System description

The system is composed by:

* A DC direct drive brushed motor
* Three carts
* Three springs
* Several weights
* Three encoders for the position
* An encoder for the motor
* A PoliArd board

The mechanism consists of up to three mass carriages interconnected by bi-directional springs. The mass carriage suspension is an anti-friction ball bearing type with approximately ± 3 cm of available travel. The linear drive is comprised of a gear rack suspended on an anti-friction carriage and pinion coupled to the motor shaft. Optical encoders measure the mass carriage positions.

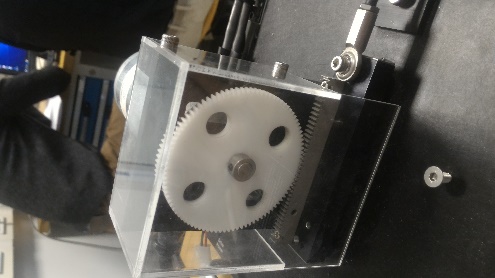
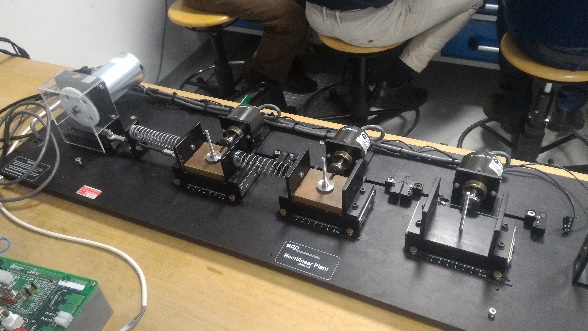
Three springs can be attached between the carts or between the first cart and the base plate and the mass of all the carts can be adjusted using the weights provided (500 ± 5 g each).

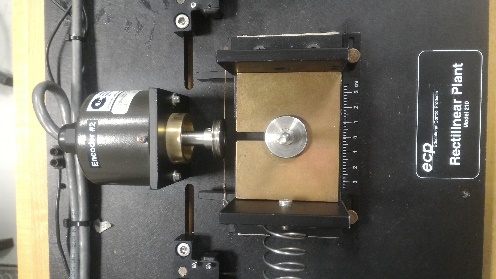
The encoder type is CP-850-4000-ECP: it is an optical incremental digital rotary shaft encoder. The number 4000 indicates the line counter (i.e. the number of equally spaced radial lines per 360 mechanical degrees on the incremental encoder code disk). A low power light source is used to generate two 90 degrees out of phase sinusoidal signals on the detectors as the moving plate rotates with respect to the stationary plate. The moving plate rotates by means of an iron string wound up in it and then attached to the cart. The position is measured by calculating the angular displacement of the disk.

The motor encoder is used to transmit both voltage and current to the Arduino board.

The 24 Volt DC motor has a case diameter of 63 mm and 194 Watts of output power. It has a nominal speed of 3700 rpm and a nominal Torque of 270 mN⋅m. The DC motor has nominally a resistance of 0.7 Ω and an inductance of 1.05 mH.

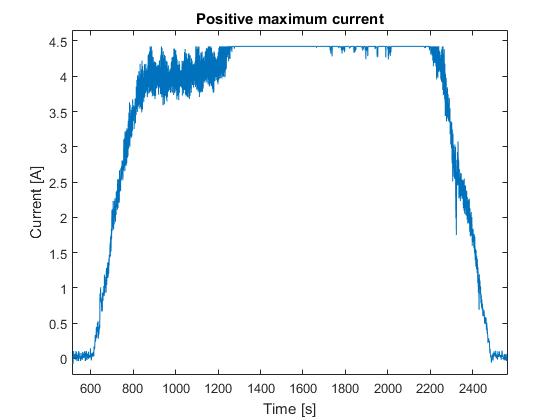
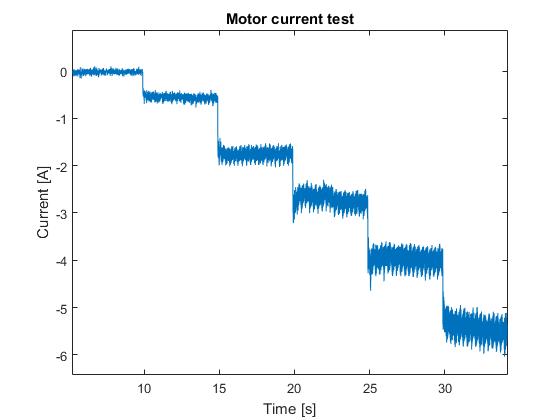
The poliArd board mounts an Arduino Due board. The board has a 24 V power supply and it’s capable of measuring up to 5 A current. It has two motor drivers and four encoder interfaces. The board is programmable and the control strategy can be implemented using Matlab 2015a. To power on the board the user has to flip the first lever from the left. The other two levers can be programmed freely.



 (!!! Tagliare parti non necessarie)

Issues

* Motor saturation

From simple test it’s easy to see that measured current saturates. What can be observed is that even though the current reaches a maximum value, the cart continues to move so this means that the sensor is not able to measure above a given current. In particular, it can be seen from Figures (specificare numero!!!) that the sensor saturates to 4,42 A and to -10 A. 

In any case the Arduino Due cannot reliably measure more than 5A current as the datasheet specifies, so from this point onward we will limit all test currents to 5A in both directions. Even in this way a difference remains between positive and negative measured current.

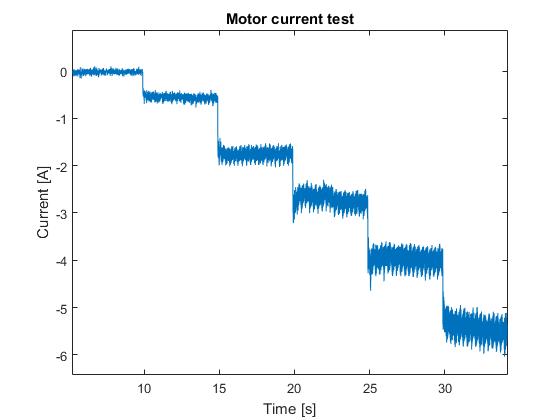
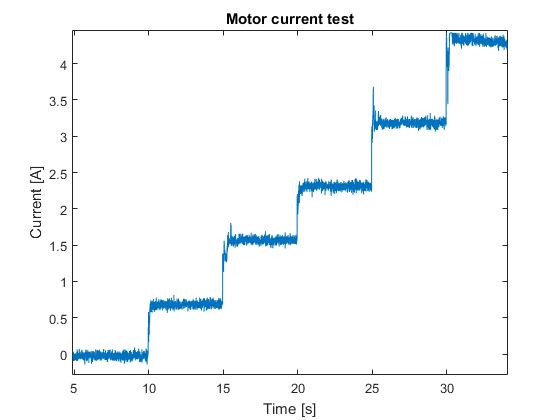
* Current Variance

After applying the protector to the system we measured the current noise at different current levels.

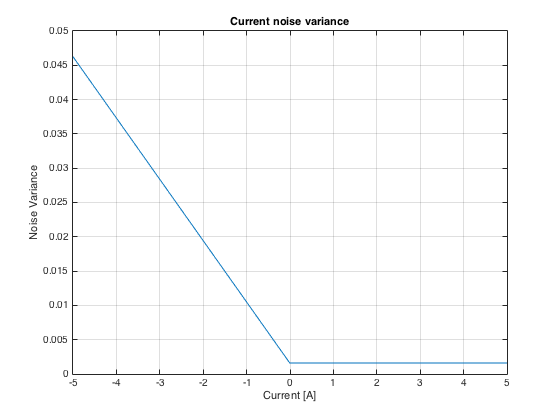
We applied a ladder type signal to measure the noise when the current was constant at different current levels. From the values obtained it can be seen that the variance of the negative current increases as the current increases in modulus instead if the current is positive the variance is small and almost constant. The fault of this behaviour for negative current is probably of the current measurement system.

|  |  |
| --- | --- |
| Mean | Variance |
| -0,020 | 0,0015 |
| -0,554 | 0,0035 |
| -1,758 | 0,0119 |
| -2,703 | 0,0274 |
| -3,964 | 0,0359 |
| -5,454 | 0,0523 |

|  |  |
| --- | --- |
| Mean | Variance |
| -0,022 | 0,0014 |
| 0,683 | 0,0013 |
| 1,568 | 0,0015 |
| 2,312 | 0,0016 |
| 3,182 | 0,0016 |
| 4,316 | 0,0022 |



From the variance values of the current it can be seen that the variance grows linearly with the modulus of the current so we decided to estimate the variance by simple linear interpolation. The results can be seen in figure(number!!!).



All these experiment were done in order to estimate the variance of the current sensor and to have enough information to build the variance matrixes for the Kalman filter.

* Encoder signal

Another problem is the one of the signal given by the encoders. This signal is a voltage impulse given to the Arduino. Unfortunately, the measure showed by Poliscope is not in centimetres so a conversion was in order. The idea was to give a reference signal in an open loop configuration. Once the transient is passed both Poliscope value and the real displacement of the cart were measured. The Tables (!!!numero) show the values of the various measurements. The final column shows the ratio between the Poliscope value and the displacement. This ratio has a mean value of about 560 and this is the value that has been used to convert all the measurements of the experiments.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **High stiffness spring** | | | | | Input (Volt) | Poliscope | Length (cm) | Ratio | | 1,5 | -120 | -0,2 | 600 | | 2 | -233 | -0,4 | 582,5 | | 2,5 | -266 | -0,5 | 532 | | 3,5 | -430 | -0,8 | 537,5 | | 4 | -500 | -0,9 | 555,5555556 | | 4,5 | -550 | -1 | 550 | | 5,4 | -650 | -1,2 | 541,6666667 | | 7,2 | -1030 | -1,8 | 572,2222222 | | 9 | -1350 | -2,4 | 562,5 | | |  |  |  |  | | --- | --- | --- | --- | | **Medium stiffness spring** | | | | | Input (Volt) | Poliscope | Length (cm) | Ratio | | 1 | -60 | -0,1 | 600 | | 1,5 | -310 | -0,55 | 563,63636 | | 2 | -530 | -1 | 530 | | 2,5 | -680 | -1,2 | 566,66667 | | 3 | -880 | -1,55 | 567,74194 | | 3,5 | -990 | -1,8 | 550 | | 4 | -1170 | -2,1 | 557,14286 | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| |  |  |  |  | | --- | --- | --- | --- | | **Low stiffness spring** | | | | | Input (Volt) | Poliscope | Length (cm) | Ratio | | 1,5 | -510 | -0,9 | 566,667 | | 2 | -900 | -1,6 | 562,5 | | 2,5 | -1050 | -1,9 | 552,632 | | 3 | -1335 | -2,4 | 556,25 | |  |  |  |  |  |  |  |

System protection

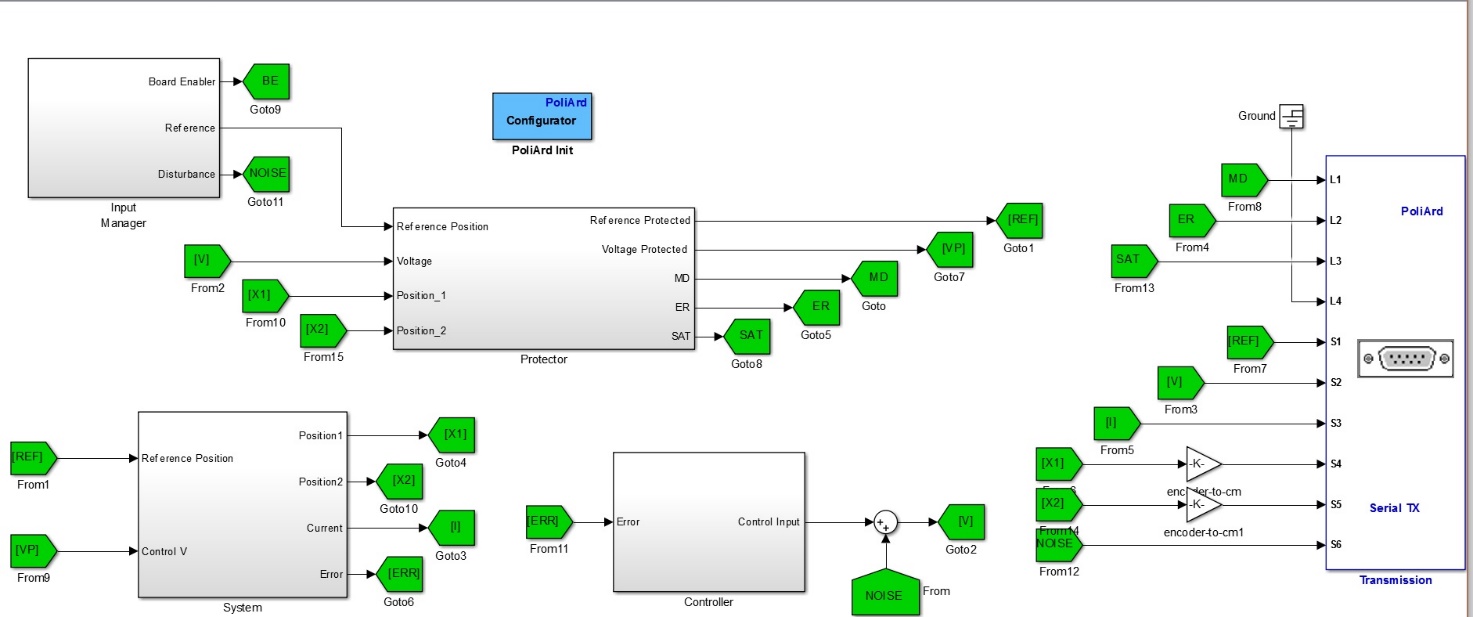
To avoid undesired behaviour of the system, a Simulink protection of the system has been designed. The system monitors current, voltage and displacement and if they overshoot a given value an alert is triggered that switches off the system to prevent any damage.

Moreover, Arduino Due doesn’t behave very well when the reference signal starts at the same time of its power on. For this reason, the system has been designed in order to start the signal only after 3 seconds the user pushed the lever number 2. This logic has been implemented using a finite state machine through the library Stateflow in Matlab.

The third lever instead it’s used to change from a reference signal to the potentiometer of the board.

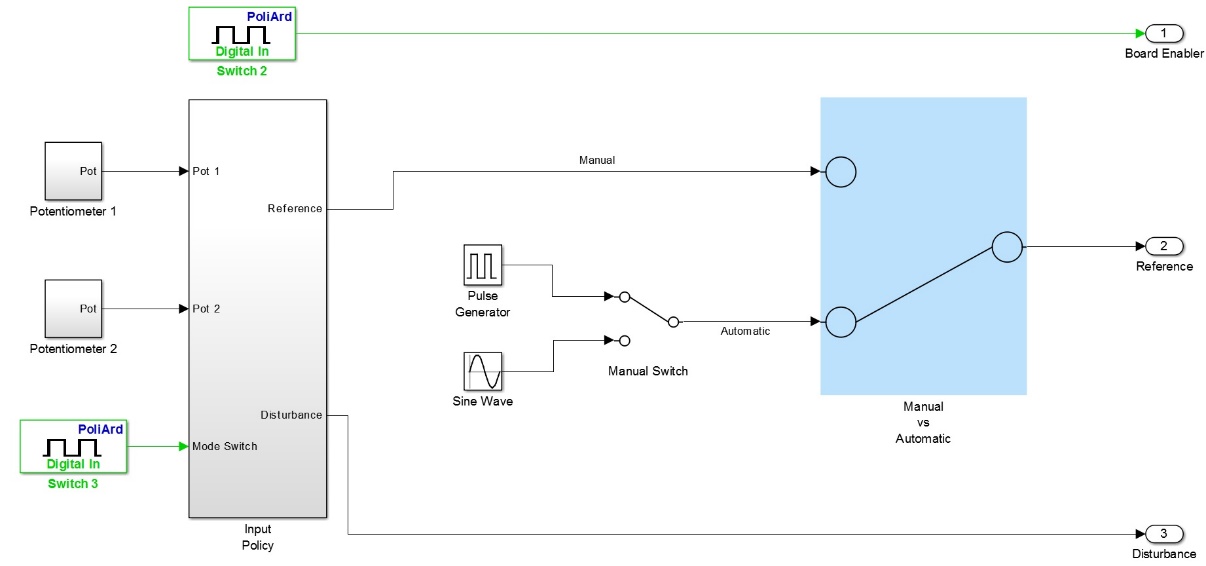
The system is composed by four macroblocks:

* Input Manager
* Protector
* System
* Controller



**Input manager**

Decides which input to use between potentiometers and signals. If the third switch of the Arduino board is enabled then the input switches from the signals to the manual potentiometers. One of the potentiometer is used to insert a disturbance in the control variable and the other one is used to give a variable and constant reference signal.



**Protector**

This part of the system is in turn composed by five protective blocks. One for the Voltage, one for the reference, three for the displacement.

The voltage protector simply saturates the voltage to avoid damage to the Arduino board.

The position protectors give an alert message if the displacement of any cart is more than ±3cm.

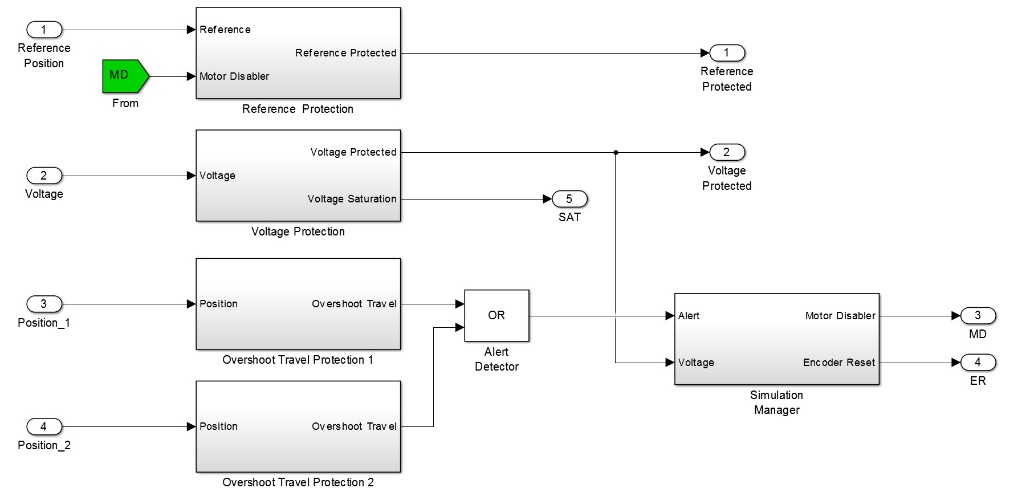
The reference protection saturates the reference to ±3 and a parallel circuit disables the input if the alert is active.

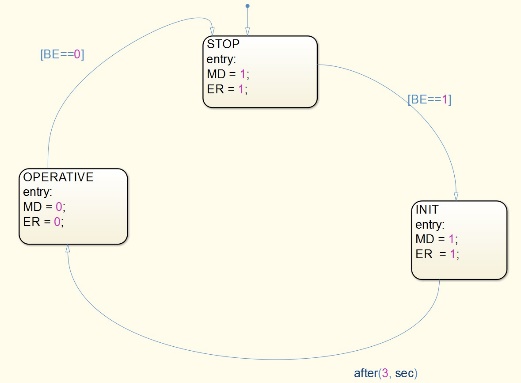
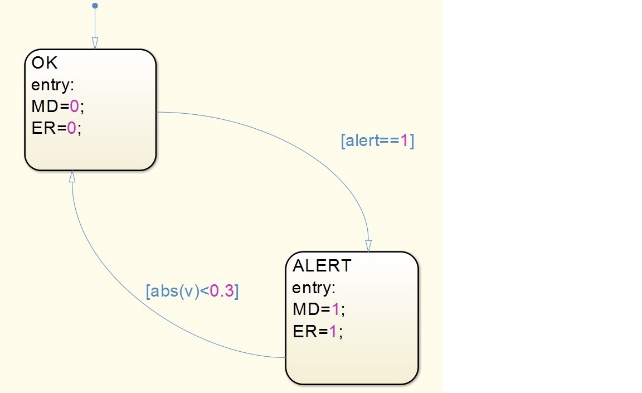
Finally, there’s a sixth block that contains two finite states machines:

* Init Manager
* Alert Manager

The first one is simply used to delay the start of the signals by 3 seconds when the board is turned on. This is done because, as said above, if the signals and the board switch on at the same time then the board behaves badly.

The second one processes all the signals and decides when to enable or disable the alert status. In particular, if the alert signal is on, the motor will be disabled and it won’t be enabled again until the voltage goes under ±0.3 V.

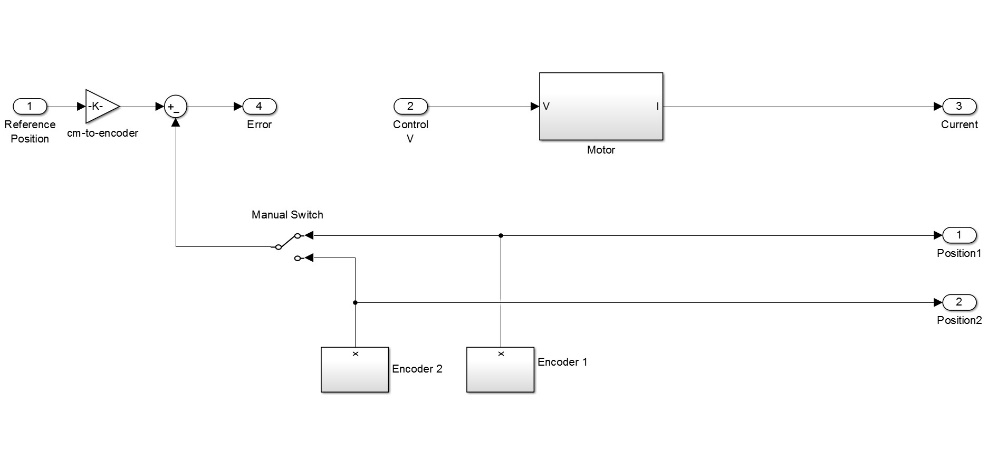


**System**

Inside the system the reference signal is subtracted to the value measured by the position encoder and this error will be used to stabilize the system. Prior to the tests, either encoder 1 or 2 can be selected as feedback.

The last block is the motor whose logic status is controlled by the motor disabler signal and whose continuous value is changed by the reference.



**Controller**

This block changes depending on which controller we want to use. When the motor disable signal is enabled there is a loop which rapidly pulls the voltage to 0 V in order to ready the system quickly. This exploit is used in order to avoid that the voltage inside the motor saturates for some reason and thus avoiding a total block of the system and the need of a manual intervention.