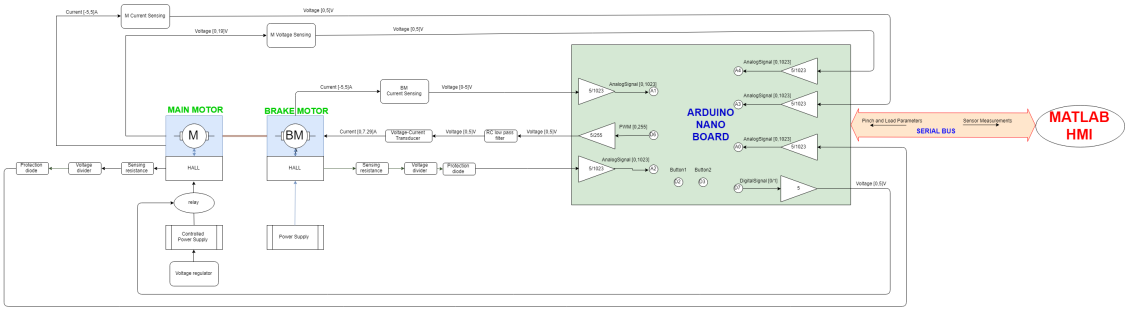

Figure 4.1 – Block diagram of the test-bench

In the following sections the steps for building the test-bench are presented. Initially, in the first section, it is described the development of the simulation model through Simulink software. Then, in the second section, it is described the components used for building the test bench. and the hardware architecture, that is the electronic components that allow the physical system to communicate with the control board. In the third section, the software architecture is illustrated, explaining how the Arduino board interacts with the bench test. The fourth section illustrates the Arduino-HMI communication protocol and all the functionality that the HMI has. Finally, in the fifth section, the results obtained are explained.

4.1 Simulation model

The first part of the project consists to create a model of the system in order to perform many simulations and to show the effect which alternative conditions have on the system. The test-bench must simulate a real scenario, where a DC motor moves a back-seat. So, initially, a model of the motor, which will be implemented in the real application, is created using Simulink. The model of this motor is useful to obtain an absorbed current profile and a load profile. that are the reference values to use in the bench motor simulation. The referenced motor is a brushed DC motor with an integrated Hall effect sensor, connected to different gears that reduce the speed increasing the torque. By means of the values obtained in the datasheet of the component and from measurements, parameters of the Motor are obtained and they assume the values shown in the table 4.1.

Variable	Description
$R_a = 0.9\Omega$	Armature resistance
$L_a = 10^{-3}H$	Armature inductance
$K_e = 3.3 \cdot 10^{-3} \frac{V}{RPM}$	Back-emf constant
$I_0 = 2.5A$	No-load current
$V_0 = 13V$	DC supply voltage in no-load condition
$J = 100g \cdot cm^2$	Rotor inertia
$gr = 724.83$	Gear ratio

Table 4.1 – Reference Motor parameters

Because of in the datasheet there is not indicated, the rotor inertia has been chosen by looking at datasheet of similar motors. The gear ratio is computed measuring the motor velocity with and without the reduction gears, that have, respectively, a value of 3240 RPM and 4.47 RPM.

Then, the model of the seat is defined in Simulink. The seat is approximated as a parallelepiped, with mass M_s and subject to a weight force P_{load}, that generates a load torque T_{load} the depends on the inclination of the seat itself, according to the following equation (where α is the angle between the horizontal axis and the seat)

$$P_{load} = M_s \cdot g \tag{4.1}$$

$$T_{load} = P_{load} \cdot a \cdot \cos(\alpha) = M_s \cdot g \cdot a \cdot \cos(\alpha) \tag{4.2}$$

In the table below are shown the parameters used for the load simulation.

Variable	Description
$M_s = 10Kg$	Seat mass
$a = 0.7m$	Distance between hinge and the seat CoG
$g = 9.81\frac{m}{s^2}$	Gravity acceleration

Table 4.2 – Load model parameters

The Simulink model that represents the seat is the following:

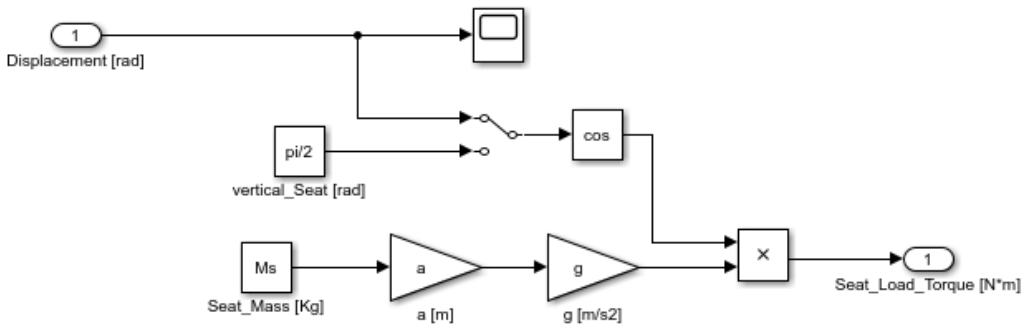

Figure 4.2 – Simulink seat model

Now, connecting the load generated by the model of the seat to the reference motor, it is possible to obtain interesting graphs about the motor absorbed current and the load generated by the seat.

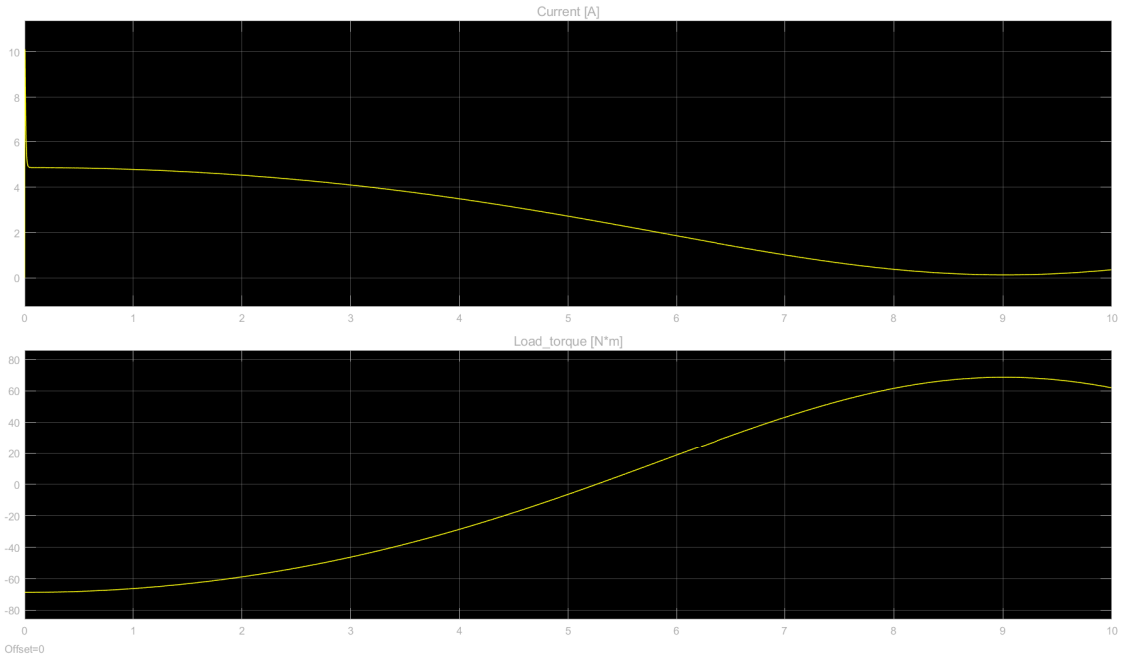

Figure 4.3 – Current and load behavior of the reference motor

To better understand the behavior of the absorbed current with respect to rotor position, it is shown as function not of the time but of the angular displacement α of the seat:

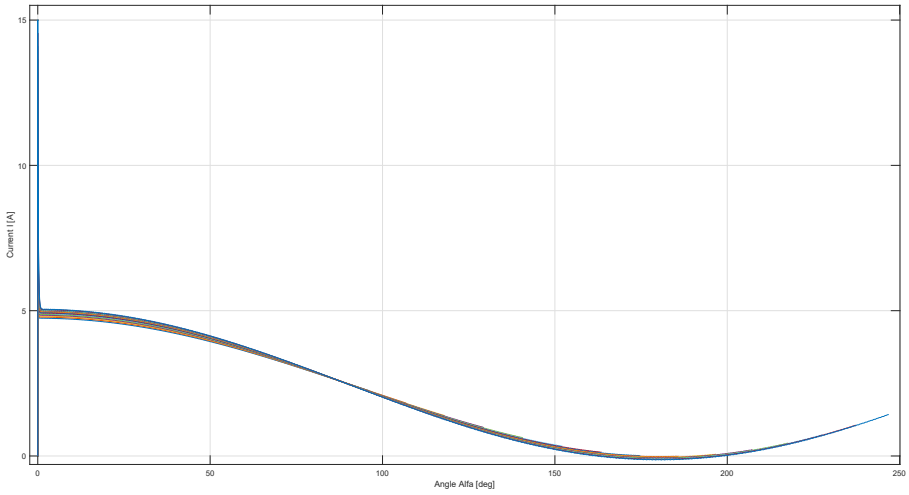

Figure 4.4 – Reference motor current as function of load angle

As it can notice, when the angle is zero degree (the seat is parallel to the horizontal line), the value of the load is maximum and, therefore, also the value of the current absorbed by the motor. At 90 degrees, the seat is in the vertical position and so the distance between the center of gravity and the seat hinge is null. This means the torque is null and the current assumes its no-load value. After 90 degree, the torque generated by the seat has the same direction of the main motor and it reduces the current needed to move itself.

The motor that will be used in the test-bench as a brake, must generate a torque which behavior must be as similar as possible to the behavior represented in the figure.

The motors used in the test bench are two identical Nidec motor (404.744), very similar to the reference motor (brushed DC motor with an integrated 2-pins Hall effect sensor).

Also these two motor are modeled using Simscape toolbox. The parameters are obtained, as for the previous motor, from the datasheet and from measurements

and they are the following:

Variable	Description
$\omega_{no-load} = 2700RPM$	No-load speed
$T_{nom} = 0.1Nm$	Nominal torque
$V_{nom} = 12V$	Nominal voltage
$P_{nom} = 21.2W$	Nominal power
$I_{nom} = 4.5A$	Nominal current
$I_0 = 0.82A$	No-load current
$V_0 = 13V$	DC supply voltage in no-load condition
$J = 150g \cdot cm^2$	Rotor inertia
$R_a = 0.17/Omega$	Armature resistance
$V_{supply} = 12V$	Voltage supply

Table 4.3 – Parameters Bench Motors (Nidec 404.744)

The two Nidec motors are connected but they rotate in the opposite direction because one of them, the brake motor, works as a brake. The connection, in the simulation, is obtained using a gearbox with a gear ratio of -1 connected to the mechanical part (the green one) of the two motors

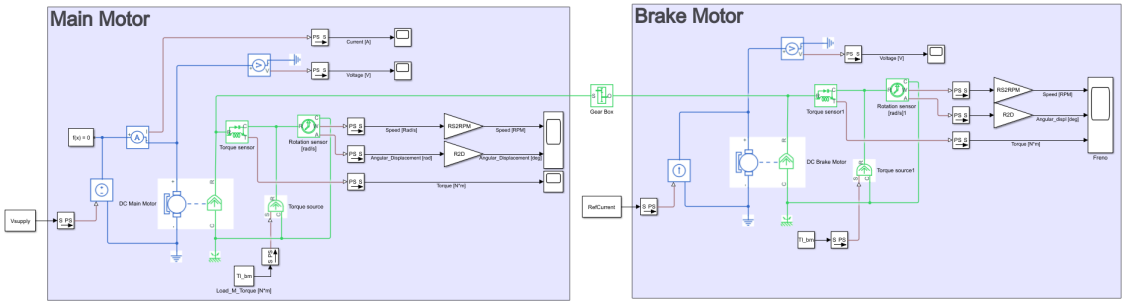

Figure 4.5 – Simscape model of the two motors connected

The Main motor is not controlled neither in velocity nor in current, in fact it is only connected to a voltage supply with a constant DC voltage. On the contrary, as

said before, the brake motor is controlled by current, in order to obtain a behavior that simulates the presence of a back-seat as a load.

As it is expected, the velocity of the two motors are the same but opposite, as it is shown in the figures:

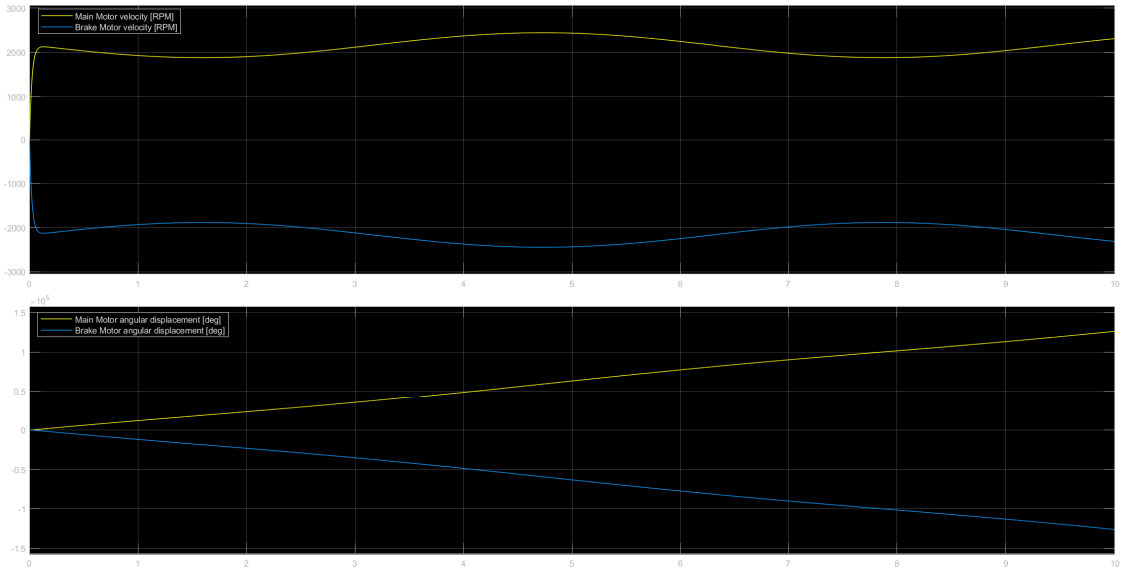

Figure 4.6 – Motors velocity and angular displacement graphs

To simulate the presence of a pinch, during the execution of the simulation, the current generated by the reference motor, and used to control the brake motor, is summed up with another current contribution. The pinch is simulated by the "Pinch Simulator" subsystem, based on parameters defined a priori, which are:

- I_{start}, that is the initial value of the current that the pinch causes;
- I_{max}, that is the maximum value that the current created by the pinch can assume;
- I_{min}, that is the minimum value that the current created by the pinch can assume;
- t_{start}, that is the time instant when the pinch occurs.

The Simulink Model of the pinch simulator is shown in the figure:

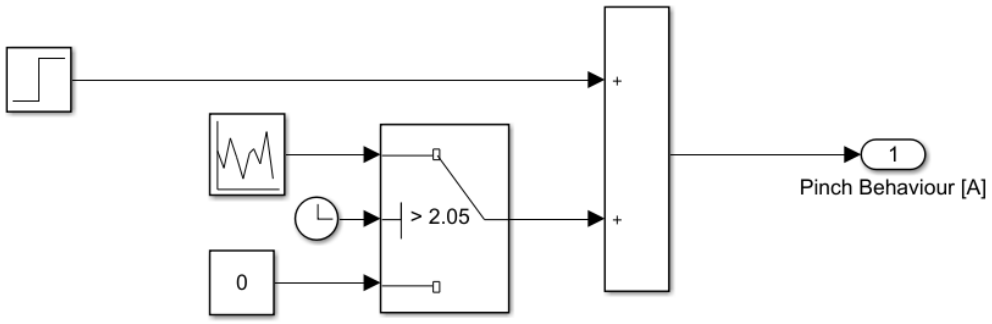

Figure 4.7 – Pinch simulator model

In the following figures, it is shown:

- on the left, an example of current behavior generated by a pinch with these parameters ($I_{start} = 2A$, $I_{max} = 2.5A$, $I_{min} = 1.8A$, $t_{start} = 2s$)
- on the right the overall current behavior

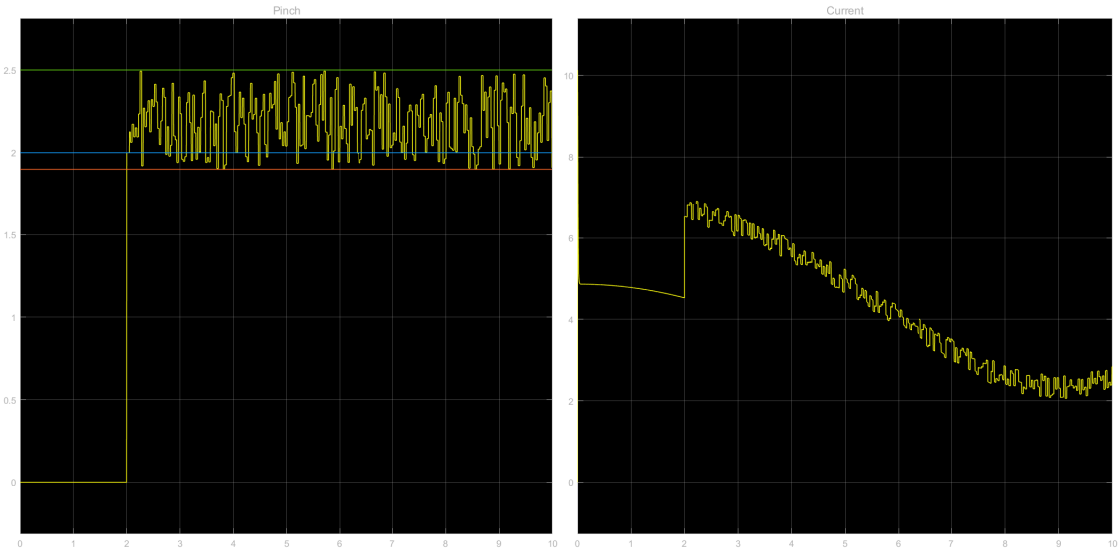

Figure 4.8 – Pinch behavior (on the right) - current behavior with pinch (on the left)

The effect that pinch causes are an increase of the torque generated by the brake motor and, as a consequence, the decrease of the main motor velocity and an increase of the absorbed current of the main motor. In fact, the absorbed current and the velocity are the two parameters of the main motor that must be taken care of, in order to apply the classical anti-pinch detection algorithm. As expected, the results from the simulation are the following:

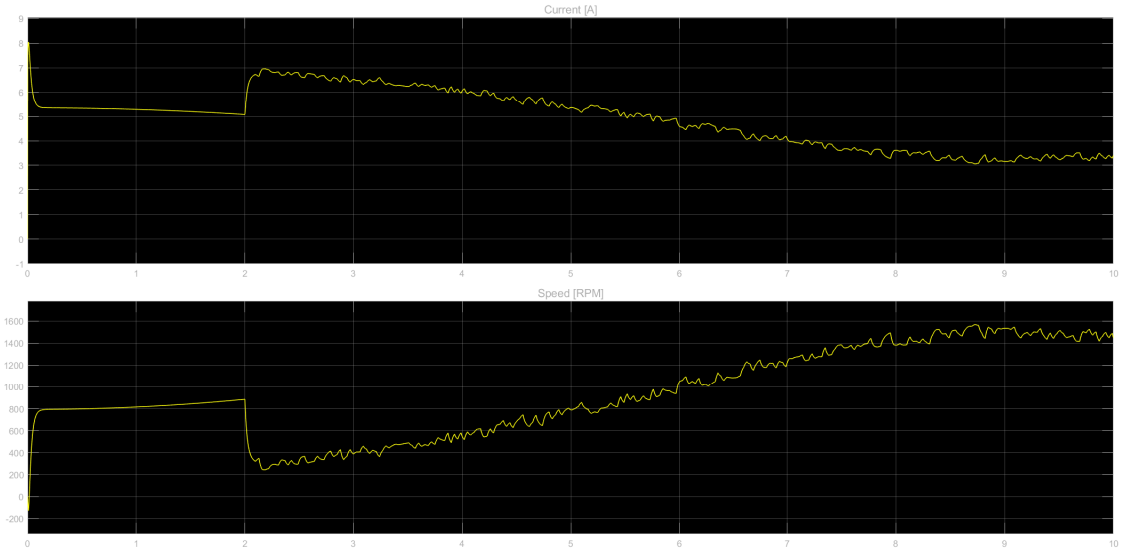

Figure 4.9 – Motor absorbed current and Motor velocity in the presence of a pinch

Finally, in the simulation is implemented a subsystem called “StopConditions”. Inside this subsystem there is a switch block that check if the motor reaches the end-of the run, set to a certain angular displacement, and there is also implemented the classic anti-pinch algorithm. The classical anti-pinch algorithm works as follow: it checks at every time step the value of the main motor current and the value of the main motor velocity, it computes the derivative of the two signal and, if the derivatives of the current is higher than a positive threshold and, simultaneously,

the derivative of the velocity (the acceleration) is lower than a negative threshold, the presence of the pinch is reported. The two thresholds are obtained by watching the behavior of the two derivatives in the presence, or not, of the pinch.

If the anti-pinch algorithm reports a pinch or the main motor reaches the end of the run, both motors are switched-off.

Another element is added in the simulation that has the only task to show when the pinch is effectively simulated. In fact, it is important comparing it with the anti-pinch detection algorithm and checking if the pinch is correctly detected or if it is a false positive.

In the figure below is shown an example of execution of the algorithm:

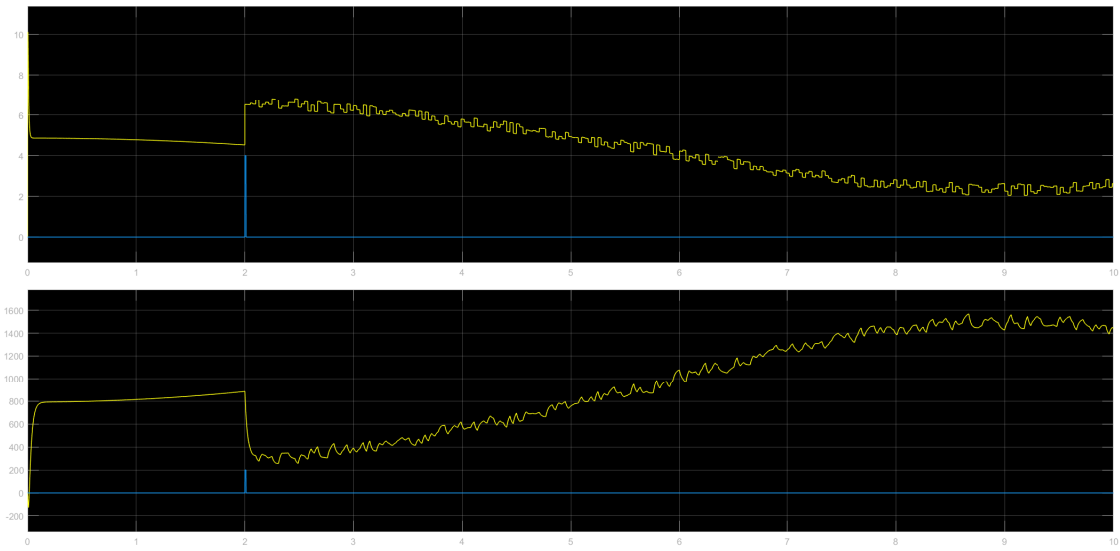

Figure 4.10 – Classic anti-pinch algorithm on the simulation

In the first image, the yellow line represents the absorbed current, while the blue line represents the pinch detection. In the second image, the yellow line represents the motor velocity and, as before, the blue line represents the pinch detection.

Putting together all the parts described, the complete simulink model is the following:

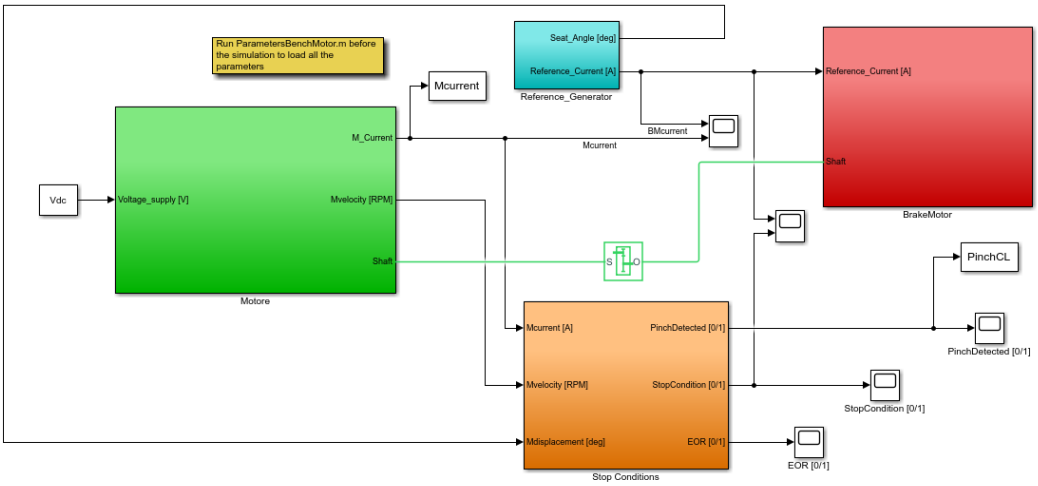

Figure 4.11 – Complete Simulink model of the system

4.2 Hardware interfaces

4.2.1 Hardware components

Aduino

The board used to control the whole system is an Arduino Nano v3.3 board.

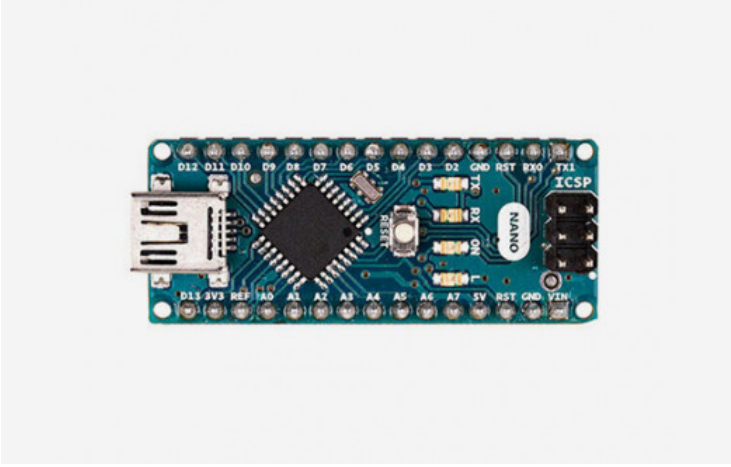

Figure 4.12 – Arduino Nano v3.3 board

Arduino is an open-source electronics prototyping platform based on flexible hardware and software. Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling actuators, like motors. The microcontroller on the board is programmed using the Arduino programming language and the Arduino development environment. Arduino projects can be stand-alone, or they can communicate with software on running on a computer (e.g. MATLAB, LabVIEW). Arduino Nano is a surface mount breadboard embedded version with integrated USB. It is a smallest, complete, and breadboard friendly [26][27]. The microcontroller mounted on the Arduino Nano board is an ATmega328P, with the following technical specification: Features:

Microcontroller	Atmel ATmega328
Operating Voltage (logic level)	5 V
Input Voltage (recommended)	7-12 V
Input Voltage (limits)	6-20 V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	8
DC Current per I/O Pin	40 mA
Flash Memory	32 KB (of which 2KB used by bootloader)
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz
Dimensions	0.70” x 1.70”

Table 4.4 – Arduino Nano technical specification

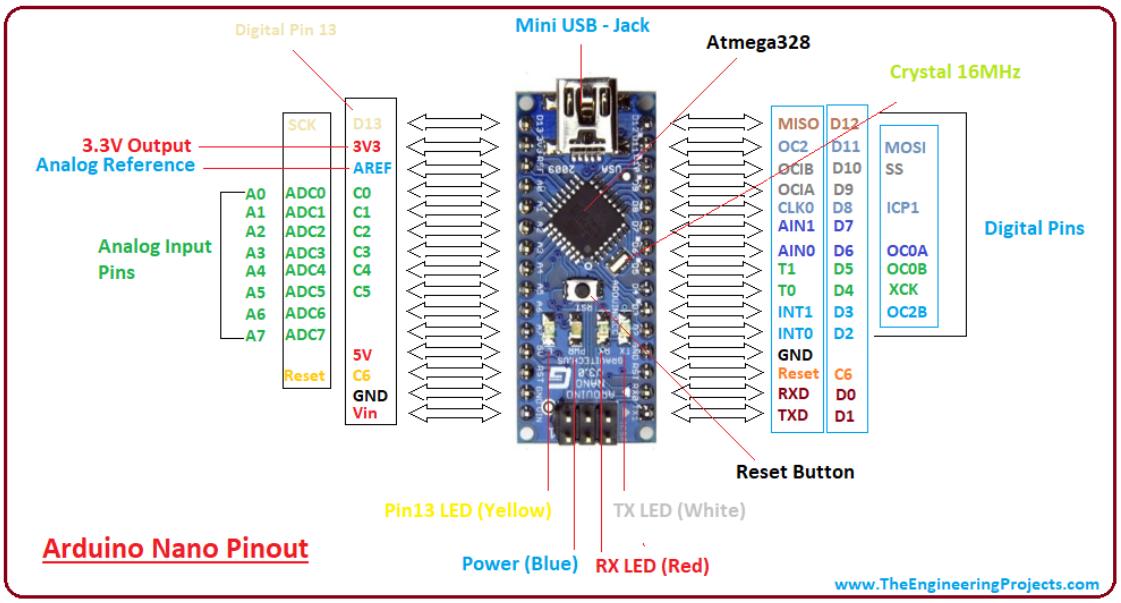

Figure 4.13 – Arduino Nano pin-out scheme

The main features are summarized in the table below:

Automatic reset during program download
Power OK blue LED
Green (TX), red (RX) and orange (L) LED
Auto sensing/switching power input
Small mini-B USB for programming and serial monitor
ICSP header for direct program download
Standard 0.1” spacing DIP (breadboard friendly)
Manual reset switch

Table 4.5 – Arduino Nano features

For this project, it is used this particular board because of its low power, that is comparable with the board implemented in a real anti-pinch system in the automotive application. Arduino is connected to a computer running MATLAB software through the serial bus. The HMI allows the user:

1. To see the graph of all the sensors measurements;
2. To change pinch and load parameters
3. To save a .csv file with all data
4. To switch on and switch off the system
5. To clear all data collected
6. To choose if an anti-pinch algorithm must be used and which one must be used (between classic algorithm and machine learning algorithm)
7. To connect or disconnect Arduino and the serial bus

DC motors

The brushed DC motors used in this project are Nidec 404.744.

Figure 4.14 – Nidec 404.744 Motor

Brushed DC motor are rotary electrical machines, power supplied by electrical direct current. The mechanical commutation occurs internally by means of stationary magnets and rotating magnets. DC motors are widely used thanks to their low initial cost, their high reliability, and because they are easy to control. The drawbacks are the high maintenance and low life for high intensity uses, especially due to the presence of the brushed. Moreover, the mechanical commutation leads

to a phenomenon known as current ripple [29]. For this project, it is used this kind of electrical machine because it is the same kind of motor used in a real automotive application. The two motors are mechanically connected. And the Arduino Nano board controls both motors. The main motor is switched-on and switched-off by Arduino using a relay, connected to a power supply. The brake motor is current-controlled through a PWM signal, from the Arduino board, using a voltage-current transducer circuit with a RC low pass filter. The Hall sensor of the two motors are measured using a sensing resistance plus a voltage divider (to convert the input voltage into the range [0, 5]V , readable by Arduino) and a protection diode. There are implemented also two current sensors, one for each motor, and a voltage sensor only for the main motor, obtained using a voltage divider.

4.2.2 Input interfaces

Since the Arduino Nano board output pins span from 0 to 5 Volts, it is necessary to design an electronic board that handle the output interface and the input interface.

The input interface must get the data that come from the sensors installed on the system and make them readable by Arduino. It is composed by 5 parts:

- Two identical circuits able to correctly measure the square wave generated by the two Hall sensors and to reduce the voltage of the signal into the interval [0, 5V]. The circuit must attenuate the voltage output of the hall sensor to a range from 0V to 5V. A simple Voltage divider circuit is used, with a protection diode. The maximum input value is 15.8V, the resistances are dimensioned to obtain 5V from 15.8V. The sensor is modelled as a current generator.

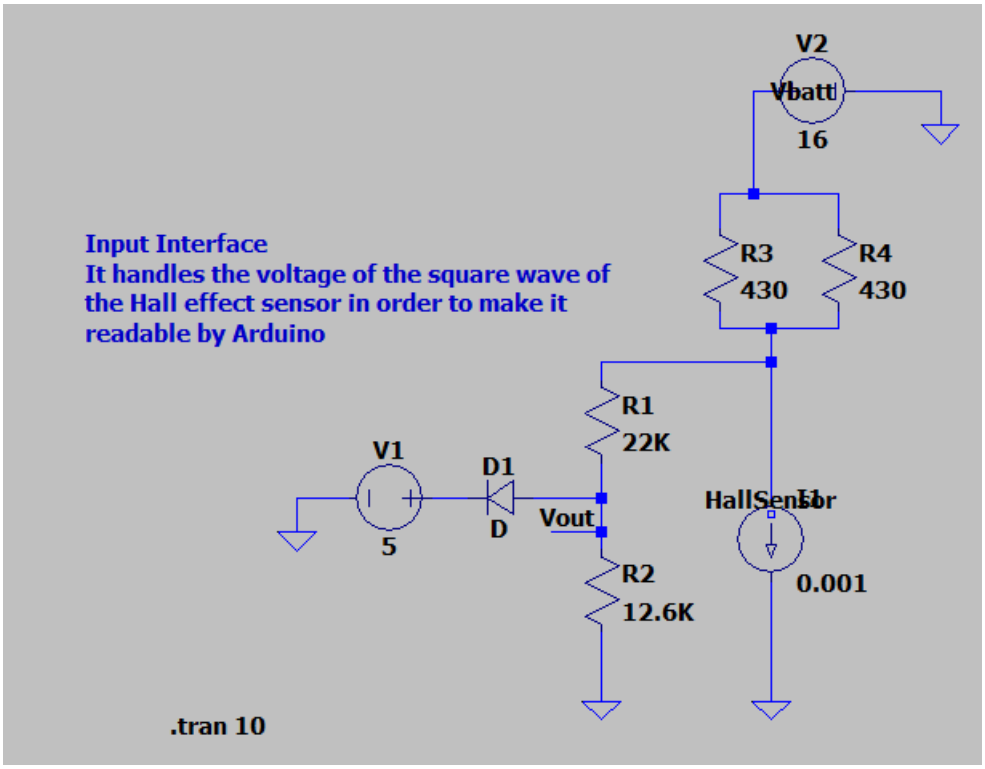

Figure 4.15 – Input interface model (using LTspice)

- Two identical current sensors ACS712x05B that measures the current absorbed by the two motors. The sensor has the following characteristics:

COMMON OPERATING CHARACTERISTICS[1] over full range of T_A, C_F = 1 nF, and V_{CC} = 5 V, unless otherwise specified

Characteristic	Symbol	Test Conditions	Min.	Typ.	Max.	Units
ELECTRICAL CHARACTERISTICS						
Supply Voltage	V_{CC}		4.5	5.0	5.5	V
Supply Current	I_{CC}	V_{CC} = 5.0 V, output open	–	10	13	mA
Output Capacitance Load	C_{LOAD}	VIOUT to GND	–	–	10	nF
Output Resistive Load	R_{LOAD}	VIOUT to GND	4.7	–	–	kΩ
Primary Conductor Resistance	$R_{PRIMARY}$	T_A = 25°C	–	1.2	–	mΩ
Rise Time	t_r	I_P = I_P(max), T_A = 25°C, C_{OUT} = open	–	5	–	µs
Frequency Bandwidth	f	–3 dB, T_A = 25°C; I_P is 10 A peak-to-peak	–	80	–	kHz
Nonlinearity	E_{LIN}	Over full range of I_P	–	1.5	–	%
Symmetry	E_{SYM}	Over full range of I_P	98	100	102	%
Zero Current Output Voltage	$V_{IOUT(Q)}$	Bidirectional; I_P = 0 A, T_A = 25°C	–	V_{CC} × 0.5	–	V
Power-On Time	t_{PO}	Output reaches 90% of steady-state level, T_J = 25°C, 20 A present on leadframe	–	35	–	µs
Magnetic Coupling[2]			–	12	–	G/A
Internal Filter Resistance[3]	$R_{F(INT)}$			1.7		kΩ

x05B PERFORMANCE CHARACTERISTICS T_A = –40°C to 85°C[1], C_F = 1 nF, and V_{CC} = 5 V, unless otherwise specified

Characteristic	Symbol	Test Conditions	Min.	Typ.	Max.	Units
Optimized Accuracy Range	I_P		–5	–	5	A
Sensitivity	Sens	Over full range of I_P, T_A = 25°C	180	185	190	mV/A
Noise	$V_{NOISE(PP)}$	Peak-to-peak, T_A = 25°C, 185 mV/A programmed Sensitivity, C_F = 47 nF, C_{OUT} = open, 2 kHz bandwidth	–	21	–	mV
Zero Current Output Slope	$\Delta I_{OUT(Q)}$	T_A = –40°C to 25°C	–	–0.26	–	mV/°C
		T_A = 25°C to 150°C	–	–0.08	–	mV/°C
Sensitivity Slope	ΔSens	T_A = –40°C to 25°C	–	0.054	–	mV/A/°C
		T_A = 25°C to 150°C	–	–0.008	–	mV/A/°C
Total Output Error[2]	E_{TOT}	I_P =±5 A, T_A = 25°C	–	±1.5	–	%

Figure 4.16 – Datasheet of the ACS712x05B

- One voltage divider that convert the supply voltage of the main motor in a voltage signal that belongs to the interval [0, 5V].

4.2.3 Output interfaces

The output interface must control the two motors. The main motor can be simply controlled by activating or de-activating a relay, while the brake motor must be controlled by current. Thus, the output interface is composed by:

- A relay that, based on the Arduino output voltage signal, can switch-on or switch-off the main motor. The relay is controlled by a PNP transistor connected to the output port of the board.
- A voltage-to-current transducer that converts the voltage generated by Arduino output pin into a current value that control the brake motor. This

circuit must convert a voltage, that span from 0V to 5V, to a current value. The transducer is composed by a Darlington transistor, since we need high level of current, and by a low pass passive filter. The circuit is the following:

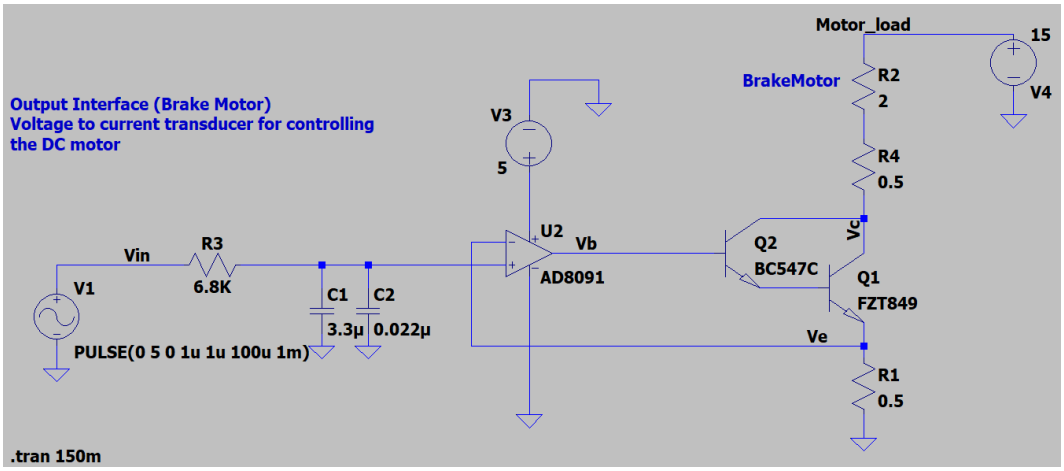

Figure 4.17 – Output interface model (using LTspice)

V1 is the signal that comes from Arduino and R3 represent the motor. To dimension the value of the resistor R1, the following constraints are considered:

- $V_{in} \in [0, 5]V$, the voltage interval that Arduino can provide
- $I_l \in [0, 7]A$, the maximum absorbed current by the motor
- $V_{ce} \cdot I_l < P$, the constraint about the power of the transistor
- $V_{ce} \geq 2V$, the constraint about the collector-emitter voltage
- $R = \frac{V_e}{I_l}\Omega$, the constraint about the value that the resistance can have

The resistance, that satisfy the constraints, has a value of $R = 0.48\Omega$ rounded to $R = 0.48\Omega$.

The schematic of the complete circuit is the following:

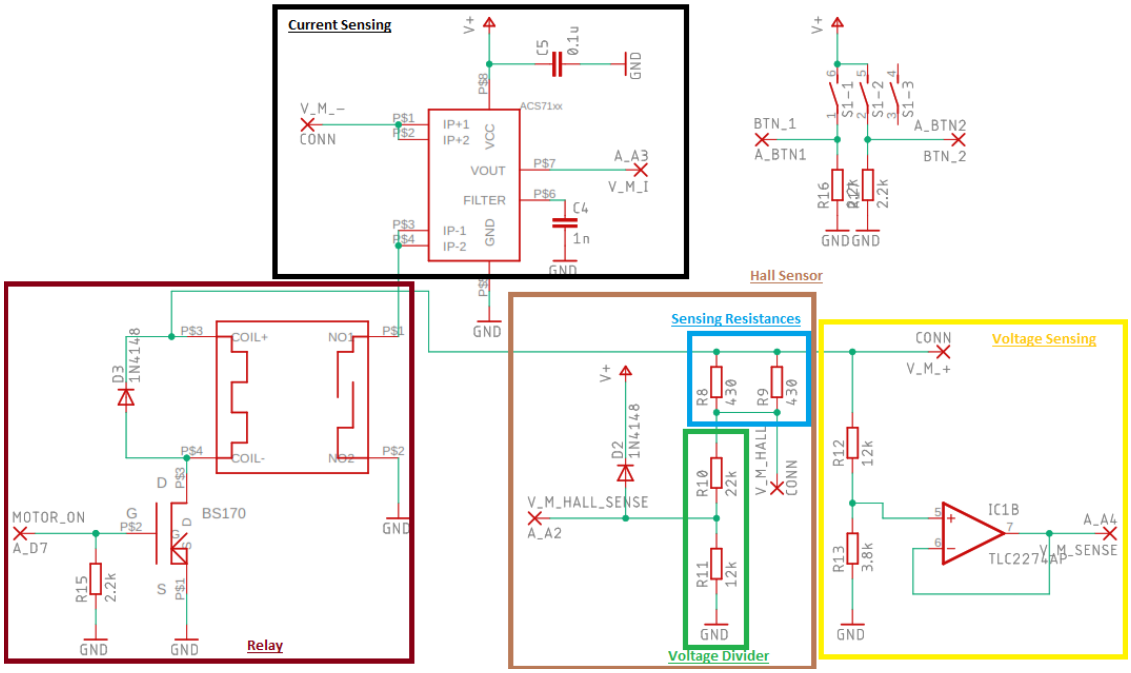

Figure 4.18 – Main motor schematic

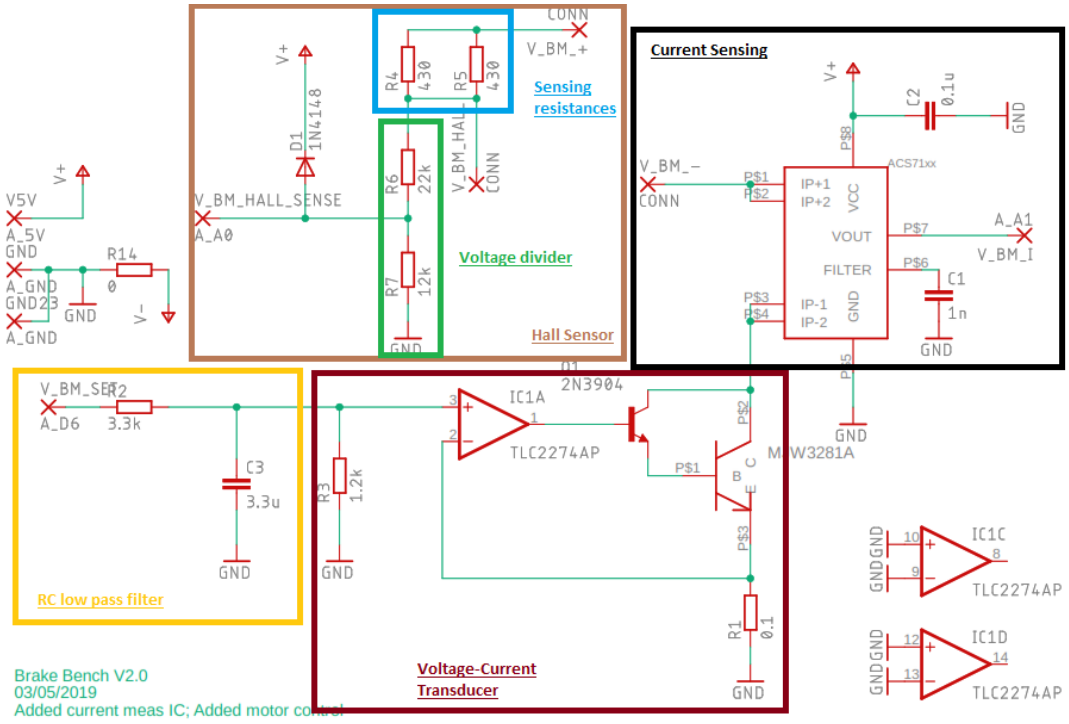

Figure 4.19 – Brake motor schematic

In the figures are also highlighted some parts that are shown in the block diagram represented in figure 4.1.

4.3 Software architecture

The software architecture, developed for the Arduino board, has three main tasks:

- The first one is to collect all the data provided by the sensors implemented in the system.
- The second one is to control the test-bench, switching-on and switching-off the two motors, and detecting pinch.
- The third one is to establish a communication channel with the HMI, created using MATLAB, in order to receive and send data.

The scheme of the software architecture implemented is the following:

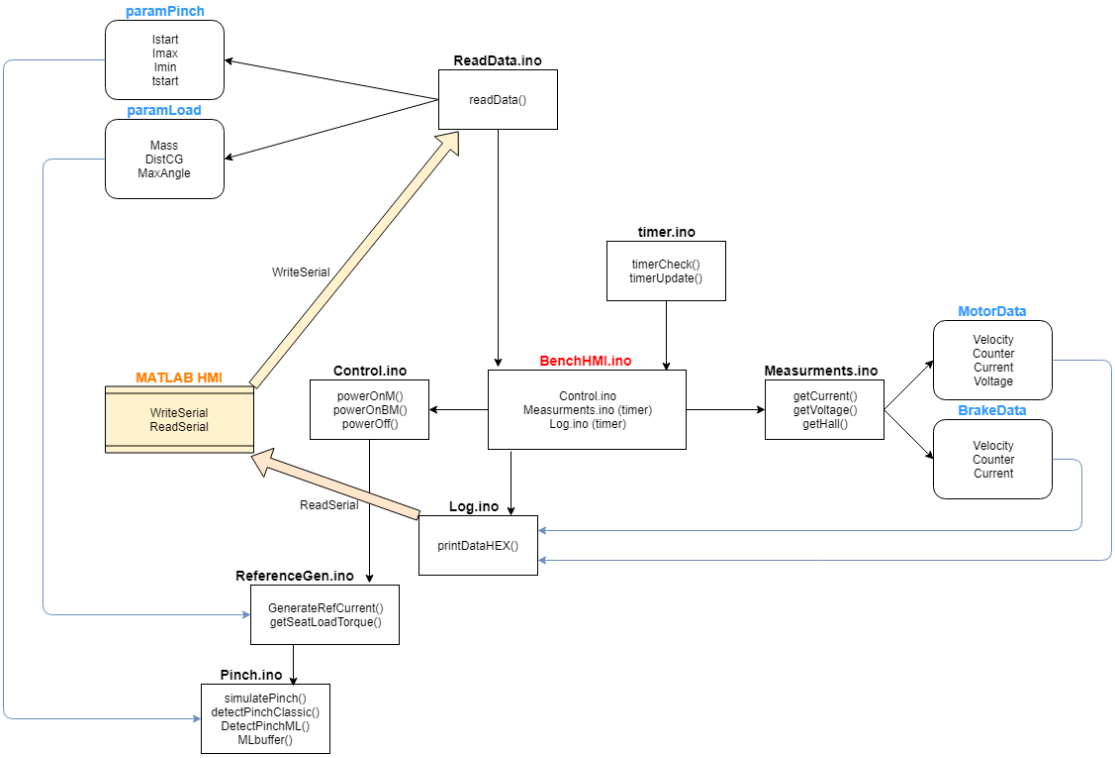

Figure 4.20 – Software architecture scheme

The main block is called "BencHMI.ino" and it calls all the other function that are categorized into different blocks. The blocks with blue names, represents the structure where parameters and measurements are stored. "paramPinch" and "paramLoad", that stores the system parameters, are filled by MATLAB through serial communication. While, "MotorData" and "BrakeData" structures are read by MATLAB. All the other blocks contain all the function used, categorized by their functionalities.

In this section, only the implementation of the classical anti-pinch algorithm will be treated, while the machine learning algorithm implementation will be explained in the next chapter.
Because of, in this project, it is used Arduino Nano board as a controller, which is not a so fast board, it is important to optimize the code and to reduce its complexity in order to do not require a great effort to the board.

4.3.1 Measurements

Arduino must acquire all the data provided by the five sensors of the system. For each sensor, there is the relative function that get the raw data and that convert it into a meaningful value. For measuring the voltage, Arduino read a raw value that span in the interval [0, 1023] (the maximum resolution of Arduino) and it must be converted into the correspondent voltage value. The conversion values are obtained by trying different configuration and adjusting the value according to measurements obtained from the oscilloscopes. In the figure below is represented the conversion gain of the current sensor.

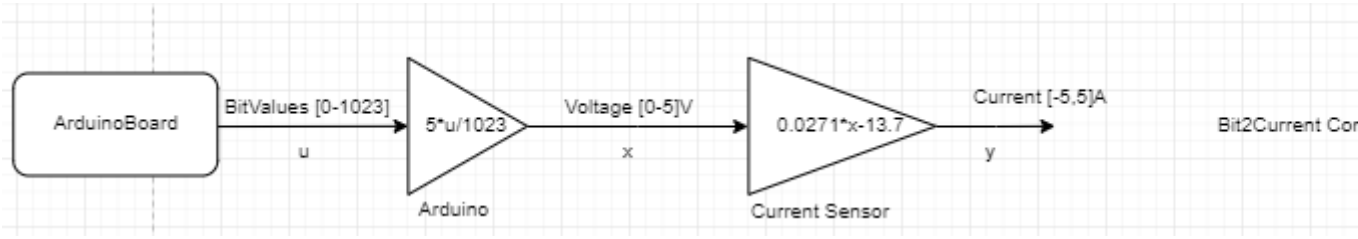

Figure 4.21 – Current sensor gain

The same approach used to convert the raw value of the voltage sensor, it is used also for the current sensor, that has the following gain values:

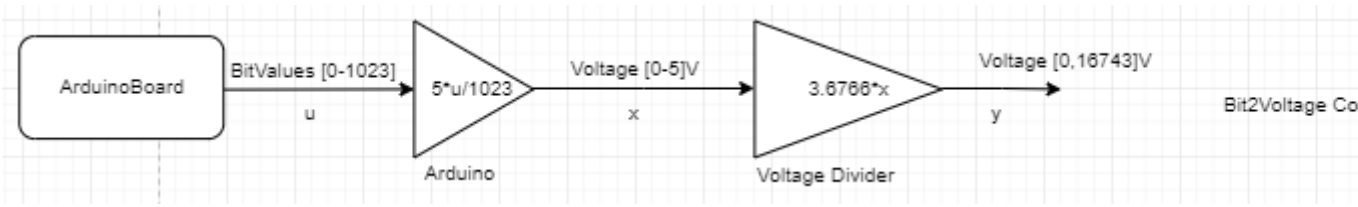

Figure 4.22 – Voltage sensor gain

But, different from the measure of the voltage sensor, this signal is filtered when it is acquired by the board. In fact, the DC motor are a source of noise because of the presence of the commutator and the brushes. For filtering the current, it is used a library, called ResponsiveAnalogRead [13], that provides an acceptable filtering value without increasing too much the complexity of the code, that slows down the system.

To acquire data from the Hall sensor is a bit more complicated. In fact, the hall sensor integrated in the motors produces a square wave. Because of the sensor has only one magnet, at every rotor revolute, the square wave has a rising edge. A counter is used to store how many revolute the motor does. Moreover, the function, that has the task to acquire the data from the Hall sensor, is able to capture the time interval between two consecutive rising edge, by checking the derivate of the square wave signal. From this interval, it can compute the velocity, expressed in RPM, of the motor. Since the mean value of the square wave is variable with the

voltage supply, it is necessary to check the presence of the rising edge using the derivative and not a fixed threshold (at different value of motor voltage supply the threshold changes).

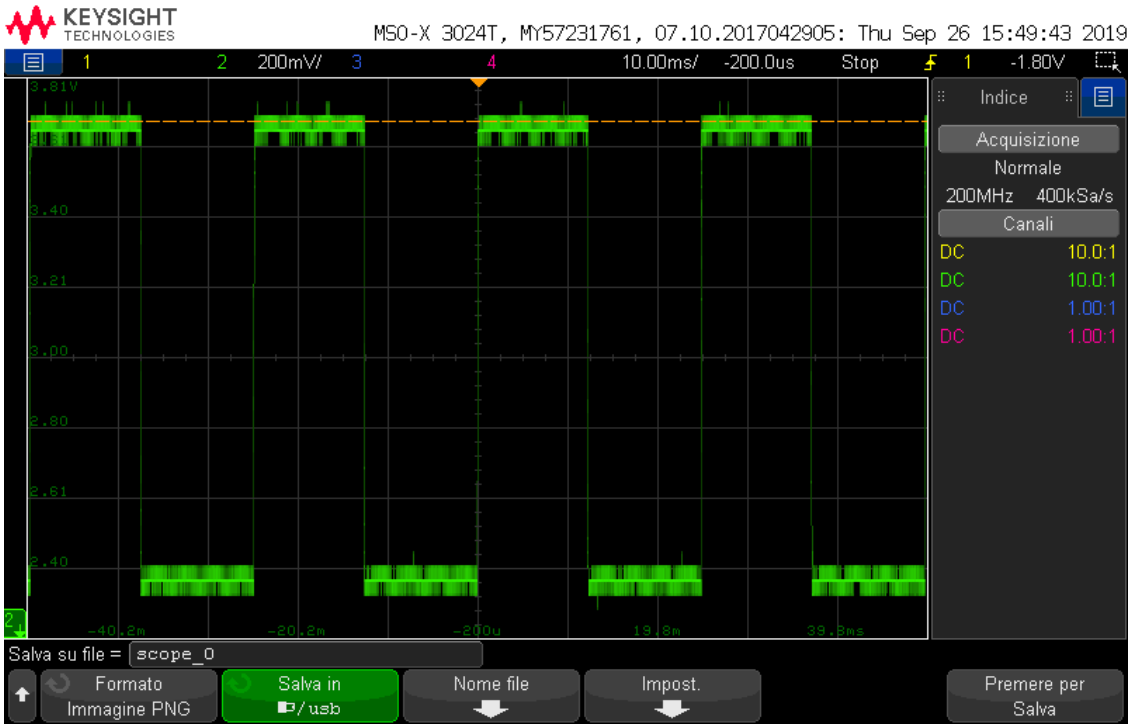

Figure 4.23 – Square wave generated by the Hall sensor

Some of the data measured have more importance than the others. In fact, the measurements required by the control functions must get within a short time period. In particular, that measurements necessary to generate the reference current for the brake motor and to check when the system must switch-off because of a pinch or an end-of-run. These measurements are the absorbed current, the velocity and number of revolute of the main motor. In order to give more or less importance to the function that acquire data from the sensors, different timer structures are implemented: setting a time parameter, every function inside the timer is executed, periodically, only when the time parameter is elapsed. The counter time that compare the elapsed time with the time parameter is reset to zero at every timer

execution.

Measurements optimization

In order to optimize the code, few expedients are applied. Firstly, every measured value is defined not as a *float* data type but as an *unsigned int*, by reducing the order of magnitude (for example converting the current value of 1.5 A into 1500 mA). In fact, mathematical operations done on floating point variable are more computationally complex than the same operations performed on integer values. Secondly, not all the measurements are performed. In fact, because of the two motors are mechanically connected, it is useless to measure, for both, velocity and number of revolute because they are the same values. In such way, it is possible to avoid doing complex and slow computation like floating-point division and product (for getting the derivative) or AnalogRead calls. Moreover, the current absorbed by the brake motor is not measured. In fact, this value is imposed by the code and the value measured is the same with the presence of noise. But the absorbed current measured from the brake motor is not useful for the control function and it is not relevant. Avoiding this measurement, it is possible to not execute an AnalogRead calls and the filtering operation on that signal.

4.3.2 Control

The control function has the tasks to switch-on the system, by giving supply to the two motors, and to switch-off, if the end-of-the-run is reached or a pinch is detected. The control function that power-on the main motor is simply a digitalWrite that close the relay connected to the main motor.
The control function that power-on the brake motor is more complex. Getting the reference current, it multiplies to it a gain value, obtained by measurements and trials, that converts that current in a PWM signal that control the brake motor.

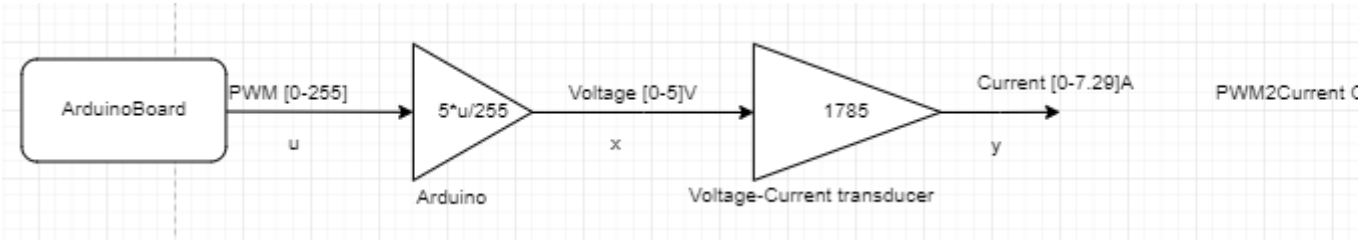

Figure 4.24 – PWM to current gain

The desired current value is obtained by a function that create a current behavior starting from the mathematical model of the seat. In order to compute the correspondence value of current, it is necessary the angular position of the back-seat, that can be easily computed using the number of revolute of the motor.

To the desired current obtained by the function described before, it is added another signal that has the goal to simulate the presence of a pinch. This signal has a behavior that can be chosen by the user by setting four parameters: initial time instant, start value of current, maximum value of current, minimum value of current. As for what described in the section 4.1, about the Simulink model, the pinch starts with a defined current I_{start} after a certain time t_{start} and the it maintains a value between a minimum I_{min} and a maximum I_{max} value.
In the following figure is shown the current behavior that is imposed to Arduino in presence of a pinch.

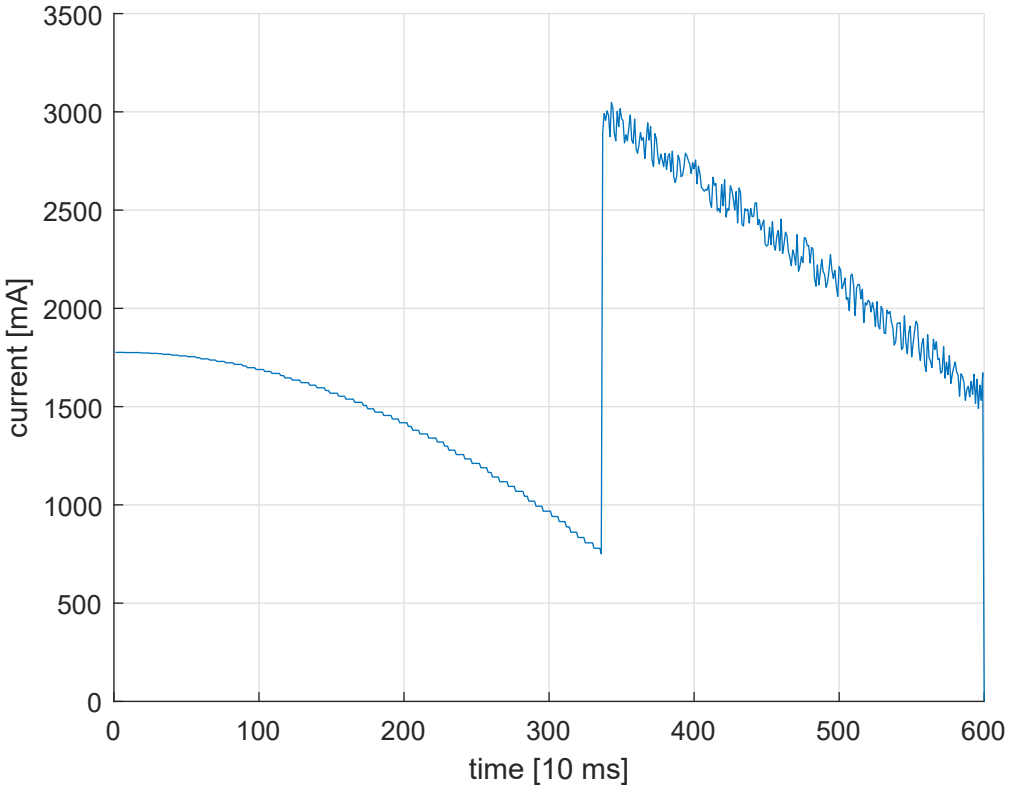

Figure 4.25 – Control current of the brake motor

To switch off the system, Arduino call a function that simply impose to zero the PWM that control the brake motor and turn off the relay connected to the main motor. Furthermore, it stops the while loop of the main part of the code until another run will be performed. This function can be called in two different situations: when the motor reaches its end of run and when the pinch is detected. The end of run is reached when the hypothetical back-seat reaches its final position. It is determined by counting the number of revolute of the motor and converting it to the angular displacement of the back-seat. The conversion value is obtained by choosing a reasonable number. In this case, it is set that 200 revolute of the motor corresponds to an angular displacement of 120° degrees of the back-seat.

The pinch is detected by using the classical method. The function that has this

task, checks the value of absorbed current by the main motor and its velocity. It computes the derivative of the two value and compare these new values with their respective thresholds. If the derivative of the current is greater than a fixed value, means that the main motor increases its effort, so it reports the presence of a possible pinch. If, simultaneously, the deceleration is less than a fixed value, means that the main motor slow down, reporting the presence of the pinch and calling the function that power off the system.
The control to the deceleration is necessary because of the current measurement is not very reliable. In fact, as explained before, the DC motors are a large source of noise. Moreover, during the starting phase, the current is very high, but this is not due to the presence of a pinch. By checking also the deceleration, the algorithm does not confuse the starting phase as a pinch because the motor is accelerating. The choice of the two thresholds are obtained in the following way: First of all, data about the derivative of the current and the derivative of the velocity are collected performing a system run without simulate a pinch.

In the figure below, there is shown the current derivative (the first graph) and the velocity derivative (the second graph) and a possible threshold choice

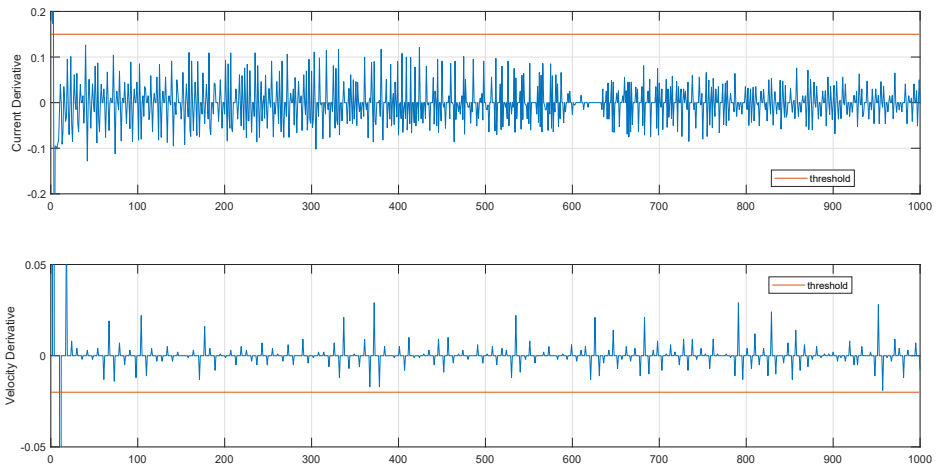

Figure 4.26 – Current derivative and velocity derivative

While the threshold on the velocity derivative must be simply smaller than the minimum value obtained by the data collected, the threshold on the current derivative must be greater than the maximum value but excluding the starting phase, where there are current peaks. Moreover, the current sensing is more afflicted by noise, so it is necessary to perform different tests to obtain the correct value of the threshold. After that, the value obtained are adjusted by performing many tests on different pinch and load condition. The final threshold values are the following:

$Threshold_{I_{der}} = 0.02$	Threshold for current derivative
$Threshold_{V_{der}} = -0.02$	Threshold for velocity derivative

Table 4.6 – Reference Motor parameters

4.3.3 Communication

Arduino hardware has a serial port, also known as UARTs, that can communicate with a personal computer through serial interface, like the USB port. Each serial port supports one Serial Transmit and one Serial Receive block [14]. The function that have the task to communicate with the MATLAB HMI are essentially two: one for receiving data from the HMI, the other for sending. The function that reads the data Matlab HMI sends, is executed at the beginning of the code, at the end of the Arduino setup phase. It remains in a waiting state until all the data are received and only then, it executes the main loop of the code. All the different information are integer number and they are separated by a comma, in order to distinguish them using the primitive function Serial.parseInt(). Using a final control bit, Arduino knows if all the data are received. More precisely, if the last information received has a decimal value of 5, the system stops to read from

the serial buffer. The final bit value is chosen as an integer number that can not be confused with the other parameters because it is too low: in fact, the other parameters, in order to have a reasonable value, are always greater than 50. The function that sends data to Matlab HMI, by writing on the serial buffer, is called periodically, from the motors start until they are turned off. The data sent are basically the measurements collected, such as the velocity, the number of revolute, the current absorbed and the voltage supply. However, another data is sent: the detection of the pinch. It is a binary value that advises if the anti-pinch algorithm has detected the pinch. This function is called, using a timer structure, every 10 milliseconds, that is a reasonable value in order to evaluate the obtaining results and to have sufficiently data to develop a machine learning anti-pinch algorithm.

Communication optimization

The simplest way to write information on serial buffer is to use the function "Serial.print()". But this kind of operation is very slow, because it converts every number digit into the correspondent ASCII value of 1-byte size [15]. Hence, it is important to minimize the quantity of data sent, in order to make faster the code execution. Firstly, to optimize the performance on the measurements, it is not necessary to send velocity and number of revolute for both motors, it is sufficient send information for only one of them (for example the main motor). Furthermore, instead of sending the brake motor absorbed current, it is sent the desired current imposed to the brake motor, in order to have a clean information (i.e. without noise) of the current behavior that simulates the load and the pinch. Secondly, always to minimize the quantity of bytes sent, "Serial.print()" has been substituted with another function called "Serial.write()". This function writes binary data to the serial port. This data is sent as a byte or series of bytes [15]. To convert a number into single bytes in order to be able to use "Serial.write()" function, it is

defined and implemented a communication protocol. Except for the pinch detection, all the other data sent are defined as *unsigned int*, which dimension is 2 bytes. Then, these data are converted into hexadecimal value and split into single byte. All these bytes are sent, and they will be reconstructed by MATLAB HMI. In the following table, the execution time of the methods are compared:

$t_{printData} = 1204\mu s$	Execution time before optimization
$t_{printDataHex} = 104\mu s$	Execution time after optimization

Table 4.7 – Execution time comparison between different serial communication protocol

The execution time is reduced by 91.36% using Serial.write() instead of Serial.print(). All the data are sent in real time.
Storing all the information in arrays could have been a good optimization strategy because, only at the end of the system execution, the data are sent, and it would have saved a lot time avoiding real time serial communication. However, it is not possible to implement this solution on Arduino because it has a low size memory that is not able to store all the variables collected.

4.3.4 Algorithm

The complete functionality of the algorithm is described by the following flow chart:

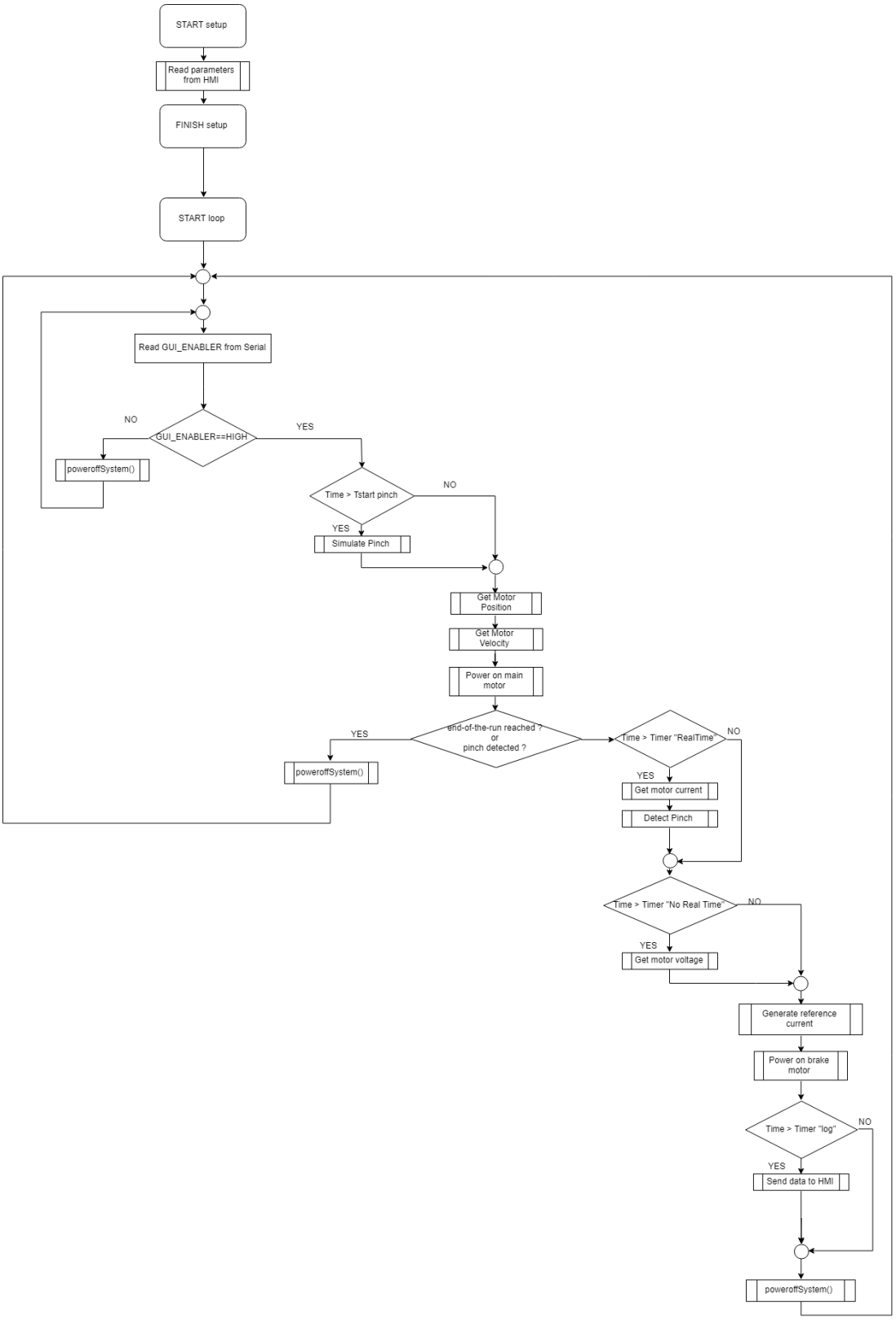
START setup
Read parameters from HMI
FINISH setup
START loop
Read GUI_ENABLER from Serial
GUI_ENABLER==HIGH
NO
YES
poweroffSystem()
Time > Tstart pinch
NO
YES
Simulate Pinch
Get Motor Position
Get Motor Velocity
Power on main motor
end-of-the-run reached ? or pinch detected ?
YES
poweroffSystem()
Time > Timer "RealTime"
NO
YES
Get motor current
Detect Pinch
Time > Timer "No Real Time"
NO
YES
Get motor voltage
Generate reference current
Power on brake motor
Time > Timer "log"
NO
YES
Send data to HMI
poweroffSystem()

Figure 4.27 – Algorithm flow chart

After receiving the data from the serial bus, the main loop starts. At every loop cycle, Hall sensor measures are computed in order to have the right position value to generate the reference current of the brake motor. The absorbed motor current is measured not as fast as possible, but it is regulated by a timer. Because of the anti-pinch algorithm needs the value of the current, it is also inserted in the same timer of the current sense function. If the pinch is detected, at the next loop cycle the system is turned off. The function that sends all the information to the serial bus is executed at the end of the loop, inside a timer, because it is the function with the lowest priority.

4.4 HMI

In order to be able to perform many test changing configuration parameters and to collect data from all these tests, an HMI (Human Machine Interface) is necessary. In particular, a GUI (Graphical User Interface) has been created. A Gui is a kind of HMI that simplify the control of software applications. It is an intuitive interface that avoid the user to learn a language or the commands defined for that specific application. GUIDE is a MATLAB toolbox that allows the developer to create a front ends interface that automate tasks using elements such as buttons, sliders, graphs, menu [30].
The HMI functionality must:

- Send data, through the serial port, the parameters of the test
- Read data, from the serial bus, all the data collected by Arduino and catalog them into a structure
- Show the graph of measurements behavior

- Save data into a comma-separated values (csv) files all the data read
- Turn on and turn off the system
- Connect and disconnect the communication with Arduino

This figure 4.28 shows the HMI graphical interface:

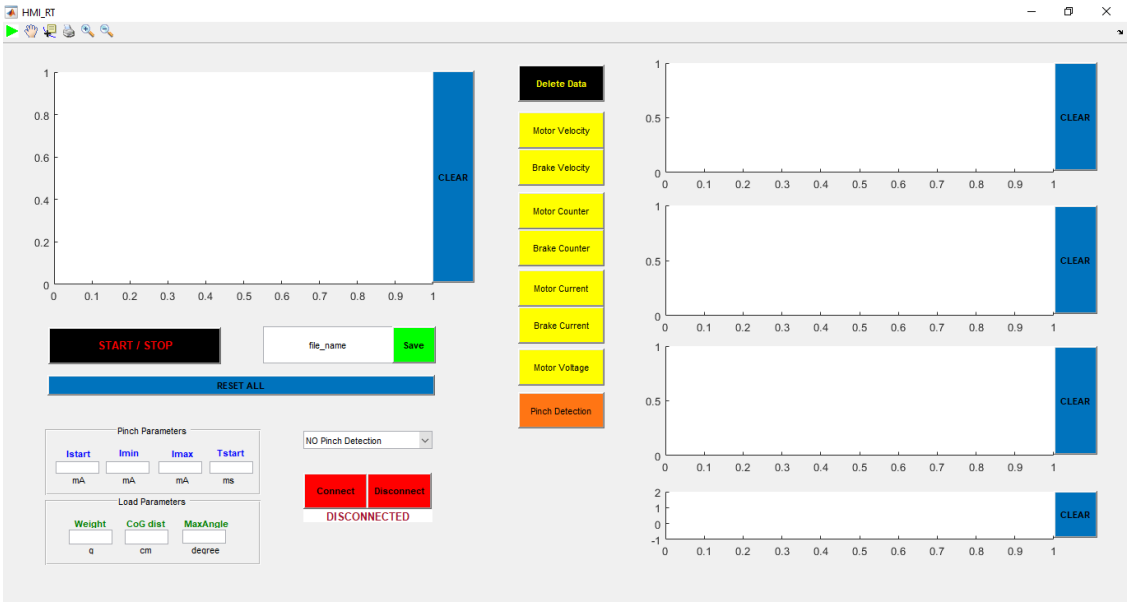

Figure 4.28 – MATLAB HMI

MATLAB HMI works by means of callback function. Every callback function is associated to an element of the GUI. For example, by pushing a button, the code inside its callback function is executed. In the same way, all the other elements work. Obviously, a callback function can be called from another function. So, it is possible to set up only one button to execute all the HMI command instead of pushing all the other buttons.

The START/STOP button is that one that perform a complete test on the bench. The callback function, associated to this button, reads all the value inside the "Pinch parameters", "Load parameters" fields and the value from the selector menu, where it is indicated the anti-pinch method. It establishes a communication channel with Arduino, and it sends all these information to the board. Then, it sends

another signal to run the test bench. During the run, the HMI continuously read the information sent by Arduino from the serial bus by using a function, called ReadSerial.m. When the run has finished, all the data received are plotted on the corresponding graphs. When the system is not in execution phase, it is possible to clear and re-draw the single graph by pressing the corresponding buttons on the GUI. It is also possible to specify a file name and save a csv file, with that file name, that contains all the data collected during the run. If the START/STOP button is pressed again, another run is performed, with the new parameters, if they are changed.

The *ReadSerial.m* function is used to read data from the serial bus and to convert it into the correct decimal values. After it reads the eleven bytes Arduino sends, the function separates the first ten bytes and the last one. in fact, the last one is the binary value that represent the presence of a pinch and it no needs conversion. While the others must be merged and converted into a decimal value. In fact, as it is described in the previous chapter, Arduino sends data split in single byte and in hexadecimal. If all the data, except for the pinch detection information, are zero, it means that the system has been switch off and the ReadSerial function is not anymore called until the next run.

Automated test

To compare results obtained from the different anti-pinch methodology, it is important to create an automated test function, which allows HMI to be automatically executed with a set of different parameters.

Using a Matlab script, the load and pinch parameters are randomly defined but within a certain interval, as follows:

$$Istart \in [1500 - 2500] \tag{4.3}$$

$$tstart \in [1200 - 5000] \tag{4.4}$$

$$DCGload \in [450 - 750] \tag{4.5}$$

$$Mload \in [7000 - 12000] \tag{4.6}$$

$$Aload \in [80 - 120] \tag{4.7}$$

The pinch parameters that corresponds to the maximum and minimum value that pinch current can have, are defined as a function of the starting pinch current.

$$I_{min} = I_{start} - 100 \tag{4.8}$$

$$I_{max} = I_{start} + 100 \tag{4.9}$$

Fifty set of parameters has been generated, and they are used to perform the automated test.

Moreover, finished every run, a CSV file is saved with the name that indicates which algorithm has been used and at which test is referred to.

Name	Values_ 1	Values_ 2	Values_ 3	Values_ 4	Values_ 5	Values_ 6	Values_ 7	Values_ 8	Values_ 9	Values_ 10	Values_ 11	Values_ 12
Motor Velo...	0	0	0	0	0	12	12	12	1613	1613	1613	1613
Motor Cou...	0	0	0	0	0	1	1	1	2	2	2	2
Motor Curr...	232	8953	7279	5686	4957	4471	4228	4633	3823	4498	3958	4228
Motor Volt...	10440	9971	10061	10061	10080	10115	10115	10134	10224	10205	10205	10151
Brake Veloc...	0	0	0	0	0	12	12	12	1613	1613	1613	1613
Brake Coun...	0	0	0	0	0	1	1	1	2	2	2	2
Brake Curre...	2486	2486	2486	2486	2486	2486	2486	2486	2486	2486	2486	2486
Pinch Dete...	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4.29 – Example of a log csv file

Also, a PNG file is created, for every test, in order to show easily a screenshot of the GUI with all the graphs of the measurements and the parameters applied.

4.5 Results

After several tests, the implemented classic anti-pinch algorithm has obtained good results. The main problem encountered during this phase, is the choice of a suitable timer value for measuring the main motor velocity and the main motor absorbed

current. Choosing a timer value too low bring to the increases of the number of operation that the board executed at every loop. In this way, Arduino board, that is not so fast, is not able to measure correctly the velocity that is a variable that anti-pinch classic algorithm must control. Choosing a timer value too high bring to an increase of the anti-pinch algorithm latency. The timer value found that is able to correctly measure the velocity but maintaining the latency low, is 5ms.
An HMI image of an example run is shown in the figure below:

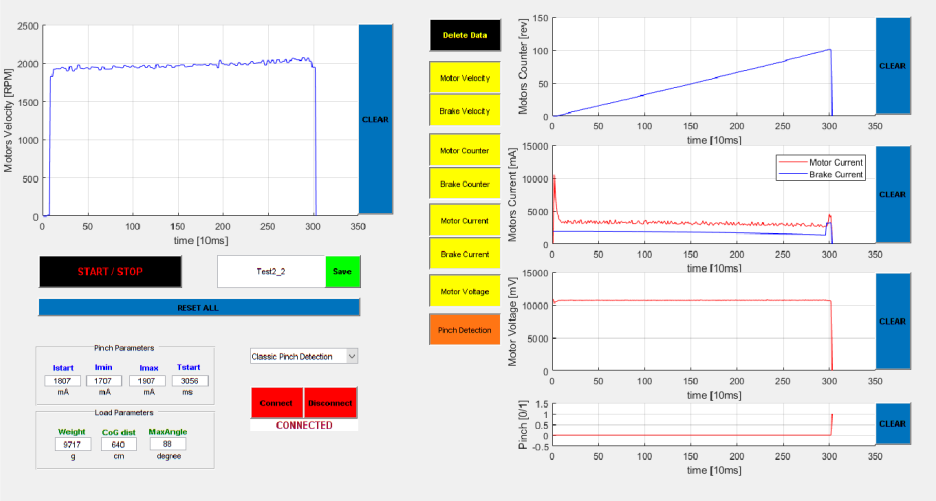

Figure 4.30 – HMI with data taken with a test using classic anti-pinch algorithm

As it is possible to notice:

- The motors velocity increases due to the simultaneous load torque decreasing. At a certain time instant, the pinch is simulated and the load torque sharply increases. The velocity falling down and then it continues to follow the load torque with the contribution of the pinch current. When the system reaches the end-of-the-run, the velocity goes to zero because both motors are turned off.

- The motors counter, that indicates the number of revolute the motors do,

increases as a ramp. Only in the time instant when pinch occurs, there is a little change of the ramp slope.

- The main motor current measured (the red line in the graph), shows the expected behavior, after the peak due to the starting phase. After that, the value decreases because of the load also decrease. When the pinch occurs, there is a peak of the current measures. At the end the absorbed current goes to zero because the system is turned off.

- The brake current shown in the graph (blue line) is not the current absorbed by the brake motor but it is the current that the control board imposed. So, it is an ideal value, not afflicted by noise. It has been chosen to show the desired current behavior instead of the measured one because, in this way, it is possible to see exactly the pinch simulation.

- The main motor voltage graph shows simply the measured supply voltage of the main motor, that it is maintained constant during all the system run by the power supply.

- The Pinch graph shows if the algorithm detects pinch, when the value is one, or not, when the value is zero. As expected, it reports the presence of the pinch a bit after the pinch simulation. This delay is due to the computational time of the algorithm (it must perform derivative operations) and due to the sampling time of the velocity and current measure.

After the execution of 50 experiments, using the random parameters explained in the previous section, give the following results:

- Precision $P = 100\%$. All the simulated pinches are correctly detected by the algorithm, also varying not only the pinch parameters but also the load parameters.

- Recall $R = 100\%$. The algorithm never detects false negatives.

- Accuracy $A = 100\%$. Since the recall and the precision are maximum, also the Accuracy is maximum.
- F_1 score $F_1 = 100\%$. As for the accuracy, also the F_1 score is maximum due to the precision and recall values obtained.
- Maximum latency $L_{max} = 40ms$. The latency is computed by seeing the log file. From when the pinch is simulated to the time the pinch is reported. The latency is quite low because the detection algorithm is not computationally complex, and it also because it is executed as soon as the new current and velocity measures are available.

The result obtained with these tests are very good. However, as explained in the second chapter, external condition such as wearing or different ambient condition, could cause a not perfect functioning of the algorithm. In fact, in presence of these different conditions, the design phase must be repeated, especially for the acceleration and current derivative thresholds.

Chapter 5

Machine learning implementation

The main purpose of this thesis is to compare the classical approach to an anti-pinch problem with an innovative approach such as the application of machine learning. As said in the third chapter, there are lots of different machine learning algorithm. The method used in this project is a supervised classification algorithm called logistic regression. Logistic regression is particularly suitable for an anti-pinch system. In fact, it is a two-class classification method, the two classes are: "presence of pinch" and "absence of pinch". Moreover, it is a fast algorithm and simple to implement, compared to the others. These characteristics make the logistic regression an appropriate choice to implement on a low power board such as Arduino Nano. In order to create a general method to implement the machine learning anti-pinch algorithm, the data-set is not taken from the measurements performed on the test-bench, but they are taken from the values obtained from the Simulink model simulation. In this way, changing the hardware component, such as the motors, could not affect the reliability of the algorithm.

5.1 Prepare dataset

A classification algorithm dataset is composed by the features and by the output variables. In the case of anti-pinch system, the output variables are represented by a binary value the indicates if pinch is occurred or not. The features are represented by the main motor absorbed current values, and they are also the variables that machine learning algorithm must continuously check to detect pinch.
To implement the logistic regression algorithm, it is necessary to prepare a dataset which contains the information that are important for the scope of the algorithm. For the aim of the project, the dataset contains different windows of many current behavior, each of them with different pinch conditions. Firstly, to obtain the dataset, the main motor current behavior, without the presence of the pinch, is taken performing a simulation using the Simulink model. To better represents the real current, it is added noise to the simulation measure.

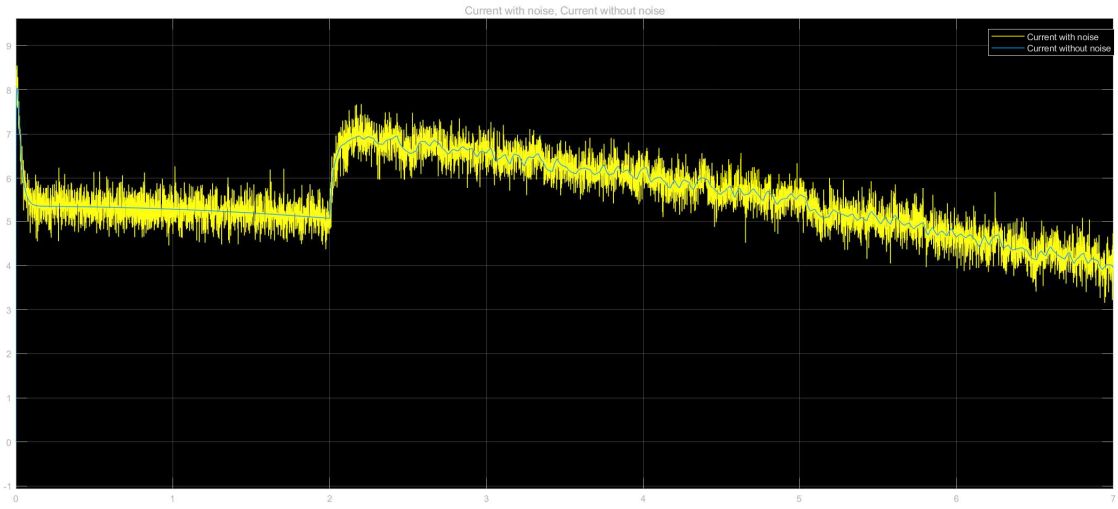

Figure 5.1 – current simulated with noise

The load parameters of the simulation are:

- $M_{load} = 10000g$

- $d_{CoG} = 700mm$

- $\alpha_{max} = 120°$

The current signal has a sampling time of 10 milliseconds and the entire simulation last after 7 seconds, so the number of samples are 7000. After that, many other simulations are performed, varying only the pinch parameters. The following table represent the different model configuration used to collect the current data:

	Name	Values_1	Values_2	Values_3	Values_4	Values_5	Values_6	Values_7	Values_8	Values_9	Values_10	Values_11	Values_12	Values_13	Values_14	Values_15
2	Ipinch start	2188	1807	1567	1705	2189	2123	1602	1840	1930	1501	2428	2011	2312	1728	2435
3	Tpinch_start	2968	3056	4043	1438	2160	2057	1745	1801	3700	2320	4201	2232	3550	1761	1532
4	Maximum Angle	87	88	110	93	108	90	99	107	84	84	89	96	117	91	120
5	CoG distance	694	640	627	598	647	651	629	743	564	550	552	724	699	743	619
6	Load Mass	8947	9717	9690	9265	10007	9081	11201	10184	10643	9253	8552	8638	9926	10609	10685

Figure 5.2 – 15 example parameters

Now the current matrix contains 232 different system runs with 7000 samples. At every simulation, another vector is stored in memory: the vector of pinch. This vector contains zeros except when pinch occurs, in that moment, it has a value equal to one. If pinch is not present during the simulation, the relative output error contains all zeros.

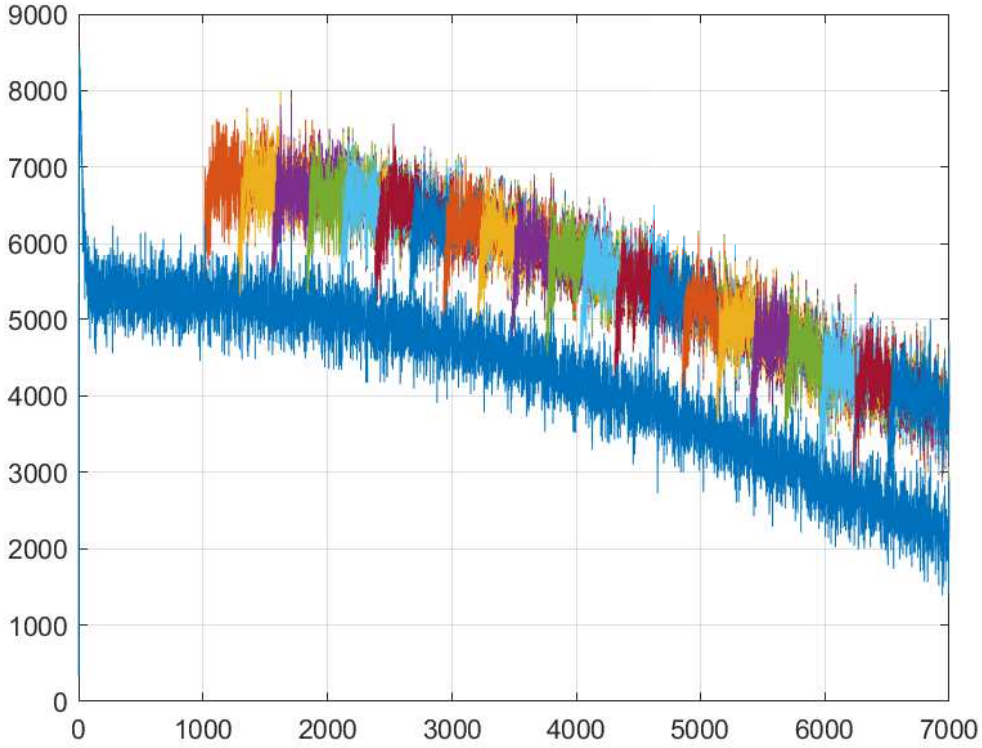

Figure 5.3 – All current runs for training

All the output vectors are concatenated into a matrix with the same dimension of the dataset.

Then, all the runs must be divided into windows, not overlapped, of n features, obtaining the input matrix of the dataset. It has been chosen different value of n in order to test all of them on the test bench.

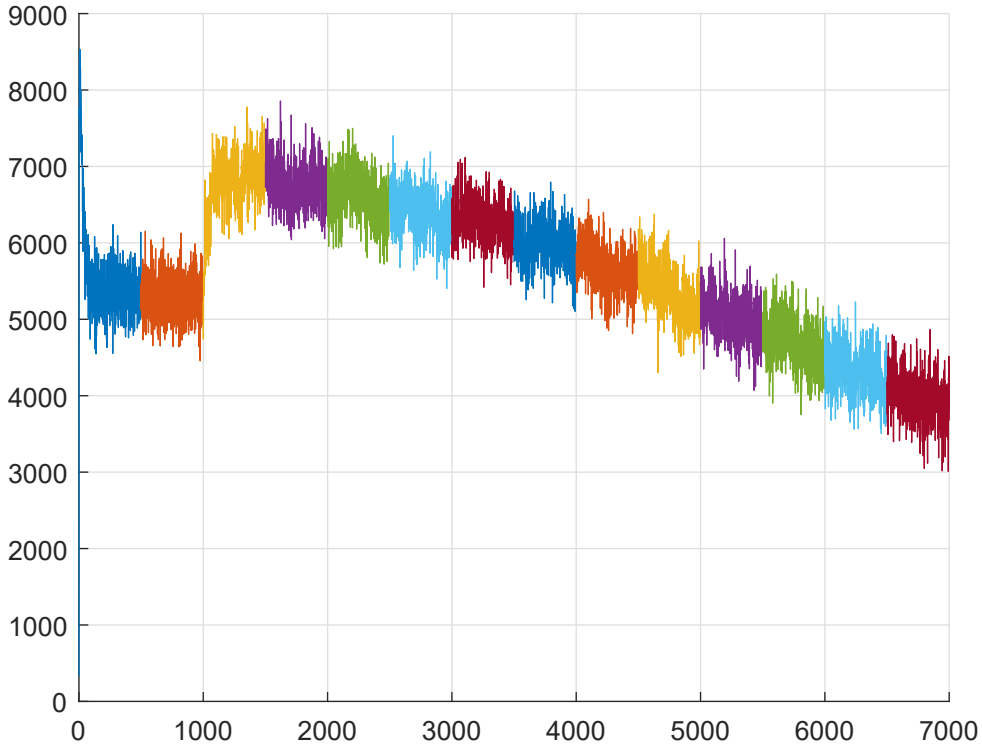

Figure 5.4 – Current divided into windows

Pinch matrix must be converted into a vector with the same length of the number of examples obtained by dividing into windows the current matrix. In fact, every window of the input must correspond to a single output value. The output value can be

- 1, if in the corresponding window, there is a pinch
- 0, if in the corresponding windows, there is not a pinch

Finally, all the examples of the dataset are shuffled in order to not have a correlation between consecutive examples. The dataset, composed by the input matrix and the output vector, is completed and the training, cross-validation and test-error are performed.

5.2 Training

The objective of the training is to find the n values of θ that minimize the cost function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)}log(h_\theta(x^{(i)})) - (1-y^{(i)})log(1-h_\theta(x^{(i)}))] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2 \quad (5.1)$$

The θ values are the parameters of the linear model

$$z = \theta^T x \quad (5.2)$$

in the logistic function

$$h_\theta(z) = \frac{1}{1+e^{-z}} \quad (5.3)$$

The dataset used for the training phase is not the entire dataset but only the 60% of it. The trainining is performed using a MATLAB function, called *fminunc*. "fminunc" function finds minimum of unconstrained multivariable function, that is the logistic regression cost function. It is a nonlinear programming solver and it returns the values of the theta vector.

θ has dimension equal to the number of features plus one. In fact, the first term is the intercept term θ_0.

To evaluate the results obtained, test and validation are performed. The 40% of the dataset not used for training, are divided into two equal parts. The test error procedure is done on one group while the cross validation on the other.

The results are very good also with a small number of features, as it illustrated in the table 5.1.

Number of features	Test Accuracy	Precision	Recall	F_1 score
$n = 20$	$A = 99.68\%$	$P = 88.97\%$	$R = 37.54\%$	$F_1 = 52.80\%$
$n = 70$	$A = 100\%$	$P = 98.46\%$	$R = 98.84\%$	$F_1 = 9.65\%$
$n = 100$	$A = 99.97\%$	$P = 99.57\%$	$R = 100\%$	$F_1 = 99.78\%$
$n = 200$	$A = 100\%$	$P = 100\%$	$R = 100\%$	$F_1 = 100\%$

Table 5.1 – Training and test results, changing the number of features

On the real test bench, the results will be probably different because of the non-idealities that does not afflict the Simulink model.

5.3 Arduino implementation

After founding different set of θ vector, the machine learning control algorithm is implemented on Arduino board. The machine learning algorithm computes the logistic function and perform a prediction about the result. To compute the logistic function, the linear model must be computed before. By means of a *for cycle*, the values of θ are multiplied with the current value inside the window considered. At the result, it is added the intercept term. Then the logistic function is computed.

If the logistic function has a value greater or equal a threshold, fixed to 0.5, the presence of the pinch is reported, and the system is turned off. While, if the logistic function is lower than the threshold, no pinch is reported.

Machine learning algorithm optimization

Due to the presence of a for cycle that must multiply a lot of parameters, that corresponds to the number of features, the machine learning anti-pinch algorithm is quite slow. Moreover, also the computation of the sigmoid function, that is a division between float numbers plus an exponential operation, has a slow execution.

The table 5.2 shows how much time the machine learning anti-pinch algorithm take on, varying the number of features:

Number of features	Linear model	Sigmoid function	Total algorithm
$n = 20$	$t_{lm} = 388\mu s$	$t_{sig} = 217\mu s$	$t_{tot} = 606\mu s$
$n = 50$	$t_{lm} = 1029\mu s$	$t_{sig} = 217\mu s$	$t_{tot} = 606\mu s$
$n = 70$	$t_{lm} = 1387\mu s$	$t_{sig} = 217\mu s$	$t_{tot} = 606\mu s$
$n = 100$	$t_{lm} = 2092\mu s$	$t_{sig} = 217\mu s$	$t_{tot} = 606\mu s$
$n = 200$	$t_{lm} = 4128\mu s$	$t_{sig} = 217\mu s$	$t_{tot} = 606\mu s$

Table 5.2 – Computing execution time of the machine learning algorithm

For the computation of the linear model nothing can be done in order to optimize it, while the sigmoid function can be optimized by substituting it with another function that does not include exponential operator, but it maintains the same properties. The new function is the following:

$$h_\theta(x) = \frac{1}{2} \cdot \frac{x}{1 + |x|} + \frac{1}{2}; \tag{5.4}$$

As it is shown in the figure below, the properties, that are important for machine learning applications, are maintained. In fact, with the same random vector, it is possible to see their similarity in the following graph:

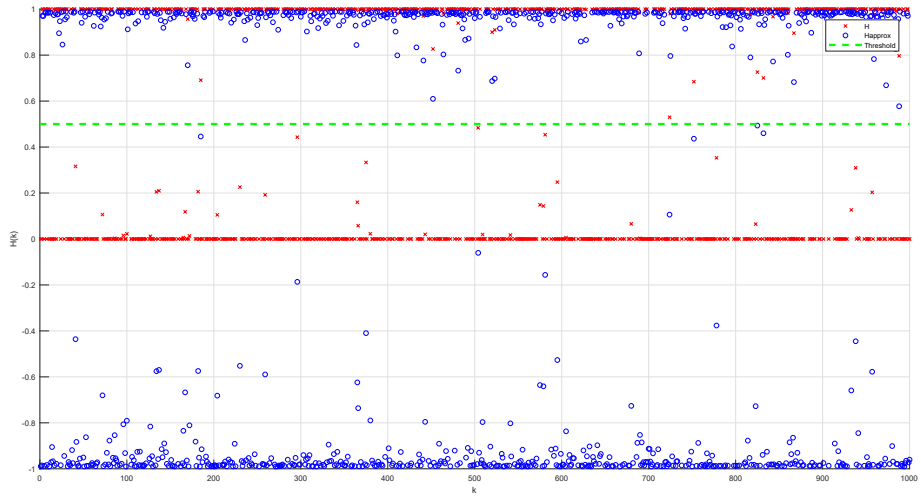

Figure 5.5 – Comparison between sigmoid and its approximation using a random vector

The number of values that are greater than 0.5 are almost the same for both functions: the sigmoid function has 506 points out of 1000 that are greater than 0.5, and also the approximated function has 506 point out of 1000. Also changing the threshold, which must be within the interval [0, 1], the results are very similar.

Thanks to this optimization, the computation of the approximated function allows to save the 79.26% of the time respect the real sigmoid function, as it is possible to see in the table below.

Sigmoid	Approximation
$t = 217\mu s$	$t = 45\mu s$

Table 5.3 – Computing execution time comparison

Hence, the total execution time of the machine learning algorithm is reduced as follows:

Number of features	Sigmoid execution time	Approximated function execution time
$n = 20$	$t_{sig} = 606\mu s$	$t_{app} = 434\mu s$
$n = 50$	$t_{sig} = 1238\mu s$	$t_{app} = 1067\mu s$
$n = 70$	$t_{sig} = 1563\mu s$	$t_{app} = 1419\mu s$
$n = 100$	$t_{sig} = 2287\mu s$	$t_{app} = 2116\mu s$
$n = 200$	$t_{sig} = 4288\mu s$	$t_{app} = 4161\mu s$

Table 5.4 – Computing execution time of machine learning algorithm comparison

To create the current buffer, necessary for the computation of the linear model, two different methods are used: the first one consists to consider not overlapped windows of absorbed current, the second one to consider overlapped windows. After the implementation descriptions, advantages and disadvantages of the two methods are described.

Not Overlapped Windows

In the not overlapped windows method, a buffer of fixed dimension n, that corresponds to the number of features, is defined. Whenever Arduino collect the measure about the main-motor absorbed current, that value is fill in the buffer. When the buffer is full, the machine learning anti-pinch algorithm is executed. Then, the cycle is repeated and only after other n elements are fill in the buffer, the anti-pinch algorithm repeats its execution.

Overlapped Windows

In the overlapped windows method, a "First In First Out" (FIFO) buffer, or circular buffer, is defined. The FIFO buffer, and the operations associate to it, is implemented by integrating the library "CircularBuffer.h". This library allows to define the dimension of the buffer and to insert the element in the first empty position

through the function “push”. If the buffer is full, that can be checked through the function “isFull”, and an element must be inserted into the buffer, the first element of the queue is deleted and the new element is added as last element, maintaining the buffer dimension constant [16]. Only when the buffer is full, the anti-pinch algorithm is executed.

The first method is very simple to implement, and it calls the anti-pinch algorithm less time respect to the second method. In fact, only every “n” measurement, it checks the presence of the pinch. The drawback is due to its reliability: if the pinch occurs near the start, or near the end, of the current window, pinch is not detected.

The second method is a bit more complicated to implement. It calls the anti-pinch algorithm much more times than the other. In fact, once the buffer is full, it checks the presence of the pinch at every new measure, slowing down all the code execution. The great advantage is due to its reliability, and it does not suffer the problem that the first method has.

To obtain a trade-off between the two methods, it is added a new parameter “WINDOWS_SHIFT” that, using the Overlapped windows method, call the anti-pinch algorithm not at every new current measurement, but at every m measurements. If “WINDOWS_SHIFT” is equal to the number of features, the overlapped windows method corresponds to the not overlapped method.

5.4 Results

Some tests are performed for both machine learning anti-pinch algorithms in order to

1. choose the number of features that returned the best results

2. compare the two approaches.

During the training phase, good results are obtained also with a small number of features: with 50 features, a 99.89% of precision is reached. But the training has been executed on a current behavior taken from the simulation model. In fact, in the real system, the results are not so good. to reach a good level of precision and accuracy, the number of features is set to 200. Tests with 70, 100, 150 number of features are tried with not acceptable results (precision always less than 60%).

Moreover, by testing, it is highlighted that the execution time is hardly dependent on number of features used. In particular, the measurements of the velocity are not always correct because the machine learning algorithm slows down the execution of the main loop and retards the measure of the hall sensor, that miss some rising edge of the square wave. In fact, as it is possible to see in the figure below, in some points the velocity error is about the half of the real value.

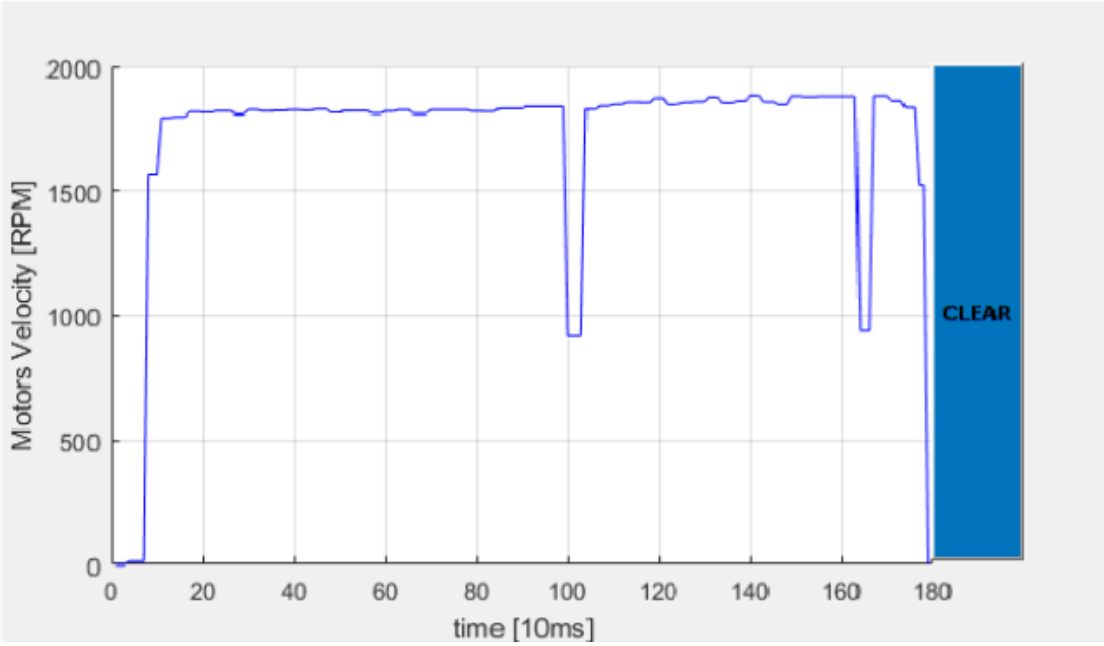

Figure 5.6 – Velocity measurements with wrong values

This is not a great problem because, in the anti-pinch algorithm implemented with the machine learning approach, the value of the velocity is never used.

Another important parameter to set up is the value of the timer that runs the current measurement and the pinch detection algorithm. the timeout value can be chosen in two ways:

1. Low value, such as $700\mu s$. In this way the algorithm works well only when sample windows are not overlapped because the buffer is not always full, and the machine learning algorithm is executed less times. The hall sensor measurements are good, the latency is good, but, as explained before, this method is not so reliable. Instead, using the overlapped sample windows method, the anti-pinch algorithm is executed at every loop cycle, once the buffer is full and Arduino Nano does not have the ability to perform correctly all the operations requested. In this case, all the measures are no correct, included the motor current, which is a necessary value to detect the presence of the pinch.

2. High value, such as $6000\mu s$. In this way, the method that uses overlapped sample windows, works with a high reliability, but the latency is increased because of the current measurement occurs every $6000\mu s$, that is the minimum value that the latency can have. Using the not overlapped sample windows method, the algorithm has too high latency. In fact, to completely fill a buffer of 200 elements, 1.2 seconds are necessary. This value is not acceptable for a safety application like anti-pinch.

From this consideration, the timer timeout has a different value in accordance with the kind of buffer used. An HMI image of an example run is shown in the figure below:

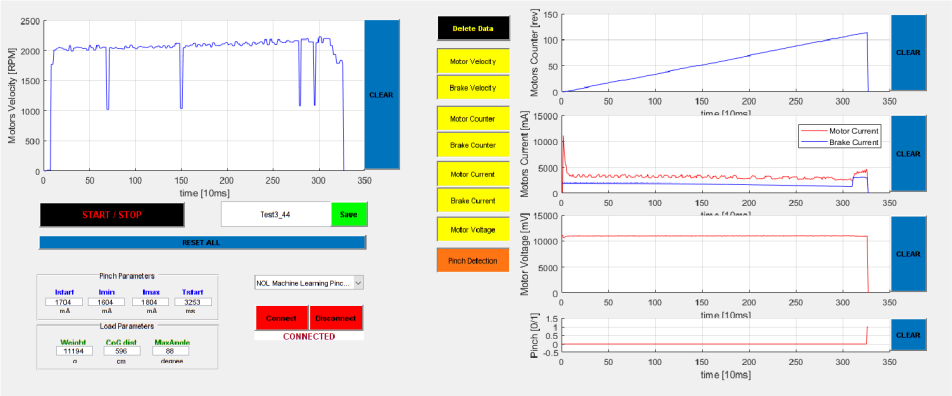

Figure 5.7 – Results using machine learning algorithm with no-overlapped windows

In this figure, the first method is used (not overlapped windows) with 200 number of features. The general behavior of the system is the same obtained with the classic anti-pinch implementation, as explain in chapter 4.5. The difference that can be notice are only few.

- The motors velocity graphs mark the errors have on the measure of the square wave generated by the hall sensor. It possible to see that, at some time instant, the velocity measure drops about an half of its real value.

- Also in the motors counter graph, it is possible to notice the wrong the Hall sensor. In fact, there are few slopes changing in correspondence of velocity decrease.

- The Pinch graph shows if the algorithm detects pinch. Now, the presence of the pinch is reported as a window, because it is restored to zero after a new set of 200 samples are fill in the buffer.

After the execution of the 50 experiments, using the always the same parameters used for testing the classic approach, the following results are obtained:

- Precision P = 80%. Not all the pinches are detected, probably due to the fact that the simulated pinch was in correspondence of the board of sample window.
- Recall R = 100%. The algorithm never detects false negatives.
- Accuracy A = 99%. Accuracy is quite high only due to the absence of false negatives, but this in not the most important parameter to evaluate the algorithm.
- F_1 score $F_1 = 88.90\%$.
- Maximum latency $L_{max} = 70ms$. The latency is not so high because the sampling time of current measure is very low.

In the following test, the second machine learning method is used, always with 200 number of features.

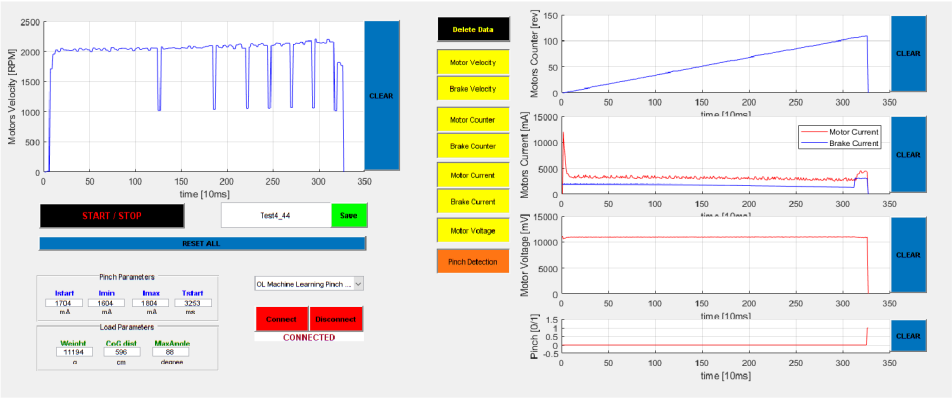

Figure 5.8 – Results using machine learning algorithm with overlapped windows

The general behavior of the system is the same as explain for the not overlapped windows method. The differences that can be notice are only on the measure from the Hall sensor.

1. The number of errors on the velocity measure is a bit higher, because the anti-pinch algorithm is executed more times than before.

Also there, the last graph represents the the pinch not as an impulse but as a window. In fact, the algorithm detects many times the pinch, since it is in the buffer controlled for many loop cycles.

After the execution of the 50 experiments, using the always the same parameters used for testing the classic approach, the following results are obtained

- Precision P = 100%. With this method, all the pinches are detected.
- Recall R = 100%. The algorithm never detects false negatives.
- Accuracy A = 100%. Accuracy is maximum.
- F_1 score $F_1 = 100\%$. F_1 score is also maximum
- Maximum Latency $L_{max} = 170ms$. The latency is bit higher than the previous tested method because the sampling time of current measure is higher ($6000\mu s$ versus $700\mu s$).

To obtain a latency value as low as possible, and to not decrease the precision, the anti-pinch control is executed ever new 20 current measures. In the following table the results comparison is summarized.

Method	Accuracy	Precision	Recall	$F_1 score$	$Latency_{max}$
NOL	99%	80%	100%	88.90%	$70ms$
OL	100%	100%	100%	100%	$170ms$
$\Delta_{\%}$	+1.01%	+25%	0%	+12.49%	+142.86%

Table 5.5 – Comparison between not overlapped (NOL) windows method and overlapped (OL) windows method

From the obtained results, the second method is more reliable than the first one, and it is used for the comparison with the classic anti-pinch algorithm. In fact, the precision, that is the most important characteristic for the project, is maximum. Although the latency is higher than before, it is however acceptable for the application.

Chapter 6

Conclusion and future implementations

Performing the same fifty run, the classic anti-pinch method is compared with the anti-pinch algorithm obtained using the machine learning method. The test parameters are chosen in a random way, as explained in the section 4.4. The saved data of all the runs allow to evaluate the performance and the precision of the two methods.
The selected machine learning algorithm is that one that uses the overlapped sample current windows as input. In fact, this method was already evaluated as more reliable and without significant drawbacks respect to the other (section 5.4).
The results obtained for the classical pinch detection, are summarized in the table 6.1.

The data are the same described in the section 4.5, for classic anti-pinch algorithm, and in the section 5.4, for machine learning anti-pinch implementation. As it is possible to see from the test executed, the results are very good in both cases, even though both methods have their disadvantages.

The classic anti-pinch works very well, it always detects the pinch and it has a low value of latency. However, it needs two kind of measure to work correctly (motor absorbed current and motor angular velocity). Moreover, the setup of the algorithm is different from every kind of motor and is dependent on external factors, like ambient condition, and requires a lot of run to obtain good results. Whereas, using the machine learning approach, the results are anyway very good. Only the latency is a bit higher due to the facts that the algorithm needs to have, not only the single value, but a set of current values. The other drawbacks found, such as the wrong measure of the motor velocity, are not so relevant because they happen sporadically, and they are not necessary for the anti-pinch correct functioning.

Method	Accuracy	Precision	Recall	$F_1 score$	$Latency_{max}$
CL	100%	100%	100%	100%	$40ms$
ML	100%	100%	100%	100%	$170ms$
$\Delta_{\%}$	0%	0%	0%	0%	+325%

Table 6.1 – Comparison between classic anti-pinch algorithm (CL) and machine learning anti-pinch (ML) algorithm

The figure belows shows the differences between velocity and current measurements using the different methods. In the last graph, the latency of pinch detection could be noticed.

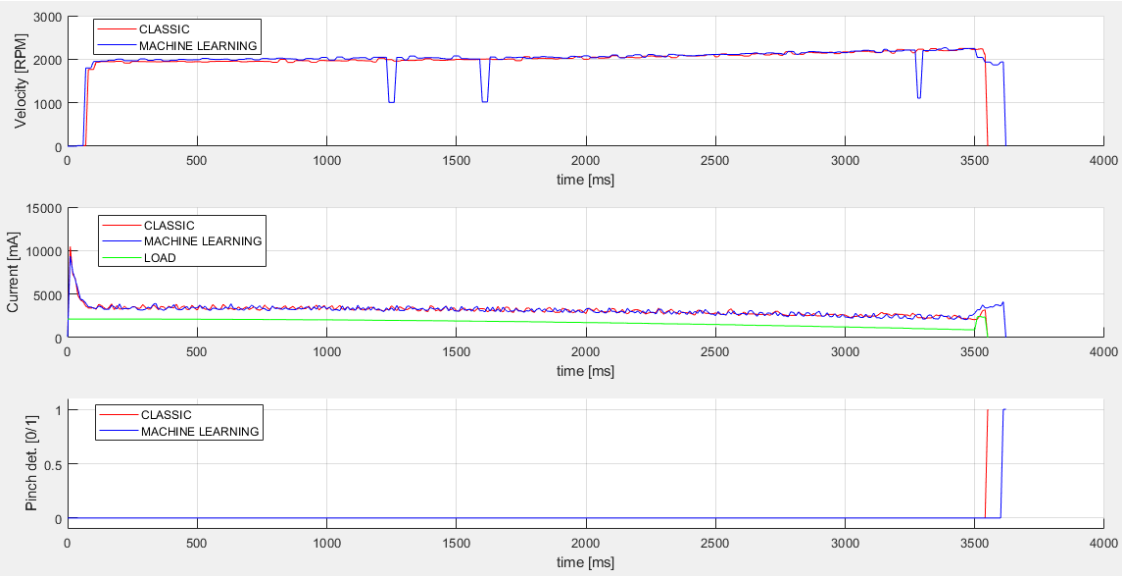

Figure 6.1 – Test results of the two algorithms

The great advantage that this innovative strategy has, respect to the classical one, is the simplicity of its integration also with new systems. In fact, the dataset collected are not taken from the real motor but from the simulation. Designing a good simulation model of a system, and taking the motor absorbed current data from it, is much faster than performing a lot of runs on a real motor, trying to test all the different situations. Moreover, the same training, validation and test algorithms of the logistic regression, are valid to every kind of problem. Thus, it is not necessary to develop a new algorithm to find the parameters of the logistic function.

Many different improvements of the system can be implemented. Future work concerns deeper analysis of the machine learning algorithm, new proposals to try different methods, system improvements by changing the control board with one faster or use two motors that act as brake, simulating the load. It could be interesting to find a way to reduce the number of features, in order to decrease the latency of the machine learning algorithm. For instance, using as input vector of the logistic regression model, formed by not only the current but also the velocity of the motor,

could improve the precision of the anti-pinch algorithm also with a small number of features. To try this implementation, it is necessary that the velocity measurement is correct, hence, it is a more powerful control board able to execute all the tasks is needed. Changing the behavior of pinch, could be another interesting improvement. In this project, the presence of an obstacle is simulated by using a step function. In the future the pinch could have another kind of waveform, such as a ramp or a sinusoid, and checking the reliability of the machine learning anti-pinch algorithm, compared to the classical one, could be interesting. Also the test-bench can be changed in order to obtain a better model of a real application system. For instance, two motors instead of one, can be used to simulate two adjacent seats, which may influence their behavior. In this way, it is possible to simulate a more realistic situation, that are impossible to try with only one motor.

Bibliography

[1] G. S. Buja R. Menis and M. I. Valla "Disturbance Torque Estimation in a Sensorless DC Drive" IEEE Trans. Industrial Electronics vol. 42 no. 4 1995 pp. 351-357.

[2] Pak, J.M., Kang, S.J., Pae, D.S. et al. Int. J. Control Autom. Syst. (2017) 15: 2443. https://doi.org/10.1007/s12555-016-0328-8

[3] I.Knight, A.Eaton, and D. Whitehead, "The reliability of electronically controlled systems on vehicles", TRL limited, Porject Report PR/SE/101/00, Tech.Rep., 2001

[4] W. S. Ra, H. J. Lee, J. B. Park, and T. S. Yoon, “Practical pinch detection algorithm for smart automotive power window control systems,” IEEE Trans. on Ind. Electron., vol. 55, no. 3, pp. 1376–1384, Mar. 2008.

[5] AVR480: Anti-Pinch System for Power Window using tinyAVR and megaAVR devices. Available at https://www.microchip.com/wwwAppNotes/AppNotes.aspx?appnote=en591446

[6] M. G. Kliffken, H. Becker, H. Lamm, H. Prüsell, and J. Wolf, “Obstacle detection for power-operated windowlift and sunroof acutation systems,” SAE Tech. Paper Series, 2001-01-0466, Tech. Rep., 2001.

[7] R. P. Gerbertz, “Method of compensating for abrupt load changes in an anti-pinch window control system,” Patent US 6 573 677 B2, Jun. 3, 2003.

[8] B. R. Wrenbeck, J. T. Kelley, and P. A. Perez, “Adaptive window lift control with pinch force based on object rigidity and window position,” Patent US 5

436 539 A, Jul. 25, 1995.

[9] A. L. Samuel, "Some studies in machine learning using the game of checkers," in IBM Journal of Research and Development, vol. 44, no. 1.2, pp. 206-226, Jan. 2000.

[10] Thomas M. Mitchell. 1997. Machine Learning (1 ed.). McGraw-Hill, Inc., New York, NY, USA.

[11] An introduction to Machine Learning, Siddharth Pandey. Available at https://www.geeksforgeeks.org/introduction-machine-learning/

[12] Introduction to Logistic Regression, Ayush Pant, Towards Data Science. Available at https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

[13] ResponsiveAnalogRead, Damien Clarke. Available at https://github.com/dxinteractive/ResponsiveAnalogRead

[14] Use Serial Communications with Arduino Hardware, © 1994-2019 The MathWorks, Inc. Available at https://www.mathworks.com/help/supportpkg/arduino/ug/use-serial-communications-with-arduino-hardware.html

[15] Language Reference, © 2019 Arduino. Available at https://www.arduino.cc/reference/en/language/functions/communication/serial/print/

[16] CircularBuffer library, Roberto Lo Giacco. Available at https://github.com/rlogiacco/CircularBuffer

[17] Andrew Ng, Machine Learning by Stanford University, Available at https://www.coursera.org/learn/machine-learning/home/welcome

[18] Hye-Jin Lee, Won-Sang Ra, Tae-Sung Yoon and Jin Park. (2005). Practical pinch torque detection algorithmm for anti-pinch window control system application.

[19] Robert P. Gerbetz "Method of Compensating for Abrupt Load Changes in an Anti-Pinch Window Control System" US Patent US2002/0190680 Al 2002.

[20] X. de Frutos "Anti-Pinch Window Control Drive Circuit" US Patent US2003/0137265 A1 2003.

[21] N. Syed-Ahmad and F. M. Wells "Torque Estimation and Compensation for Speed Control of a DC Motor Using an Adaptive Approach" Proceedings of the 36 Midwest Symposium on Circuits and Systems 1993 vol. 1 pp. 68-71.

[22] John Paul Mueller, Luca Massaron, 3 Types of Machine Learning. Available at https://www.dummies.com/programming/big-data/data-science/3-types-machine-learning/

[23] Hye-Jin Lee, Won-Sang Ra, Tae-Sung Yoon and Jin-Bae Park, "Robust pinch estimation and detection algorithm for low-cost anti-pinch window control systems," 31st Annual Conference of IEEE Industrial Electronics Society, 2005. IECON 2005., Raleigh, NC, 2005, pp. 6 pp.-.

[24] H. W. Kim and S. K. Sul "A New Motor Speed Estimator Using Kalman Filter in Low-Speed Range" IEEE Trans. Industrial Electronics vol. 43 no. 4 1996 pp. 498-504.

[25] F. Viggiano, S. Bellagarda, "Anti-Pinch Emulator Plan", EMA s.r.l.

[26] Product description. Available at http://www.gravitech.us/arna30wiatp.html

[27] Overview and tech specs. Available at https://store.arduino.cc/arduino-nano

[28] Mitchell, Tom. (2006). The Discipline of Machine Learning.

[29] DC motors, Wikipedia contributors, "Wikipedia, The Free Encyclopedia". Available at https://en.wikipedia.org/wiki/DC_motor

[30] matlab-gui, © 1994-2019 The MathWorks, Inc. Available at https://www.mathworks.com/discovery/matlab-gui.html