POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering

Master's Degree Thesis

Control and positioning of a pneumatic cylinder

Supervisor

Prof. Luigi MAZZA

Eng. Giuseppe PEPE

Candidates

Salvatore ARENA

JULY 2021

Abstract

This thesis work is based on the control and positioning of pneumatic and oil-pneumatic cylinders. These actuators are chosen since their high power-to-weight ratio. Also, they presents more advantage as a low cost maintenance.

The entire panel is designed in order to optimize the overall dimension of the means that are used. In particular, fixed devices are PLC , HMI and two cylinders: pneumatic cylinder and oil-pneumatic cylinder. Meanwhile, there are two pneumatic configurations: four electro-pneumatic 2/2 NC valves and two 3/2 NC valves.

The goal of this work is using different controls in order to manage the cylinders position, as shown in the following chapters.

This thesis work is performed in collaboration with my colleague Incarbone Carla, belonging to the same master's degree.

At the end of each section, a comparison between different control technologies are made, in order to obtain better work conditions and response parameters for each experiments. Also, in the final chapter, is reported a global comparison in order to highlight the best strategy control.

Table of Contents

List of Tables

List of Figures

Acronyms

PWM
Pulse width modulation

PFM
Pulse frequency modulation

PPM
Pulse position modulation

PLC
Programmable Logic Control

PID
Proportional Integrative Derivative

TIA
Totally Integrated Automation

DC
Duty Cycle

NC
Normally Close

HMI
Human Machine Interface

PTO
Pulse Train Output

OB

Organization Block

SCL

Structured Control Language

TF

Transfer Function

LTI

Linear-time invariant

Chapter 1

State of art and used components

1.1 State of art

Today pneumatic technology has superseded by electric and hydraulic technologies. This is due to the compressibility of the medium and due to the strong non-linearities present in the pneumatic models, which make difficult controls.
By adopting different solutions, in this thesis work, the goal is tried to obtain good performances with simple and reusable technologies.
In the literature (Chillari, Muscone, Guccione), the studies carried out on position control through the use of digital and analog valves, which led to good results. The studies carried out have a final architecture using a PID controller, adding 1-DOF.
In this thesis work the chosen valves are solenoid valves on/off, instead using proportional valves. This because the cost of the regulator is reduced and because on/off solenoid valves have a relatively large flow area compared to proportional valves.

In this work, the goal is to obtain results comparable to these, but with a simpler architecture.

The proposed solutions are:

- Proportional control with base control using 2/2 and 3/2 NC valves
- Proportional control with forced DC using 2/2 and 3/2 NC valves
- PID control using 2/2 and 3/2 valves
- Proportional control using driver speed up with 2/2 valves

In particular, the study is based on the optimization of the tracking error of the cylinder piston's position. The valves used in this project are high speed digital valves. These valves, modulating the DC, emulate proportional valves.

The control in position has non-linearity and therefore the parameters of the equations are very difficult to identify.
Furthermore, by restricting the parameters to their optimum neighborhood, it is possible to use normal methods. In the literature, these methods are described in mathematical languages, although they lead to non-linear equations. But this leads to models that are not reusable, and difficult to understand.
Simulating in this thesis the different experiments, in practice, with simulations, different results have reported for each single system developed.

The used system is described as follows. Each piston have 2 chambers, which are connected to two valves, respectively. Q0.1 and Q0.2 supply the air from the compressor, while Q0.0 and Q0.3 discharge it toward the environment. So, by opening and closing the valves, the desired position is obtained.

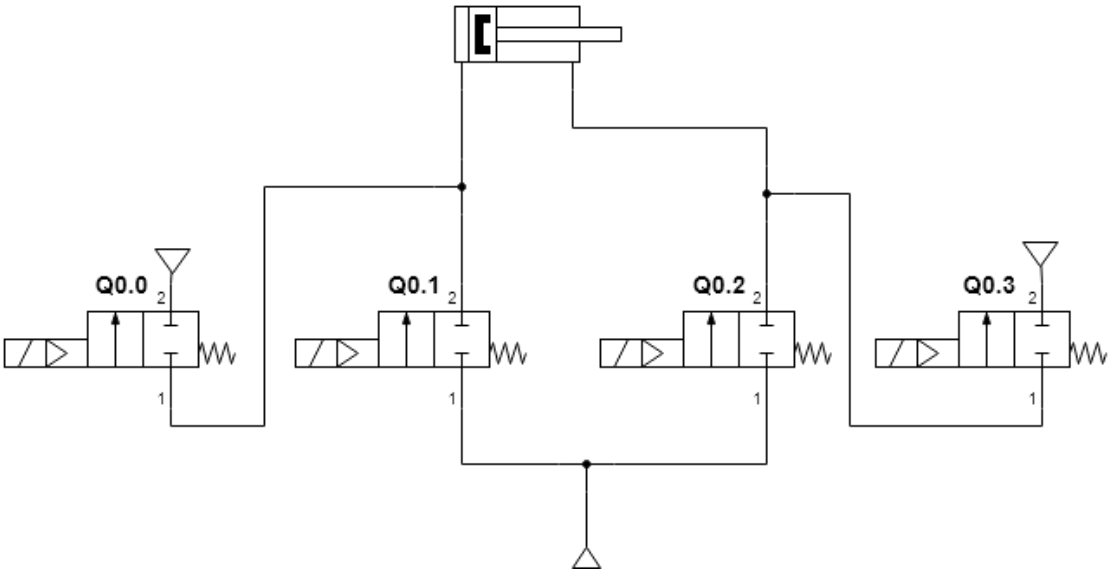


Figure 1.1: System using valves 2/2 NC

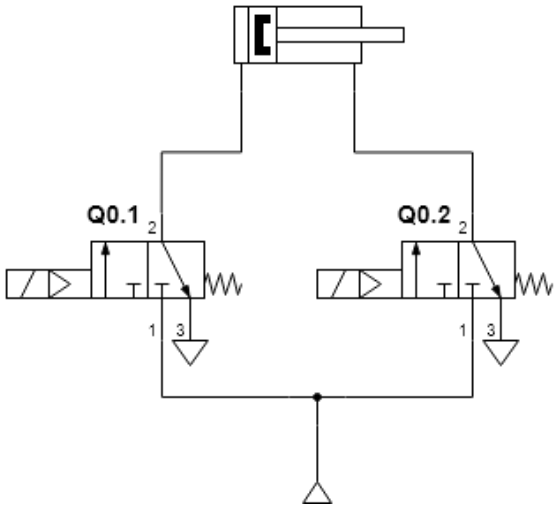


Figure 1.2: System using valves 3/2 NC

In these experiments a PWM (Pulse Width Modulation) control is used for the valves control. Also, the duty cycle (DC) of the valves is used, according to the figures below, in particular using a base proportional control and a forced DC control.

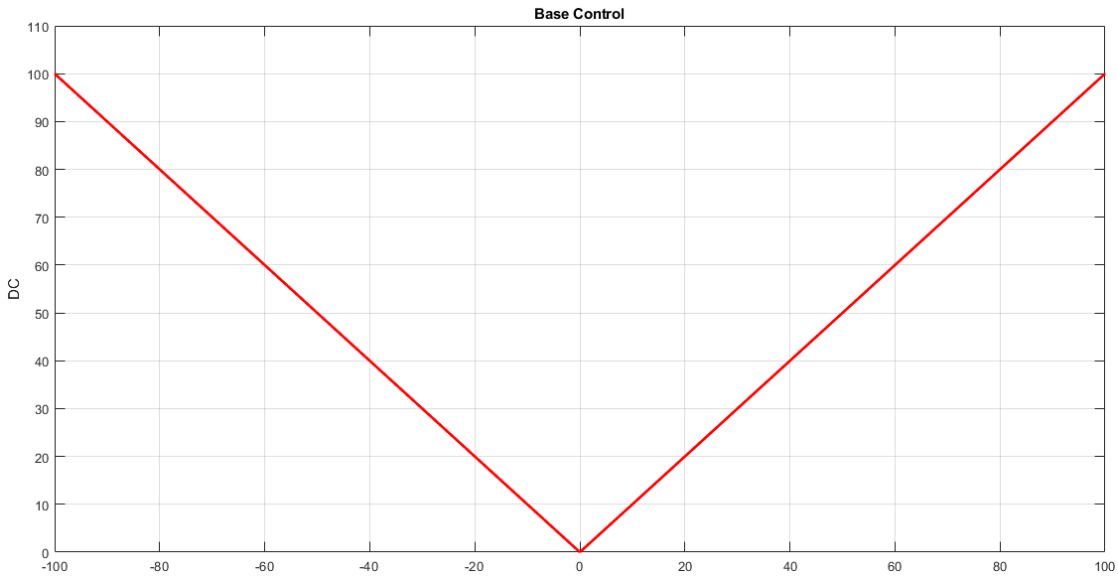


Figure 1.3: System description: Base control

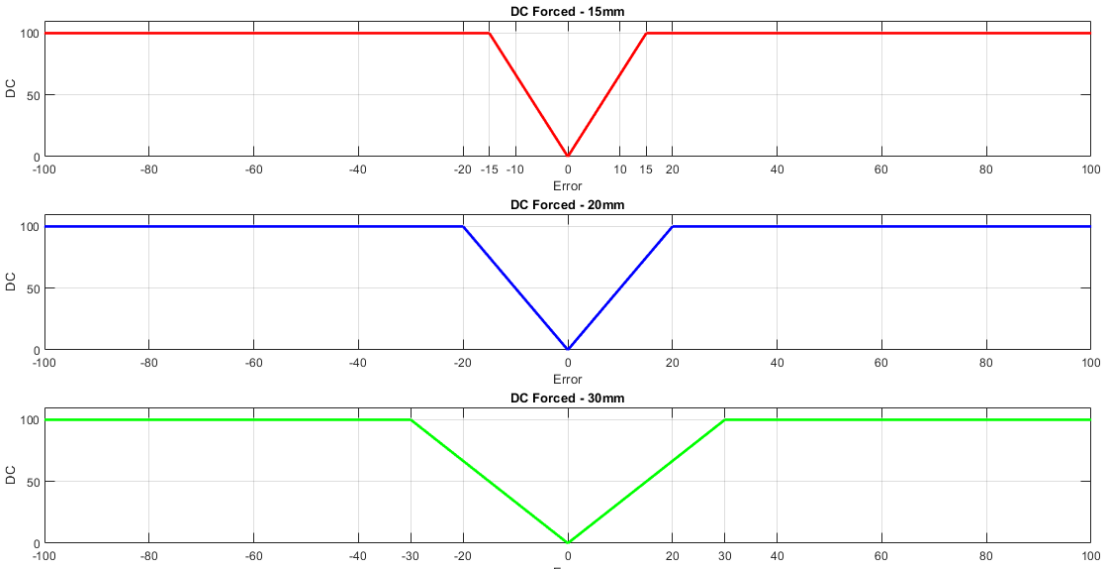


Figure 1.4: System description: Forced DC control

As in literature (K.U. Fil; V.N. Iliukhin; P.I. Greshniakov), also in this work, several experiments are proposed as the working frequency increases.
In detail, the responses over time were analyzed, as regards the use of the pneumatic cylinder and the oil-pneumatic cylinder, a frequency of 10Hz is chosen, after various tests, presented in the following chapters.
Now, the used instrumentation is analyzed and then proceed in detail with the experiments developed.

1.2 Used components

The panel is designed in order to have all the used components in one bank. The instrumentation used, in the thesis work, are:

- Stabilized power supply +24V
- Compressed air at 5bar
- PLC Simatic S7-1200 1214C DC/DC/DC
- HMI TP700-Comfort 7'
- Pneumatic cylinder Pneumax EA.A.0100.T.0100.C
- Oil-pneumatic cylinder Pneumax 1450.100.A.D.D

- Gefran transducer F055549 (Voltage)
- Gefran transducer F060773 (Current)
- Two Pneumax 3/2 valves MA3651F3C3KK00
- Four Pneumax 2/2 valves MA3411F0C33400
- Four cables female 3 poles A120947-ND 1-2273001-1
- For Matrix 2/2 valves OX821103C2KK
- Two Matrix 3/2 valves MX721103C3KK
- Matrix Driver SpeedUp P/N HSDB990012
- Three cables female 4 poles A120993-ND 2273011-1

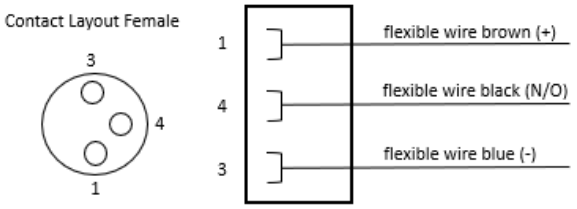
Also, in order to measure the correctness of signals the oscilloscope and the signals generator are used:

- Oscilloscope Gwinstek GDS-1054B
- Signals generator GFG 8216A

Figure 1.5: Oscilloscope

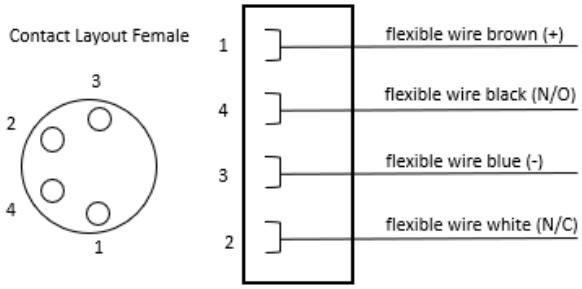
Figure 1.6: Signals generator

For the connection between the transducers and the PLC and also between the valves and the PLC the following cables are used:



PIN	FUNCTION
1	Power Supply +
4	Shield
3	Power Supply –

Figure 1.7: 3 poles wire and its datasheets



PIN	FUNCTION
1	Power Supply +
2	Output
3	Power Supply –
4	Shield

Figure 1.8: 4 poles wire and its datasheets

The obtained characteristic of the two transducers are:

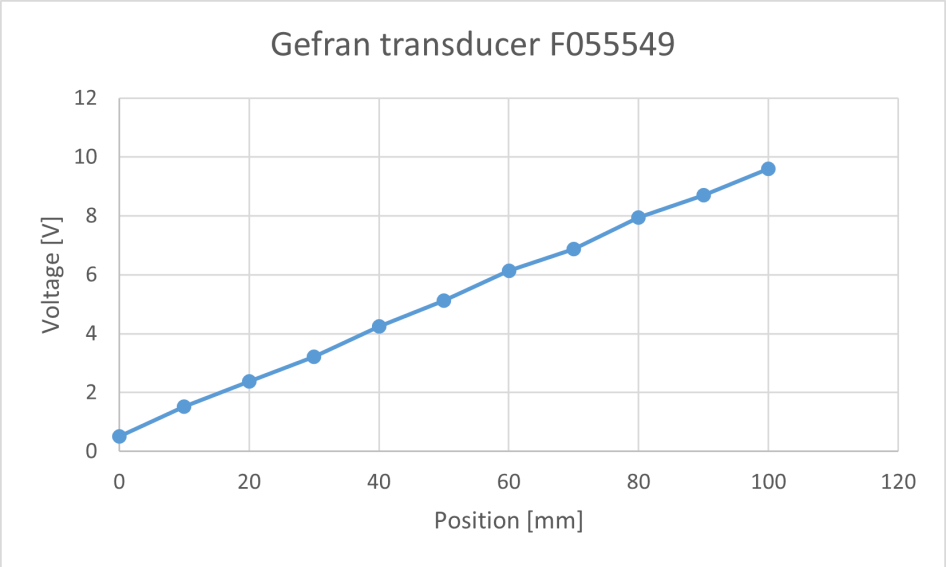


Figure 1.9: Gefran transducer F055549

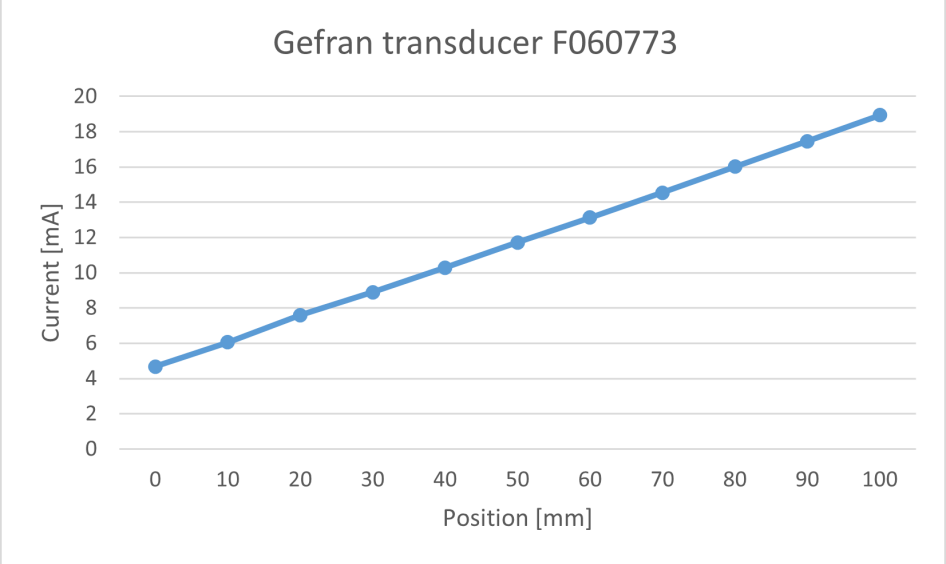


Figure 1.10: Gefran transducer F060773

These measurements are obtained each 10mm, with measure the voltage/current sensor's outputs.

Chapter 2

Proportional algorithm with pneumatic cylinder

2.1 Proportional control using valve 2/2 NC

In this section, using the pneumatic cylinder, is used a proportional algorithm with two different modulations. The first modulation is pure proportional and the second one is with a forced amplitude. The used pneumatic connection is the same for each control. Also, to simplify analysis operations the used name convention is:

- Q0.0 : outlet outstroke
- Q0.1 : outstroke
- Q0.2 : instroke
- Q0.3 : outlet instroke

For both algorithms, the DC modulation depends on the difference between *Setpoint* and *Feedback* (Error). In order to exploit as better as possible the valves, the pulse generator of the PLC is activated. In particular, the pulses generator PTO1/PWM1 and PTO2/PWM2 are set, as shown in fig. 2.1 and fig. 2.2

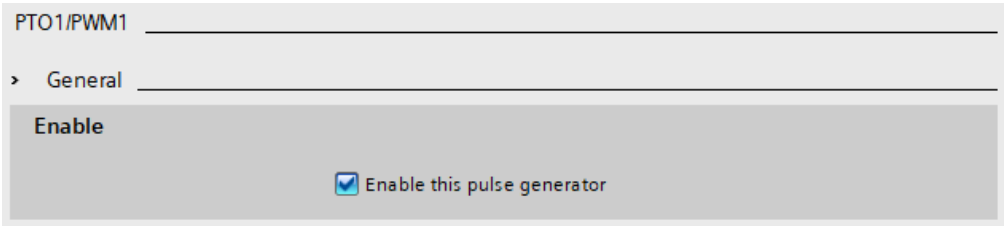


Figure 2.1: Enable of PTO1/PWM1

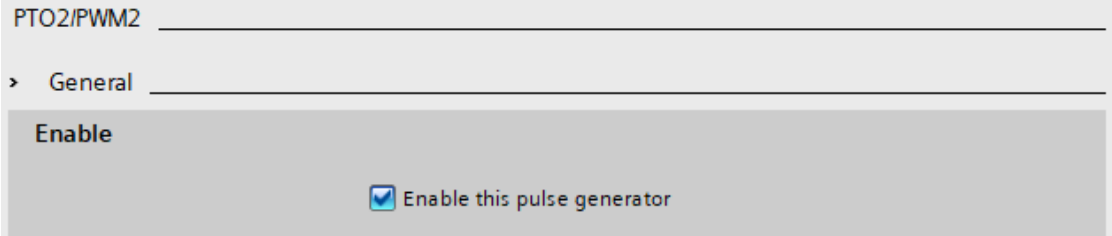


Figure 2.2: Enable of PTO2/PWM2

The parameters for both the pulses generator are set with a cycle time equal to 100ms and also a pulse output ports are Q0.1 and Q0.2, as follows:

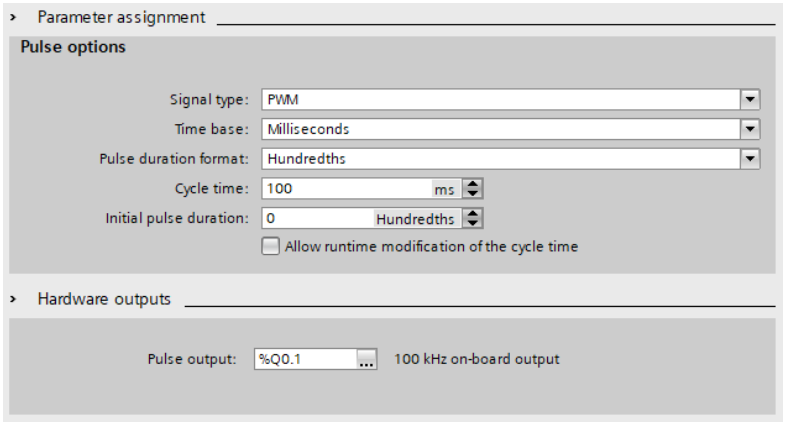


Figure 2.3: Parameters of PTO1/PWM1

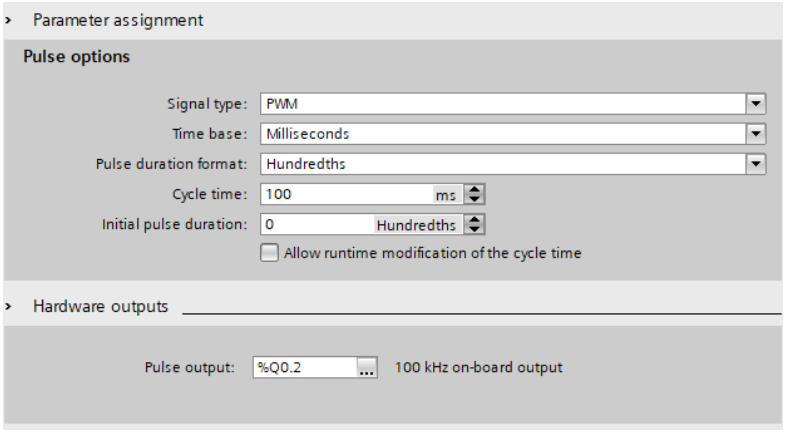


Figure 2.4: Parameters of PTO2/PWM2

Moreover, the final configurations which are set are the address for both ports,

in particular:

Figure 2.5: Address of PTO/PWM ports

After these setting configurations, in order to have a correct behaviour two blocks are used , as shown in fig. 2.6.
The first one is the $CTRL\_PWM$, which allows the use of the pulse train, in particular it use the 265 address or 266 address to activate the corresponding PTO/PWM port.
The second is the $MOVE$ block, that allows to transfer the calculated error to the destination port.

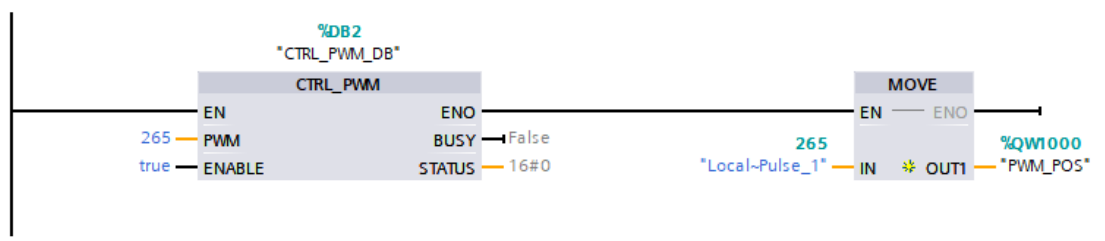


Figure 2.6: CTRL_PWM Block and MOVE Block

The system's dynamics change based on the chosen square wave's period. In order to have a better system response, six different experiments are made, as shown in the fig.2.7

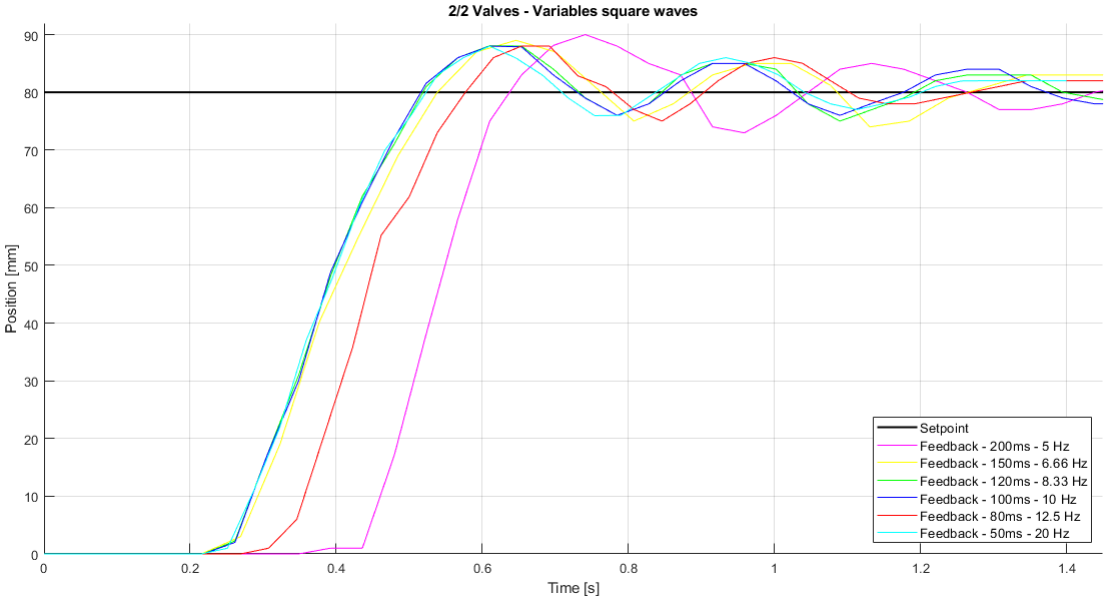


Figure 2.7: Signals modulated experiments

So, it is possible to observe that the more efficient periods are with a frequency modulation equal to: 50ms, 80ms and 100ms:

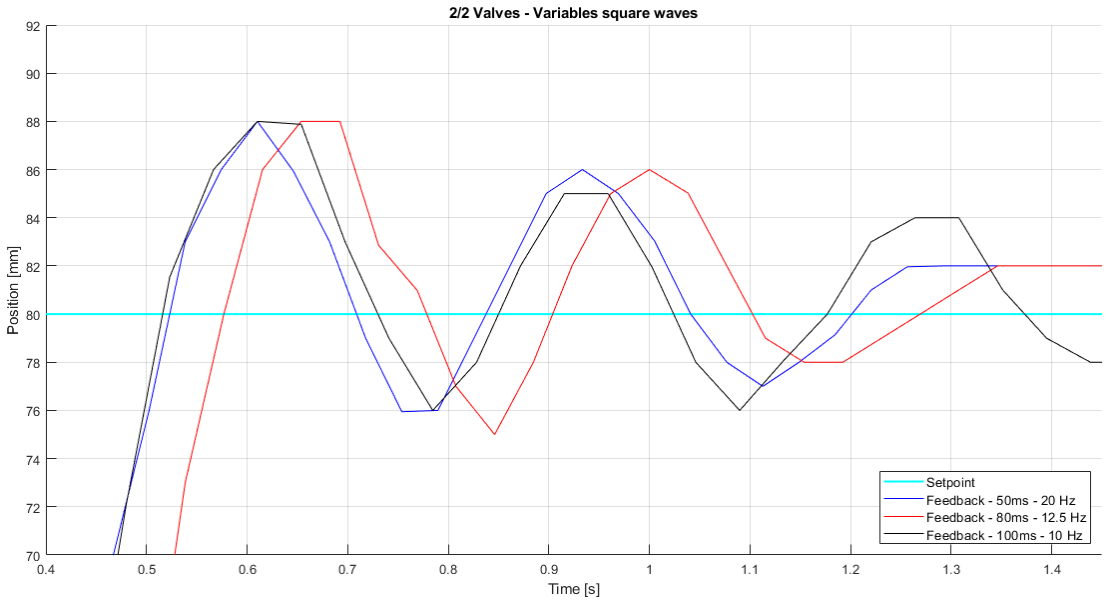


Figure 2.8: Best signals modulated

The chosen period is equal to 100ms for each measure, based on the DC, the

response of the system corresponds with the correct modulation.
To verify the behaviour of the PWM modulation, through the oscilloscope (fig.1.5), the different measurements are analyzed:

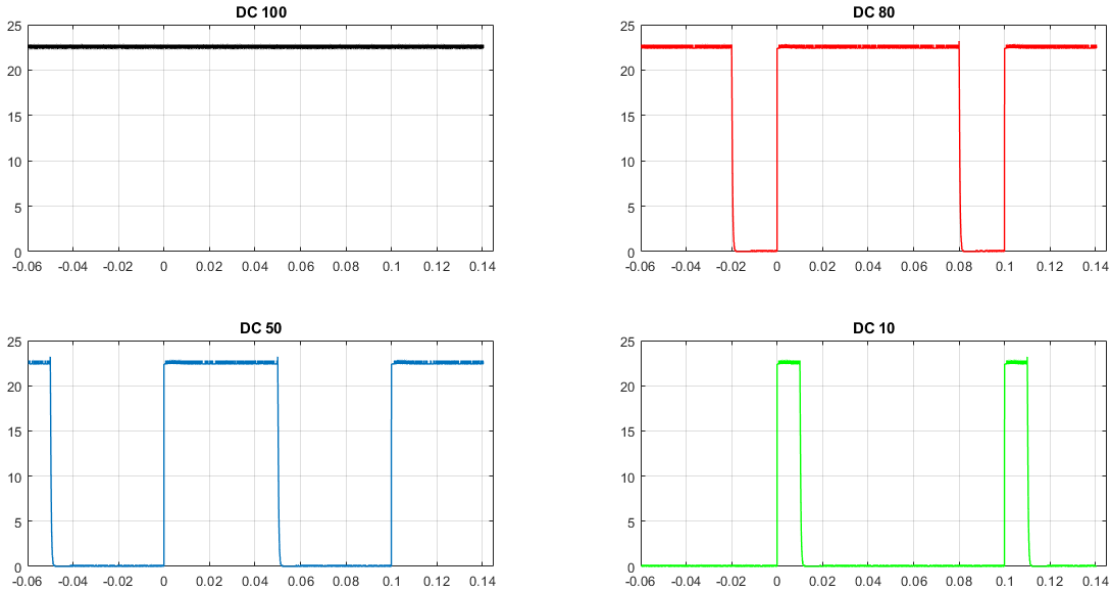


Figure 2.9: Measurements made with the oscilloscope

2.1.1 Base Control

For the base control, the algorithm is based on a difference between *Setpoint*, which is the final point that the cylinder must achieve, and *Feedback*, which is the real position of the cylinder.

In particular, all the segments designed and used in TIA portal are presented in the following section.
Indeed, the variables used are reported in the table below:

DATA BLOCK

Static		
FEEDBACK	Int	0
ABS_ERROR	Int	0
SETPOINT	Int	0
NORMX_MIN	Int	1450
NORMX_MAX	Int	26450
SCALEX_MIN	Int	0
SCALEX_MAX	Int	100
NORMX_VALUE	Int	0
NORMX_OUTPUT	Real	0.0
ERROR	Int	0
INCREASE	Bool	false
DECREASE	Bool	false
DC_POS_0_100	Int	0
DC_NEG_0_100	Int	0
EMERCENCY/STOP	Bool	false
START	Bool	false

PLC VARIABLES

PWM_POS	Int	%QW1000
PWM_NEG	Int	%QW1002
OUTLET_INSTROKE	Bool	%Q0.3
OUTLET_OUTSTROKE	Bool	%Q0.0
OUTSTROKE	Bool	%Q0.1
INSTROKE	Bool	%Q0.2
MERKER_INCREASE	Bool	%M0.1
MERKER_DECREASE	Bool	%M0.2
MERKER EMERCENCY	Bool	%M0.3
IW64	Int	%IW64

Figure 2.10: Data Block and PLC variables - pneumatic cylinder

NAME	USE
FEEDBACK	Int variable used to manage the stroke position
ABS_ERROR	Int variable used to carry out the converted value of the error coming from the ABS block
SETPOINT	Int variable used as reference
NORMX_MIN	Predetermined Int variable used as minimum element on the normx block
NORMX_MAX	Predetermined Int variable used as maximum value on the normx block
SCALEX_MIN	Predetermined Int variable used as minimum element on the scalex block
SCALEX_MAX	Predetermined Int variable used as maximum value on the scalex block
NORMX_VALUE	Int variable used as input value of the normx block
NORMX_OUTPUT	Real variable used to carry out the output value of the normx block
ERROR	Int variable calculated as difference between SETPOINT and FEEDBACK
INCREASE	Bool variable equal to TRUE when the system is in increasing phase
DECREASE	Bool variable equal to TRUE when the system is in decreasing phase
DC_POS_0_100	Int variable used to carry out the integer value of the Duty cycle in increasing phase
DC_NEG_0_100	Int variable used to carry out the integer value of the Duty cycle in decreasing phase
EMERGENCY/STOP	Bool variable used to manage the stopping phase
START	Bool variable used to start the system
PWM_POS	Int variable used to manage the PWM in increasing phase
PWM_NEG	Int variable used to manage the PWM in increasing phase
OUTLET_INSTROKE	Bool variable used to manage the output port Q0.3 of the PLC
OUTLET_OUTSTROKE	Bool variable used to manage the output port Q0.0 of the PLC
OUTSTROKE	Bool variable used to manage the output port Q0.1 of the PLC
INSTROKE	Bool variable used to manage the output port Q0.2 of the PLC
MERKER_INCREASE	Bool variable equal to TRUE when the system is in increasing phase
MERKER_DECREASE	Bool variable equal to TRUE when the system is in decreasing phase
MERKER_EMERGENCY	Bool variable equal to TRUE when the system is in emergency case
IW64	Int variable used to read the value coming from the input port IW64

Figure 2.11: Data block and PLC variables - USE - 2/2 valves - pneumatic cylinder

The first segment shown in fig 2.12, the transducer segment, deals of reading the IW64 input. Thanks to the use of the $NORMX$ and $SCALEX$ blocks it is possible to adapt the $NORMX\_VALUE$ for the correct behaviour of the algorithm.
The minimum reading value by the transducer is equal to 1450, which is associated by the PLC to the minimum position of the piston (0 mm). While, the maximum value, which corresponds instead to the maximum stroke of the cylinder (100 mm) is equal to 26630.
The $NORMX$ block allows to normalized the input $NORMX\_VALUE$, which is read trough the sensor, to a real output between 0 and 1.
While, the $SCALEX$ block is used to scaled the real output into an integer value between 0 and 100 [mm]. In this way, the system know the actual position of the piston, and it is possible to use this output as *Feedback* for each part of the algorithm.

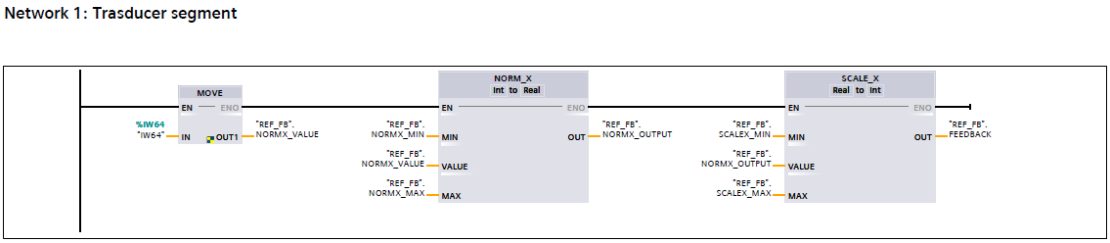


Figure 2.12: Network 1: Transducer segment of the pneumatic cylinder

The segment related to the Emergency is shown in fig. 2.13. It is designed in order to stop the system in emergency cases. The stop of the piston is instantaneous and it happens in every position.
To restart the algorithm the user must to push the button $START$ present on the HMI panel (A.5). Both buttons $START$ and $EMERGENCY/STOP$ reset and set, respectively, the merker M0.3.

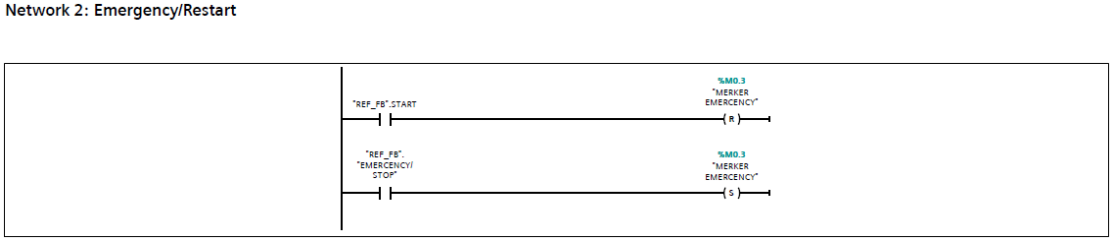


Figure 2.13: Network 2: Emergency/Restart

The network shown in the fig. 2.14 is the most important segment. In particular, this peace of code managed the outstoke and the instroke of the piston. Based on

the comparison between the *ERROR* and zero. If the comparison in the first rung is TRUE, so the error value is moved to the integer variable *DC_POS_0_100*. This variable allows to control the square wave modulation sent to the Q0.1 port like a PWM signal. In this way is possible to control the pressure in the rear chamber. On the other side, the front chamber is controlled by the Q0.3 port, which connect the front chamber to the exhaust, so the outstroke occurred.
While, if the *ERROR* is less than zero, the activated rung is the second. So, the error is negative and thanks to an *ABS* block is converted in a positive value (*ABS_ERROR*).
After that, the signal is sent to the Q0.2 port. So, the Q0.0 port is activated and the pressure of the rear chamber will be equal to the atmospheric pressure, and the instroke happens.
In this segment, also, there are two merkers (increase and decrease).
Alternatively, these merkers are set and reset. In this way, the PTO1/PWM1 is asserted, instead the PTO2/PWM2 is deactivated, and viceversa. Therefore, is avoid the simultaneous activation of both pulse generators.

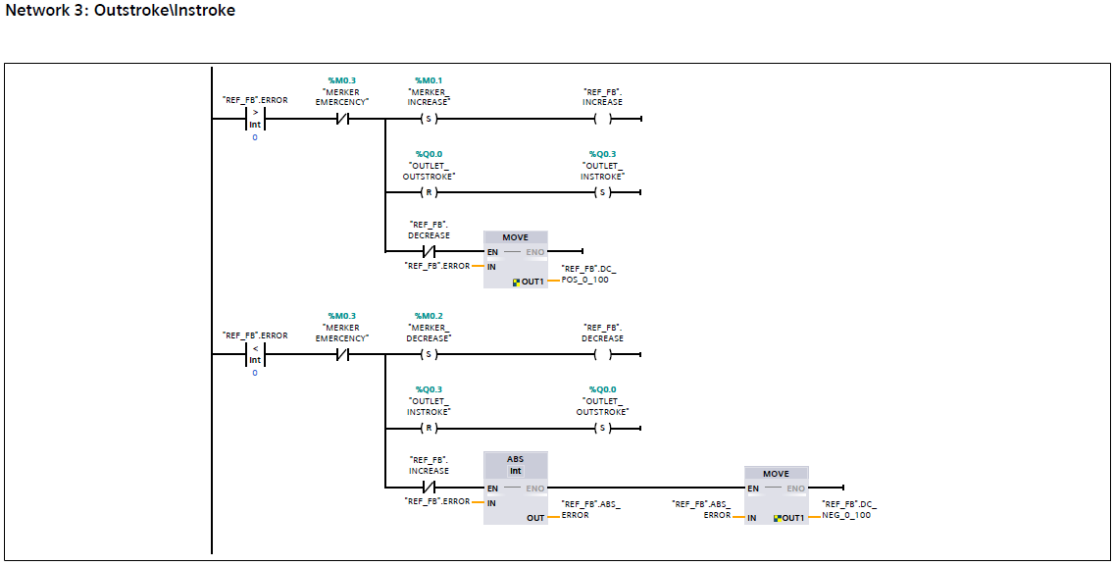


Figure 2.14: Network 3: Outstroke/Instroke

The four network (fig.2.15) is related to the error calculation between the integer value of the *Setpoint* and the *Feedback*. The difference is the integer variable *ERROR* useful in the previous segments, which allows the outstroke or the instroke of the cylinder's piston.

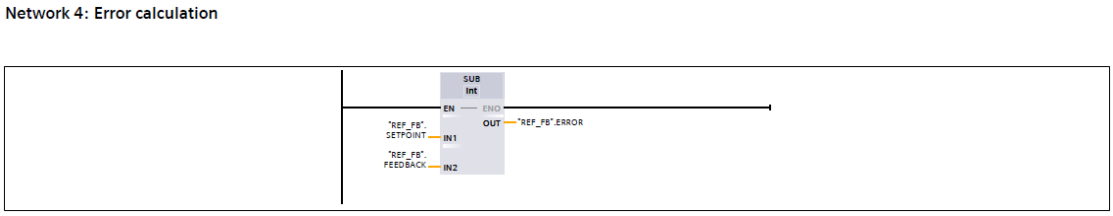


Figure 2.15: Network 4: Error segment

The following segment (fig.2.16) deals to hold the activation of the pulse train and to manage the modulation of the DC.
Moreover, the merker_Increase asserted the first PWM generator (265), so if the $CNTL\_PWM$ block is enabled, but in order to transfer the correct DC, from the integer DC variable to the Q0.1 (QW1000) port, a $MOVE$ block is needed.
In the same way, the merker_decrease works using the PWM (266) and sent the square wave to the Q0.2 (QW1002) port.

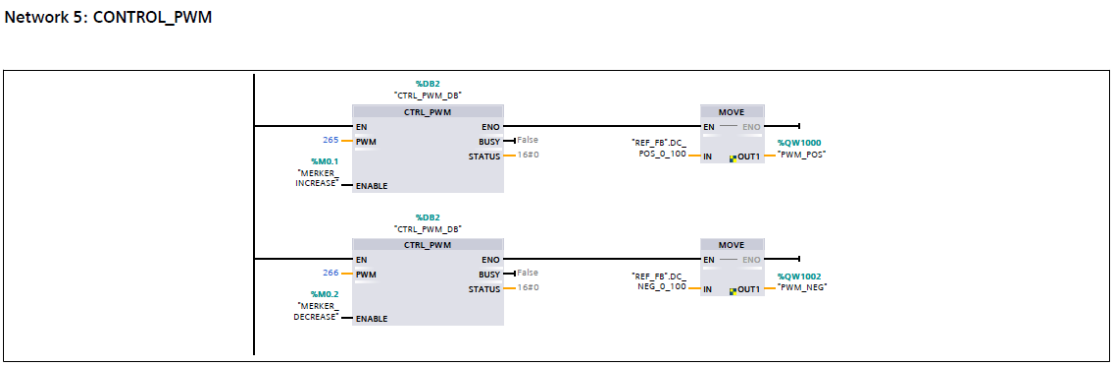


Figure 2.16: Network 5: Control PWM blocks and PWM modulation

The last designed control is used for two different cases (2.17). One for emergency cases, in particular if the Emergency_merker is activated, it provides to reset Q0.0, Q0.3 and the merkers increase and decrease. The second one, happens if the $ERROR$ is equal to zero, so means that the $Feedback$ achieved the $Setpoint$ and the piston must be stopped.

Network 6: Final_position

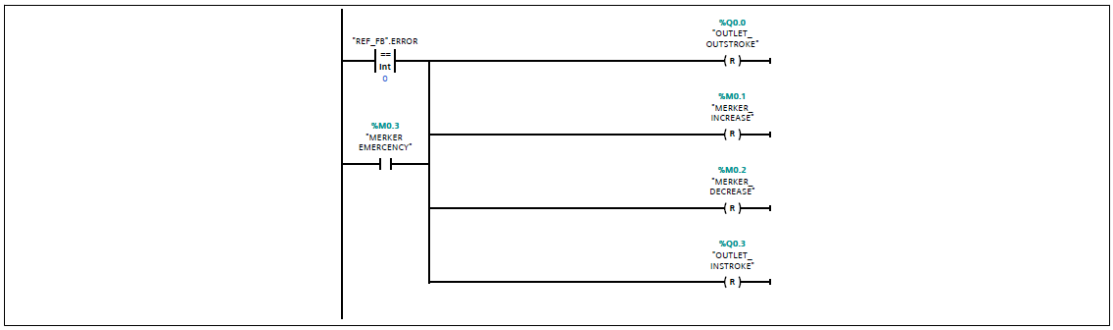


Figure 2.17: Network 6: Final Position of the piston

After the explanation of each segments, the main possible situations are:

1. $Setpoint > Feedback$

2. $Setpoint < Feedback$

3. $Setpoint = Feedback$

In any case, the response of the system occurs in a proportional way. As follows it is possible to observe the system response, in terms of dynamic behaviours.

Overshoot %	Rise Time [s]	Settling Time [S]
6,25	0,396	1,195

Table 2.1: System response - Base control - 2/2 Valves - pneumatic cylinder

Also, the graph below shows the cylinder's characteristic in time:

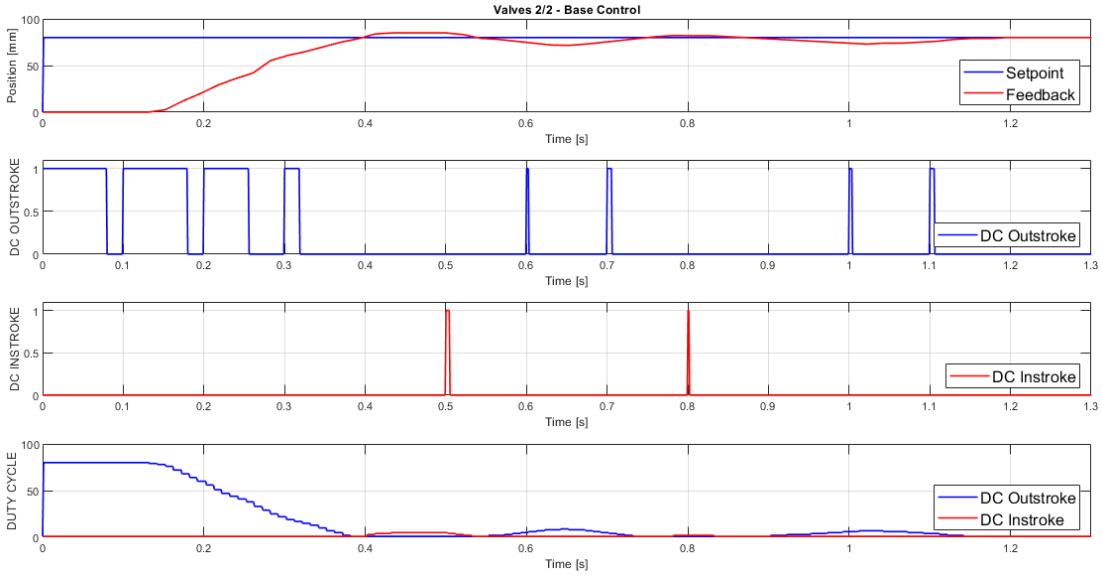


Figure 2.18: Dynamic response - Base control - 2/2 Valves - pneumatic cylinder

2.1.2 Control with forced duty cycle

In this section, the analyzed control is made in order to improve the characteristic of the previous scenario.
Also, other variables are used , as shown in the following table with the respective use:

NAME	USE
FEEDBACK	Int variable used to manage the stroke position
ABS_ERROR	Int variable used to carry out the converted value of the error coming from the ABS block
SETPOINT	Int variable used as reference
NORMX_MIN	Predetermined Int variable used as minimum element on the normx block
NORMX_MAX	Predetermined Int variable used as maximum value on the normx block
SCALEX_MIN	Predetermined Int variable used as minimum element on the scalex block
SCALEX_MAX	Predetermined Int variable used as maximum value on the scalex block
NORMX_VALUE	Int variable used as input value of the normx block
NORMX_OUTPUT	Real variable used to carry out the output value of the normx block
ERROR	Int variable calculated as difference between SETPOINT and FEEDBACK
INCREASE	Bool variable equal to TRUE when the system is in increasing phase
DECREASE	Bool variable equal to TRUE when the system is in decreasing phase
DC_POS_0_100	Int variable used to carry out the integer value of the Duty cycle in increasing phase
DC_NEG_0_100	Int variable used to carry out the integer value of the Duty cycle in decreasing phase
EMERGENCY/STOP	Bool variable used to manage the stopping phase
START	Bool variable used to start the system
INRANGE_POS	Int variable used as upper threshold of the INRANGE block
INRANGE_NEG	Int variable used as lower threshold of the INRANGE block
OUTPUT_IN_POS	Bool variable used to manage the use of the forced duty cycle in increasing phase. It's setted when the DC is forced.
OUTPUT_IN_NEG	Bool variable used to manage the use of the forced duty cycle in decreasing phase. It's setted when the DC is forced.
PWM_POS	Int variable used to manage the PWM in increasing phase
PWM_NEG	Int variable used to manage the PWM in increasing phase
OUTLET_INSTROKE	Bool variable used to manage the output port Q0.3 of the PLC
OUTLET_OUTSTROKE	Bool variable used to manage the output port Q0.0 of the PLC
OUTSTROKE	Bool variable used to manage the output port Q0.1 of the PLC
INSTROKE	Bool variable used to manage the output port Q0.2 of the PLC
MERKER_INCREASE	Bool variable equal to TRUE when the system is in increasing phase
MERKER_DECREASE	Bool variable equal to TRUE when the system is in decreasing phase
MERKER_EMERGENCY	Bool variable equal to TRUE when the system is in emergency case
IW64	Int variable used to read the value coming from the input port IW64

Figure 2.19: Data block and PLC variables - USE - Forced DC Control

In particular, the goal is to improve the rise time to achieve the desired position (*Setpoint*) in a faster way. In this case is chosen to use the DC, because in the previous control the DC reached 80%, so the DC is not exploited to the maximum.

So, the used algorithm is based to use the maximum flow air trough the forced DC. In this way, the DC reached the 100% and the opening time of the valve is greater. The main particular of this algorithm is the used threshold, which allows to handle the forced DC.

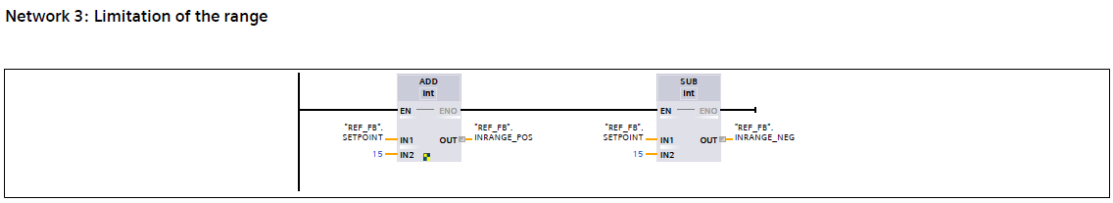


Figure 2.20: Network 3: Limitation of the range

In the segment represented in the fig.2.20, the range of action of the forced DC is calculated through the sum and the difference between the *Setpoint* and the threshold. The threshold chosen in this experiment is equal to 15 [mm].

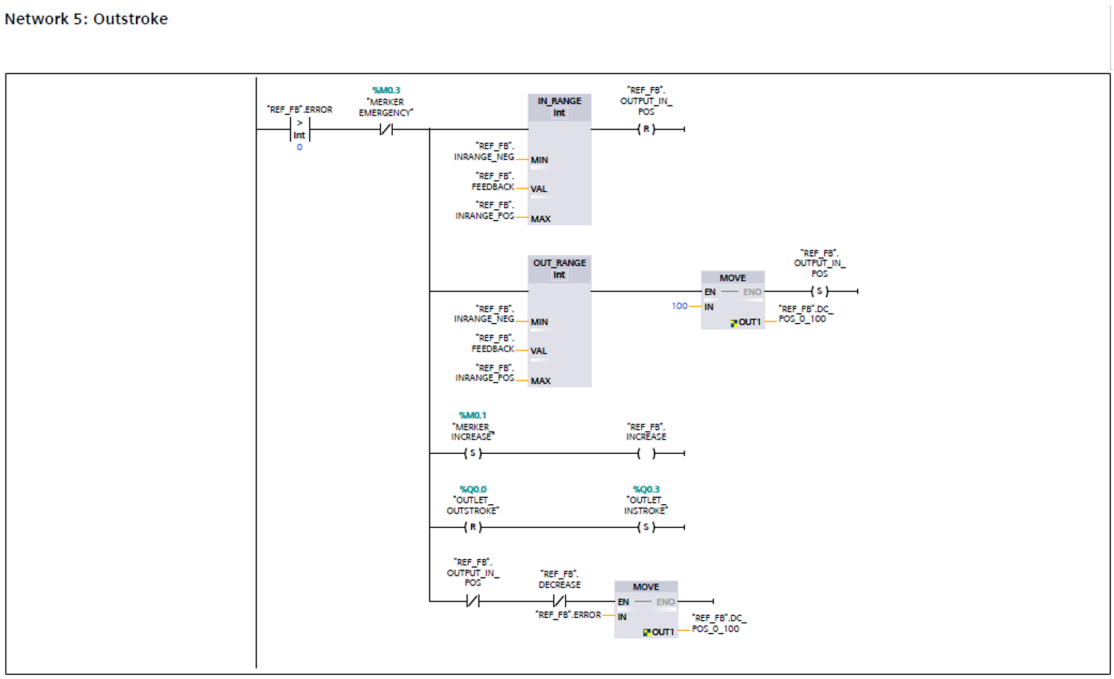


Figure 2.21: Network 5:Outstroke trough the OUT-Range

Thanks to the OUT-RANGE Block, present in segment, the DC is controlled. So, if the OUT-RANGE have a true output the DC is forced to the maximum value (100%) and the response of the system is better. This is obtained by the use of the *MOVE* Block. So, directly the integer value 100 is passed to the integer

variable $DC\_POS\_0\_100$. Otherwise, OUT-RANGE is not verified, the DC is calculated as the previous case, represented in the fig.2.14.
Also, in the network 5 (fig.2.21), is possible to observe the use of other useful ports. In particular, in the second rung is present a $OUTPUT\_IN\_POS$, which is the output set with memory, that allows to open the NC contact in the last rung in order to move the $ERROR$ to the DC. On the other hand, if the $OUTPUT\_IN\_POS$ is reset the contact remained closed and the system works in a proportional way. So, as shown in fig.2.21 when the *Feedback* is far from the *Setpoint* the DC is 100%. In this way it is possible to observe an improvement of the performance, having a lower rise time.
In case you want the proportional control from a higher range, the actual 15 must be change with a greater value. Indeed, this approach with this range shows an high overshoot, as in fig.2.22 .

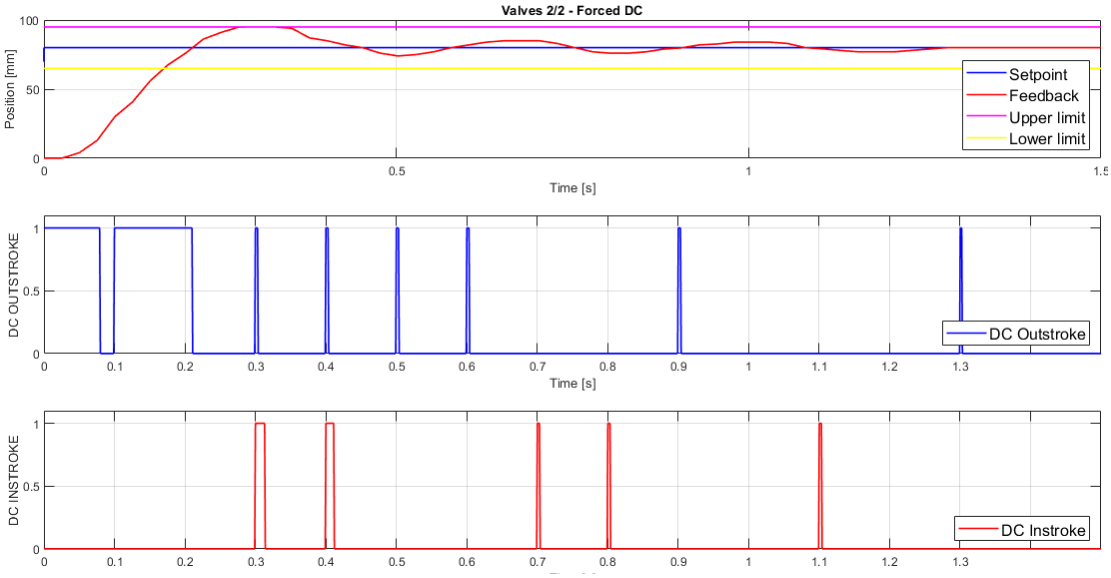


Figure 2.22: Response with a range ±15 [mm] - 2/2 Valves

As in fig.2.22, it is possible to observe that the system response matched with the algorithm, because between 65 mm and 95 mm the control is proportional DC, otherwise out the range the DC is forced to 100%. The percentage overshoot , in this particular case, is equal to 18.75%. So, to improve it, it's possible to use an higher range. In the table below, are shown other experiments with different ranges, that decrease the pick of the overshoot:

Range [mm]	Overshoot %	Rise Time [s]	Settling Time [S]
±15	18,7	0,31	1,795
±20	15	0,32	1,750
±30	10	0,375	1,575

Table 2.2: System response - Forced DC - 2/2 Valves - pneumatic cylinder

So, it is possible to observe by the table 2.2, that the overshoot and the settling time decrease with respect to the range increasing. On the other hand, the rise time is worsened. This because the maximum opening time value of the valves decrease.
The comparison between Base control and the Forced DC control algorithms is shown in the table 2.3, the first has a better settling time and overshoot in terms of response; while, the second one has a more efficient rise time behaviour, as previously stated.

Algorithm	Overshoot %	Rise Time [s]	Settling Time [S]
Base Control	6,25	0,396	1,195
Forced DC Control	18,75	0,31	1,795

Table 2.3: Comparison algorithms pneumatic cylinder - 2/2 Valves

2.1.3 Evaluation of tracking error

In this section is analyzed the tracking error in both cases: Base control and Forced DC control.
This error is evaluated with a sinusoidal and saw-tooth input, in order to recognize the limit frequency of the real system until the error's system is perceptible. Thanks the use of a signals generator (fig.1.6), the variable *Setpoint* is obtained connecting the output of the signals generator with the input port IW66.
Since, the PLC's inputs accept only positive voltage values, the sine wave is obtained by introducing an offset to the generator. As shown in the segment 2.23 , the input port IW66 is normalized and after scaled, in order to have a right value between 10 and 90 mm.

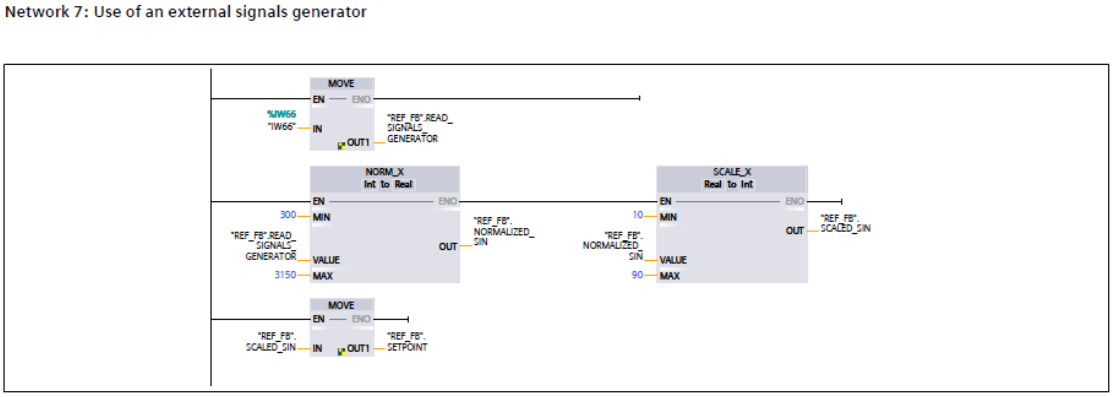


Figure 2.23: External signals generator

In this way, it is possible to change the frequency of the sine wave, which allows different experiments. The obtained data are evidenced in the following figures, with the frequencies between 0.5Hz and 2Hz:

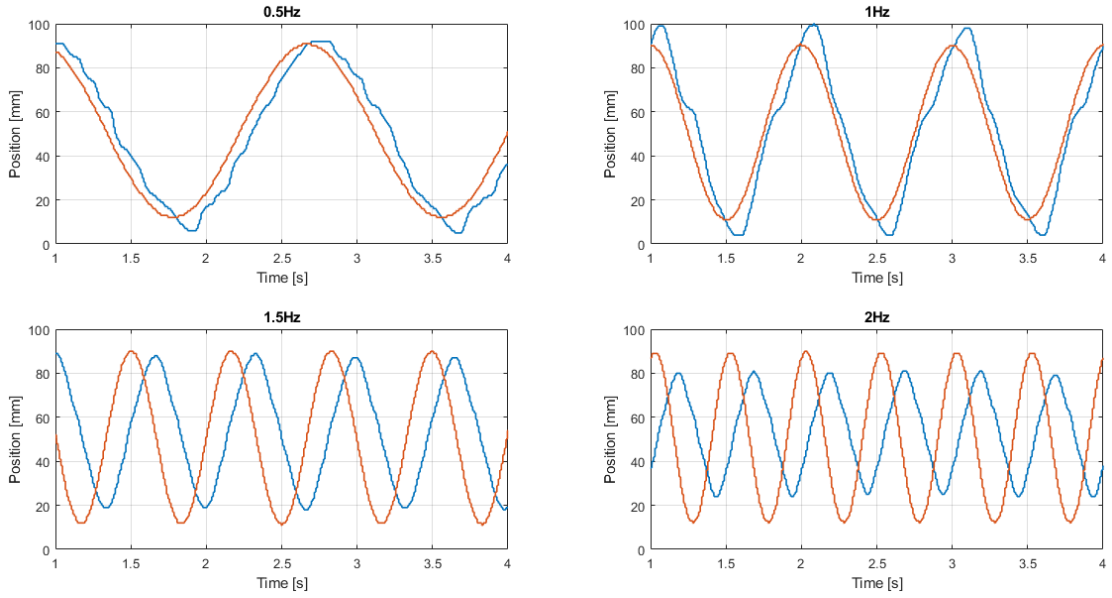


Figure 2.24: Tracking error - Base Control

The position error in fig. at 0.5Hz is quite small, otherwise at 2Hz is detectable a very huge tracking error. Thanks to these analysis of data the found limit frequency is less than 1Hz. On the other hand, by the analysis of the tracking error due to the forced DC algorithm the obtained results are:

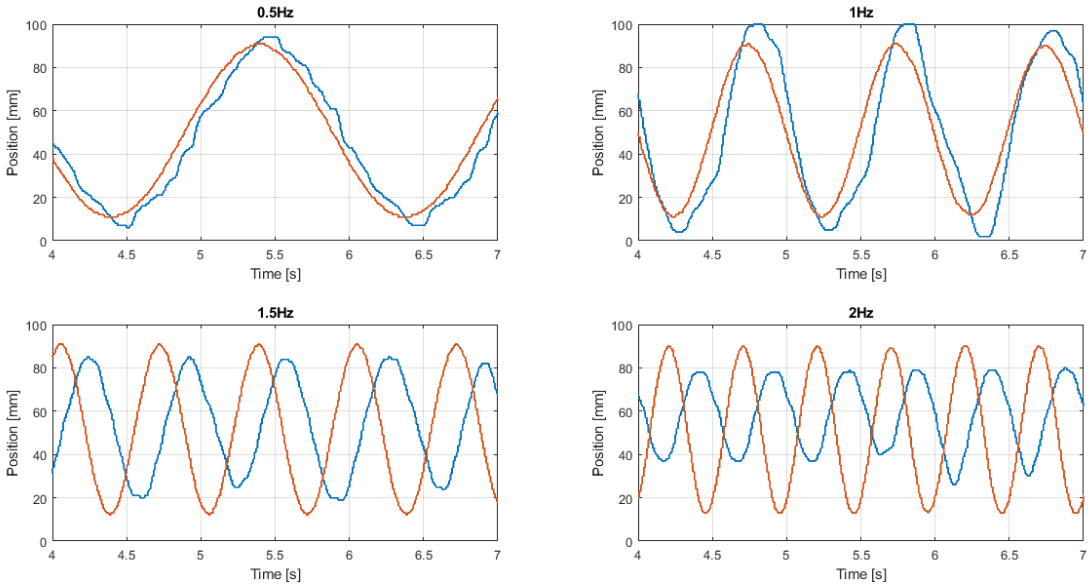


Figure 2.25: Tracking error - Forced DC Control

Trough the results, it is possible to see which the tracking error is appreciable around 1 Hz, after this frequency the error becomes considerable. In conclusion, the best results are obtained by the forced DC control. This because the valve is exploited to the best of its capabilities.
Introducing, also other experiments made with a saw-tooth wave the obtained results are presented in the following figures. The conclusions drawn are the same as before, so that with the Forced DC algorithm, a more correct behavior is obtained.

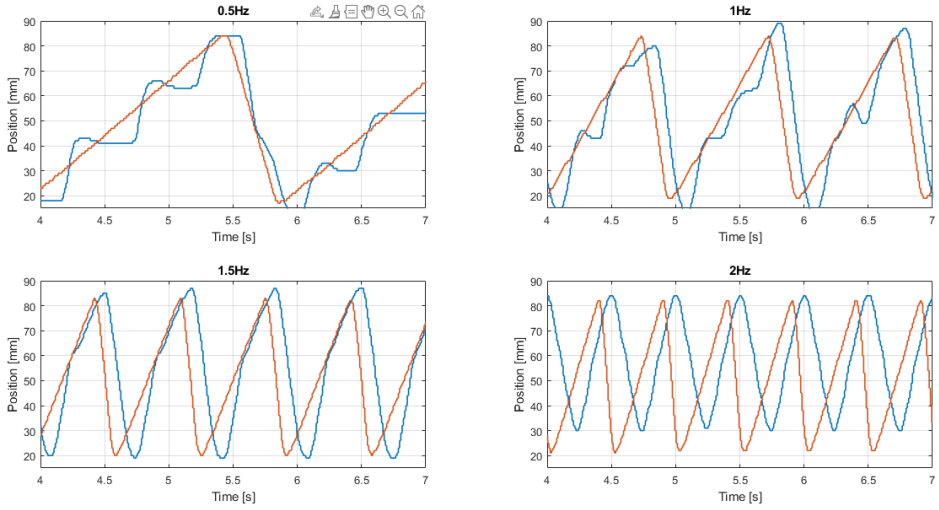


Figure 2.26: Tracking error - Saw-tooth wave - Base Control - pneumatic cylinder

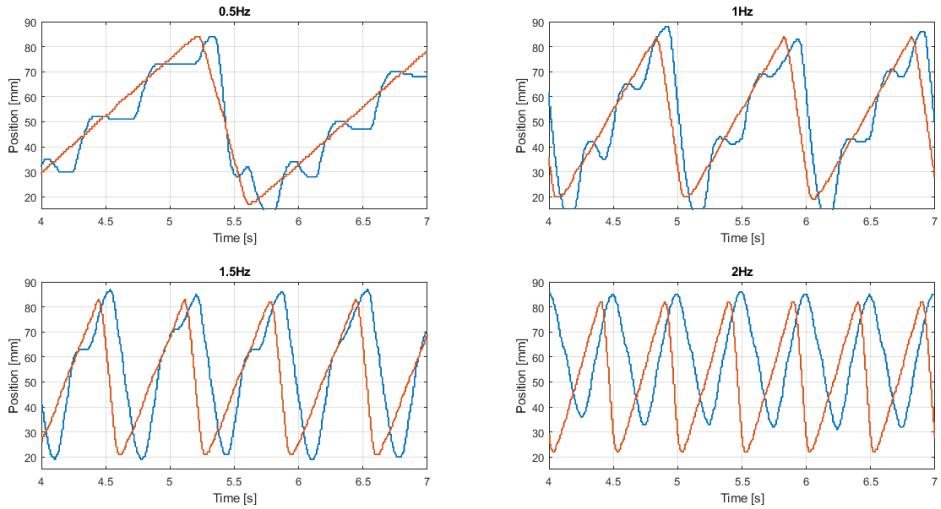


Figure 2.27: Tracking error - Saw-tooth wave - Forced DC Control - pneumatic cylinder

2.2 Proportional control using 3/2 valves NC

In this section, the proportional control is obtained through the using of 3/2 valves NC. The main differences between 2/2 valves and 3/2 valves are the flow rate and the work frequency.
The maximum work frequency of the 3/2 valves is 200 Hz lesser than the 2/2 valves. Taking into account these differences, experiments are made. In particular, these experiments are related to a base control (proportional) and to a forced DC.

2.2.1 Base Control

For the base control, the algorithm is based, as before, on a difference between *Setpoint*, which is the final point that the cylinder must achieve, and *Feedback*, which is the real position of the cylinder. The used variables are reported in the table below:

DATA BLOCKS

Static		
FEEDBACK	Int	0
SETPOINT	Int	0
NORMX_MIN	Int	1450
NORMX_MAX	Int	26450
SCALEX_MIN	Int	0
SCALEX_MAX	Int	100
NORMX_VALUE	Int	0
NORMX_OUTPUT	Real	0.0
ERROR	Int	0
ERROR_ABS	Int	0
INCREASE	Bool	false
DECREASE	Bool	false
DC_POS_0_100	Int	0
DC_NEG_0_100	Int	0
EMERGENCY/STOP	Bool	false
START	Bool	false

PLC VARIABLES

IW64	Word	%IW64
PWM_POS	Int	%QW1000
PWM_NEG	Int	%QW1002
OUTSTROKE	Bool	%Q0.1
INSTROKE	Bool	%Q0.2
MERKER_INCREASE	Bool	%M0.1
MERKER_DECREASE	Bool	%M0.2
MERKER EMERGENCY	Bool	%M0.3

Figure 2.28: Data block and PLC variables - 3/2 valves

In the first segment there is the transducer segment, deals of reading the IW64 input. Thanks to the use of the NORMX and SCALEX blocks it is possible to obtain the NORMX_VALUE. Also, into the block a minimum and a maximum are associated, in order to have a correct reading value between 0 mm and 100 mm for the piston's stroke.
Also in this case, an emergency segment is designed. After that, if this appears, it is possible to restart the project, by using the button *Start*.

NAME	USE
FEEDBACK	Int variable used to manage the stroke position
ERROR_ABS	Int variable used to carry out the converted value of the error coming from the ABS block
SETPOINT	Int variable used as reference
NORMX_MIN	Predetermined Int variable used as minimum element on the normx block
NORMX_MAX	Predetermined Int variable used as maximum value on the normx block
SCALEX_MIN	Predetermined Int variable used as minimum element on the scalex block
SCALEX_MAX	Predetermined Int variable used as maximum value on the scalex block
NORMX_VALUE	Int variable used as input value of the normx block
NORMX_OUTPUT	Real variable used to carry out the output value of the normx block
ERROR	Int variable calculated as difference between SETPOINT and FEEDBACK
INCREASE	Bool variable equal to TRUE when the system is in increasing phase
DECREASE	Bool variable equal to TRUE when the system is in decreasing phase
DC_POS_0_100	Int variable used to carry out the integer value of the Duty cycle in increasing phase
DC_NEG_0_100	Int variable used to carry out the integer value of the Duty cycle in decreasing phase
EMERGENCY/STOP	Bool variable used to manage the stopping phase
START	Bool variable used to start the system
PWM_POS	Int variable used to manage the PWM in increasing phase
PWM_NEG	Int variable used to manage the PWM in increasing phase
OUTSTROKE	Bool variable used to manage the output port Q0.1 of the PLC
INSTROKE	Bool variable used to manage the output port Q0.2 of the PLC
MERKER_INCREASE	Bool variable equal to TRUE when the system is in increasing phase
MERKER_DECREASE	Bool variable equal to TRUE when the system is in decreasing phase
MERKER_EMERGENCY	Bool variable equal to TRUE when the system is in emergency case
IW64	Int variable used to read the value coming from the input port IW64

Figure 2.29: Data block and PLC variables - USE - 3/2 valves

The most important segment is related to manage the outstroke and the instroke of the piston. Based on the comparison between the ERROR and zero, the comparison is the same as before. So, if the first rung is TRUE, the error is greater than zero and the outstroke is performed.
Otherwise, if the error is less than zero, the rung which is activated is the second and the instroke of the piston appears, after that the error is converted in an 'abs' value. It is noted that the position control for the 3/2 valves is different from that used for the 2/2 valves.
In particular, outstroke outlet and instroke outlet are absent.
So, thanks to *CTRL_PWM* block and its relative merkers, the pulse generators are activated in order to perform strokes, based on the DC.

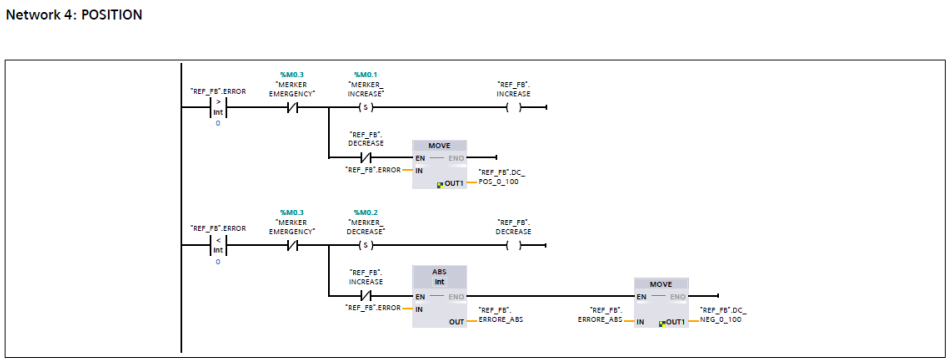


Figure 2.30: Network 4 - Position Control

Now, the final segment managed the emergency and the final position. Through the outputs Q0.1 and Q0.2 the PT01/PWM1 and the PT02/PWM2 are reset and so the piston is stopped, in any case.

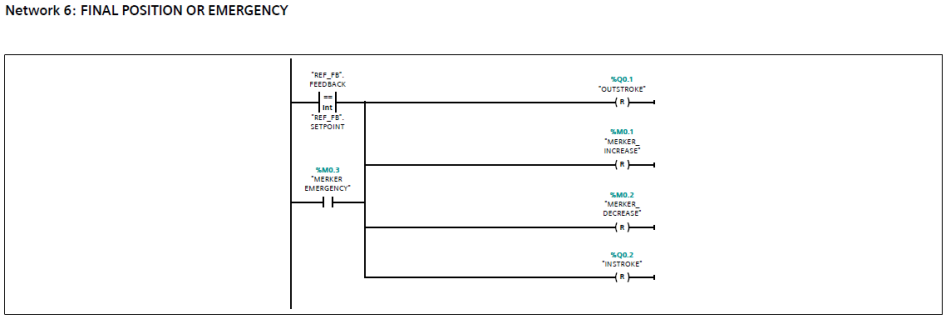


Figure 2.31: Network 7 - Final Position or Emergency

By simulation on TIA portal, the obtained response is:

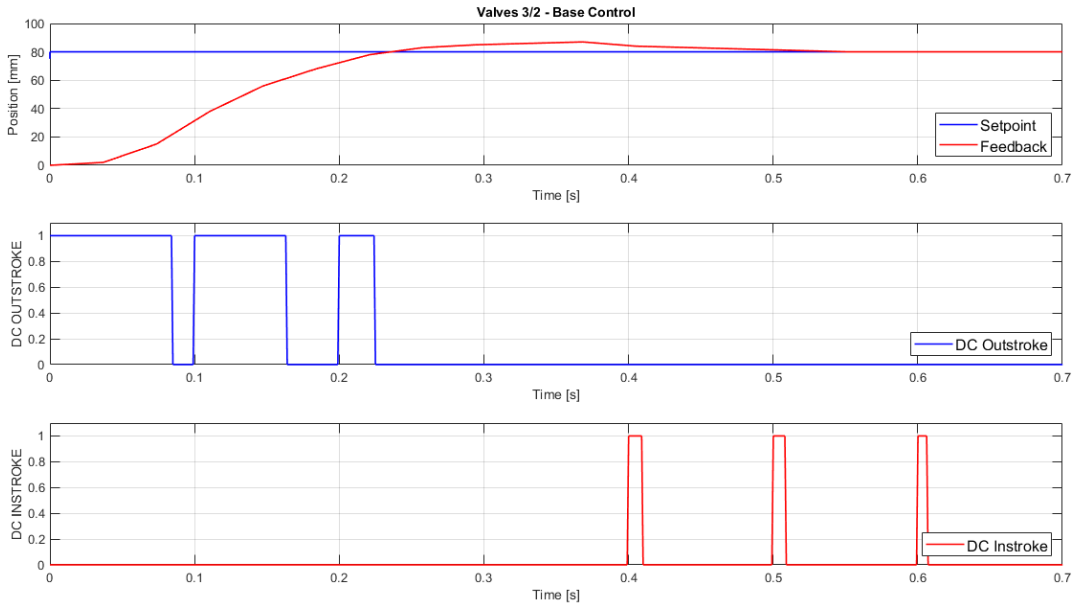


Figure 2.32: Dynamic response - Base control - 3/2 Valves - pneumatic cylinder

As shown in fig.2.32, the dynamic response of the real system is fast. Indeed, the overshoot is only one and there are not undershoots. Moreover, the settling time is very good with respect the use of 2/2 valves.
The obtained results are:

Overshoot %	Rise Time [s]	Settling Time [s]
8,75	0,234	0,551

Table 2.4: System response - Base control - 3/2 Valves - pneumatic cylinder

2.2.2 Control with forced duty cycle

In this section, the following control is made in order to improve the characteristic and the response of the base control. The used variables are:

DATA BLOCK

Static		
FEEDBACK	Int	0
SETPOINT	Int	0
NORMX_MIN	Int	1450
NORMX_MAX	Int	26450
SCALEX_MIN	Int	0
SCALEX_MAX	Int	100
NORMX_VALUE	Int	0
NORMX_OUTPUT	Real	0.0
ERROR	Int	0
ERROR ABS	Int	0
INCREASE	Bool	false
RANGE_INCREASE	Bool	false
RANGE_DECREASE	Bool	false
DECREASE	Bool	false
DC_POS_0_100	Int	0
DC_NEG_0_100	Int	0
EMERGENCY/STOP	Bool	false
START	Bool	false
RANGE_MIN	Int	0
RANGE_MAX	Int	0

PLC VARIABLES

IW66	Word	%IW66
PWM_POS	Int	%QW1000
PWM_NEG	Int	%QW1002
OUTSTROKE	Bool	%Q0.1
INSTROKE	Bool	%Q0.2
MERKER_INCREASE	Bool	%M0.1
MERKER_DECREASE	Bool	%M0.2
MERKER EMERGENCY	Bool	%M0.3

Figure 2.33: Data block and PLC variables - DC Forced - 3/2 valves

Also, the use of variables are explained in the fig.2.34.

In this case is chosen to use the maximum flow air in order to speed up the system trough the forced DC.
In this way, the DC reached the 100% and the opening time of the valve is greater. By the using of different threshold the DC is forced. The key idea of using thresholds is the same as shown in the forced DC algorithm, using 2/2 valves.
Only 2 segments are changed from the design used on the TIA portal, with 2/2 valves. In particular, are the two segments treated in the base control, using 3/2 valves (fig.2.30 and fig.2.31).

NAME	USE
FEEDBACK	Int variable used to manage the stroke position
ERROR_ABS	Int variable used to carry out the converted value of the error coming from the ABS block
SETPOINT	Int variable used as reference
NORMX_MIN	Predetermined Int variable used as minimum element on the normx block
NORMX_MAX	Predetermined Int variable used as maximum value on the normx block
SCALEX_MIN	Predetermined Int variable used as minimum element on the scalex block
SCALEX_MAX	Predetermined Int variable used as maximum value on the scalex block
NORMX_VALUE	Int variable used as input value of the normx block
NORMX_OUTPUT	Real variable used to carry out the output value of the normx block
ERROR	Int variable calculated as difference between SETPOINT and FEEDBACK
INCREASE	Bool variable equal to TRUE when the system is in increasing phase
DECREASE	Bool variable equal to TRUE when the system is in decreasing phase
RANGE_INCREASE	Bool variable used to manage the use of the forced duty cycle in increasing phase. It's setted when the DC is forced.
RANGE_DECREASE	Bool variable used to manage the use of the forced duty cycle in decreasing phase. It's setted when the DC is forced.
RANGE_MIN	Int variable used as lower threshold of the INRANGE block
RANGE_MAX	Int variable used as upper threshold of the INRANGE block
DC_POS_0_100	Int variable used to carry out the integer value of the Duty cycle in increasing phase
DC_NEG_0_100	Int variable used to carry out the integer value of the Duty cycle in decreasing phase
EMERGENCY/STOP	Bool variable used to manage the stopping phase
START	Bool variable used to start the system
PWM_POS	Int variable used to manage the PWM in increasing phase
PWM_NEG	Int variable used to manage the PWM in increasing phase
OUTSTROKE	Bool variable used to manage the output port Q0.1 of the PLC
INSTROKE	Bool variable used to manage the output port Q0.2 of the PLC
MERKER_INCREASE	Bool variable equal to TRUE when the system is in increasing phase
MERKER_DECREASE	Bool variable equal to TRUE when the system is in decreasing phase
MERKER_EMERGENCY	Bool variable equal to TRUE when the system is in emergency case
IW66	Int variable used to read the value coming from the input port IW66

Figure 2.34: Data block and PLC variables - USE - DC Forced - 3/2 valves

As for the previous algorithm using 2/2 valves, it is noted that as the range increases, the rise time considerably worse. As far as, concerned overshoot and settling times, they improve. Instead, in the case in which the range decreases the behavior is opposite.
As in fig.2.35, it is possible to observe that the system response matched with the algorithm and the system response is better than the system response obtained by the base control.

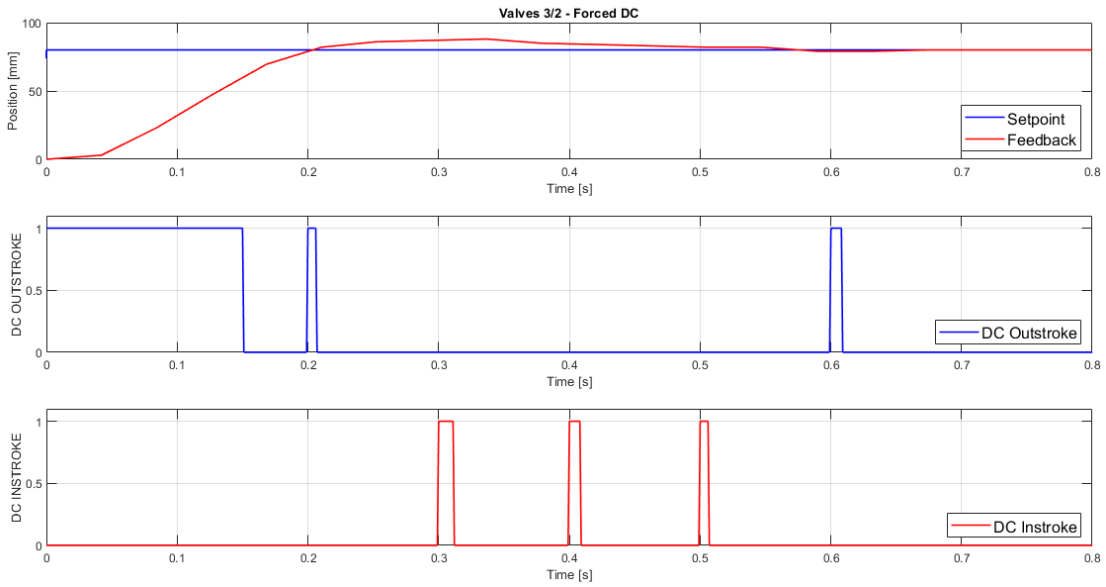


Figure 2.35: Response with a range ±15 [mm] - 3/2 Valves

The DC is forced between values less than 65 mm and higher than 95 mm the control. Otherwise inside these value the control is proportional.

Range [mm]	Overshoot %	Rise Time [s]	Settling Time [s]
±15	10	0,203	0,67
±20	9	0,215	0,642
±30	7	0,228	0,605

Table 2.5: System response - Forced DC - 3/2 Valves - pneumatic cylinder

In the table 2.5 are shown other experiments with different ranges.
So, it is possible to observe that the overshoot and the settling time decrease with respect to the range increasing. On the other hand, the rise time is worsened, as we expect.

The comparison between base control and the forced DC control algorithms is shown in the table 2.6, the first has a better settling time and overshoot in terms

of response; while, the second one has a more efficient rise time behaviour, as previously stated.

Algorithm	Overshoot %	Rise Time [s]	Settling Time [s]
Base Control	8,75	0,234	0,551
DC Forced	10	0,203	0,67

Table 2.6: Comparison algorithms pneumatic cylinder - 3/2 Valves

2.3 Comparison between 2/2 valves NC and 3/2 valves NC

In this section is analyzed the comparison between the base control using 2/2 valves and 3/2 valves and forced DC control using 2/2 valves and 3/2 valves.

The main different that can be evaluated are:

- Overshoot
- Rise time
- Settling time

So, between this experiment and the experiment using 2/2 valves the different behaviours are analyzed using a base control:

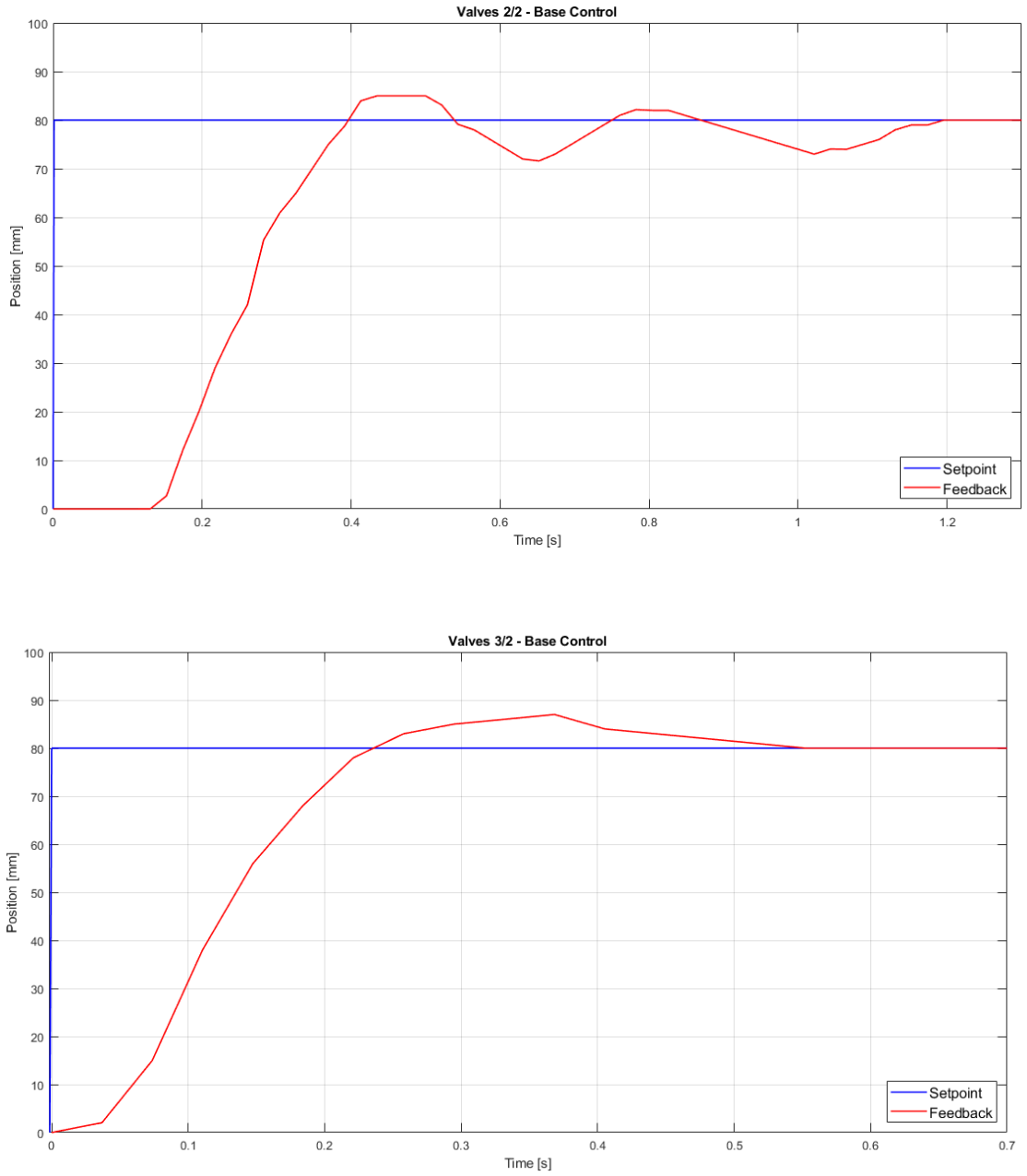


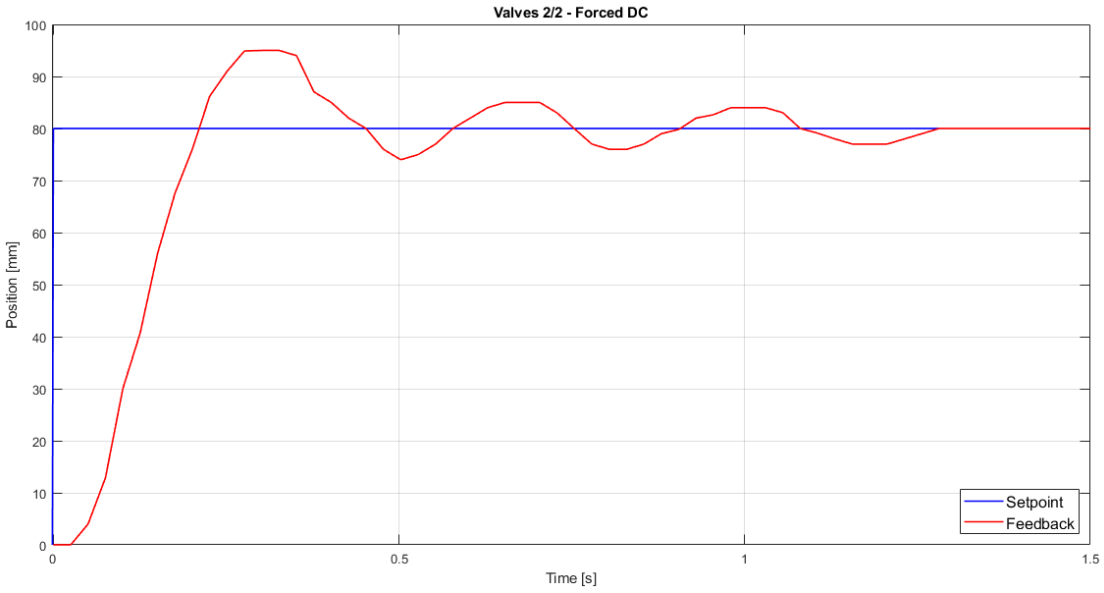
Figure 2.36: Base Control comparison - pneumatic cylinder

By the table 2.7 it is possible to note that with using the 2/2 valves the dynamic response shown only a better pick, otherwise using 3/2 valves rise time and settling time are improved. Moreover, the base control presents only one overshoot, while using 2/2 valves the response have many overshoot and also undershoots.

Algorithm	Overshoot %	Rise Time [s]	Settling Time [s]
2/2 Base Control	6,25	0,396	1,195
3/2 Base Control	8,75	0,234	0,551

Table 2.7: Base control comparison- pneumatic cylinder

Now, the analyzed response is for the forced DC using 2/2 and 3/2 valves. In this particular case, the responses are:



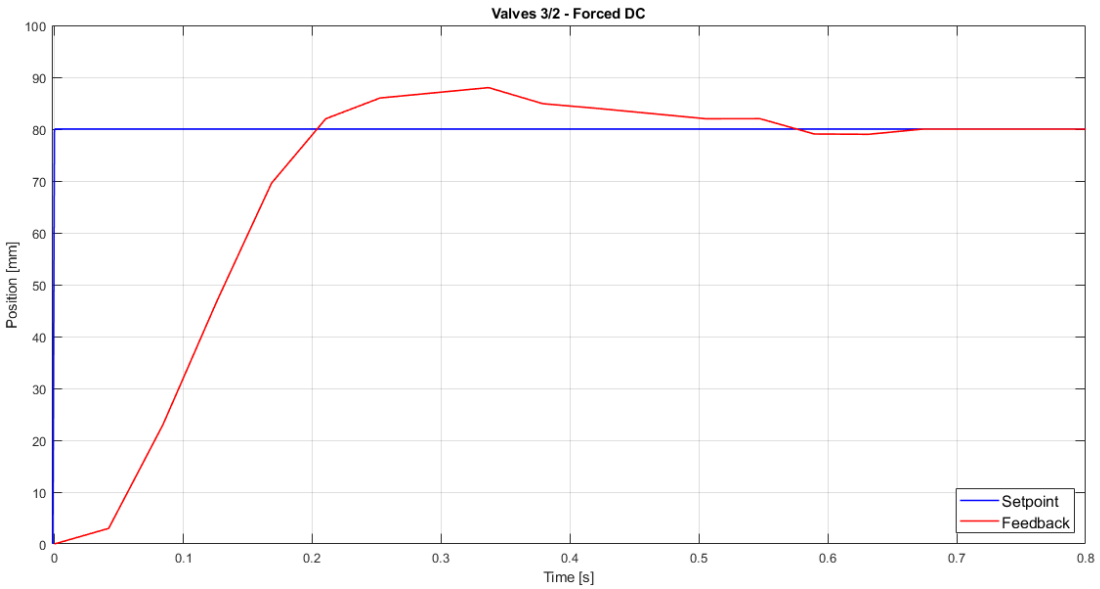


Figure 2.37: Forced DC comparison - pneumatic cylinder

As before, with the 3/2 valves the response are better. In particular, in terms of transient the response is better for the rise time and settling time. As regards the overshoot, now, is improved using 3/2 valves, as shown in the table below:

Algorithm	Overshoot %	Rise Time [s]	Settling Time [s]
2/2 Forced DC	18,7	0,31	1,795
3/2 Forced DC	10	0,203	0,67

Table 2.8: Forced DC comparison - pneumatic cylinder

So, in conclusion it is possible to say that with the 3/2 valves the response is better, with an acceptable overshoot and rise time and settling time comparable.

Algorithm	Overshoot %	Rise Time [s]	Settling Time [s]
2/2 Base Control	6,25	0,396	1,195
3/2 Base Control	8,75	0,234	0,551
2/2 Forced DC	18,7	0,31	1,795
3/2 Forced DC	10	0,203	0,67

Table 2.9: Final comparison - pneumatic cylinder

In particular, the best results are obtained with a 3/2 base proportional control.

Chapter 3

Proportional algorithm with oil-pneumatic cylinder

3.1 Proportional control using 2/2 NC valves

In this section, using the oil-pneumatic cylinder, the controls which are used are: a base control with a proportional algorithm and a forced DC control, as before.
With this oil-pneumatic cylinder the expected results are different with respect the pneumatic cylinder experiments results.
Because this cylinder has an hydraulic damper, it affects its dynamic behaviors.
Based on the designed algorithms the main characteristic what, perhaps, changes are the response of rise time and overshoot. In particular, there will be fewer overshoots and undershoots in the system, at the same work frequency (10 Hz).

3.1.1 Base Control

The main configuration are the same. So, the difference between *Setpoint* and *Feedback* is done. The designed segments are:

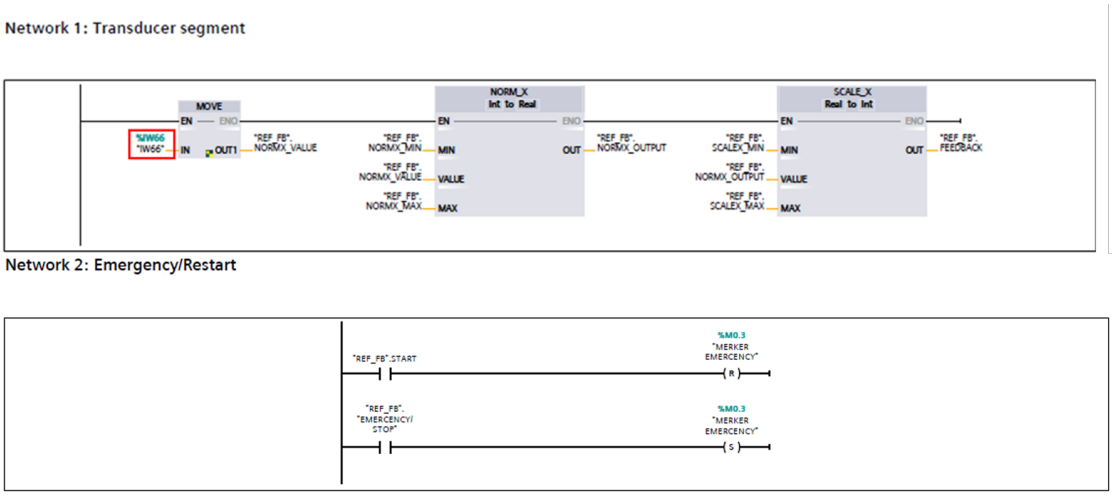


Figure 3.1: Networks 1-2

Now, the used transducer is the IW66 port of the PLC. In order to have the signal *Feedback* between 0 and 100 [mm], this $NORM\_X\_VALUE$ signal is normalized and scaled. Also, in order to manage the $EMERGENCY/STOP$ signal to block the system in any cases, the Network 2 is designed. So, by the use of HMI's buttons it's possible to asserting these rungs. Once, the emergency phase is over, by the Start the system is reactivated.

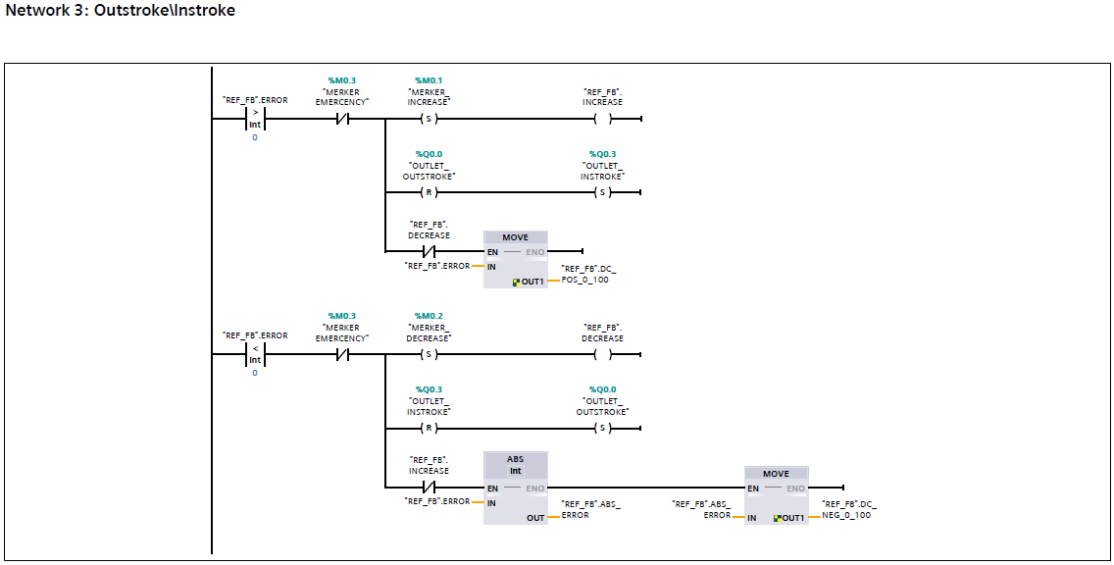


Figure 3.2: Networks 3

After that , the Segment 3 regards the comparison between *Error* and zero

value. If the *Error* is greater than zero the rung which works is the first. While, if the *Error* is less than zero works the second one. The control is proportional in both cases and the *Error* value is moved toward the control PWM.

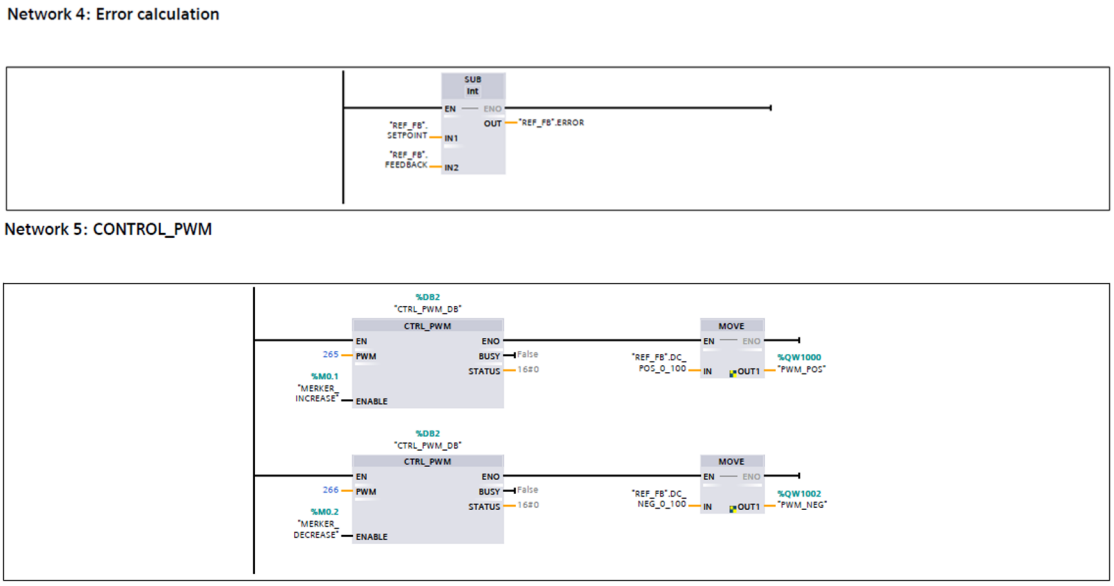


Figure 3.3: Networks 4-5

By the using of the pulse generators, two squares waves are created, one in the increasing phase and one other in the decreasing phase. The used ports are the same treated in the configuration in chapter 2. The $CTRL\_PWM$ block allow to activate the pulse train and $MOVE$ block to pass the signal to the output ports Q0.1 and Q0.2.

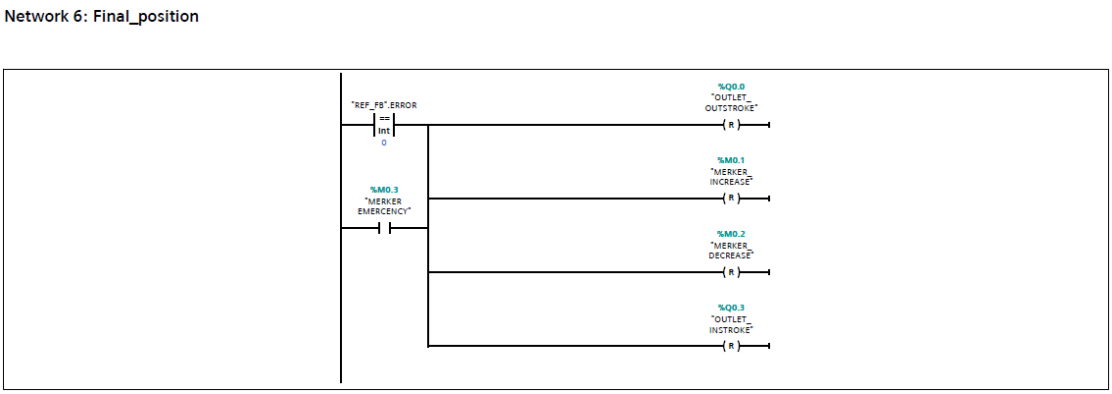


Figure 3.4: Networks 6

The final position is reached when the *Error* value is equal to zero, so all the

ports are reset. Moreover, this segment is used for the emergency case. As regards the PLC variables and the Data block used are:

DATA BLOCK

Static		
FEEDBACK	Int	0
SETPOINT	Int	0
NORMX_MIN	Int	6680
NORMX_MAX	Int	27000
SCALEX_MIN	Int	0
SCALEX_MAX	Int	100
NORMX_VALUE	Int	0
NORMX_OUTPUT	Real	0.0
ERROR	Int	0
ERROR_ABS	Int	0
INCREASE	Bool	false
DECREASE	Bool	false
DC_POS_0_100	Int	0
DC_NEG_0_100	Int	0
EMERGENCY/STOP	Bool	false
START	Bool	false

PLC VARIABLES

IW66	Word	%IW66
PWM_POS	Int	%QW1000
PWM_NEG	Int	%QW1002
OUTLET_INSTROKE	Bool	%Q0.3
OUTLET_OUTSTROKE	Bool	%Q0.0
OUTSTROKE	Bool	%Q0.1
INSTROKE	Bool	%Q0.2
MERKER_INCREASE	Bool	%M0.1
MERKER_DECREASE	Bool	%M0.2
MERKER EMERGENCY	Bool	%M0.3

Figure 3.5: Data Block and PLC variables - oil-pneumatic cylinder

NAME	USE
FEEDBACK	Int variable used to manage the stroke position
ABS_ERROR	Int variable used to carry out the converted value of the error coming from the ABS block
SETPOINT	Int variable used as reference
NORMX_MIN	Predetermined Int variable used as minimum element on the normx block
NORMX_MAX	Predetermined Int variable used as maximum value on the normx block
SCALEX_MIN	Predetermined Int variable used as minimum element on the scalex block
SCALEX_MAX	Predetermined Int variable used as maximum value on the scalex block
NORMX_VALUE	Int variable used as input value of the normx block
NORMX_OUTPUT	Real variable used to carry out the output value of the normx block
ERROR	Int variable calculated as difference between SETPOINT and FEEDBACK
INCREASE	Bool variable equal to TRUE when the system is in increasing phase
DECREASE	Bool variable equal to TRUE when the system is in decreasing phase
DC_POS_0_100	Int variable used to carry out the integer value of the Duty cycle in increasing phase
DC_NEG_0_100	Int variable used to carry out the integer value of the Duty cycle in decreasing phase
EMERGENCY/STOP	Bool variable used to manage the stopping phase
START	Bool variable used to start the system
PWM_POS	Int variable used to manage the PWM in increasing phase
PWM_NEG	Int variable used to manage the PWM in increasing phase
OUTLET_INSTROKE	Bool variable used to manage the output port Q0.3 of the PLC
OUTLET_OUTSTROKE	Bool variable used to manage the output port Q0.0 of the PLC
OUTSTROKE	Bool variable used to manage the output port Q0.1 of the PLC
INSTROKE	Bool variable used to manage the output port Q0.2 of the PLC
MERKER_INCREASE	Bool variable equal to TRUE when the system is in increasing phase
MERKER_DECREASE	Bool variable equal to TRUE when the system is in decreasing phase
MERKER_EMERGENCY	Bool variable equal to TRUE when the system is in emergency case
IW66	Int variable used to read the value coming from the input port IW66

Figure 3.6: Data Block and PLC variables - USE - oil-pneumatic cylinder

As shown in the fig:3.5 the used signals are the same which are used in the previous algorithm with the pneumatic cylinder, expect for the transducer signal IW66. The experiments resulted for this base control with a proportional pure control are:

Overshoot %	Rise Time [s]	Settling Time [S]
7,5	0,83	1,26

Table 3.1: System response - Base control - 2/2 Valves - oil-pneumatic cylinder

By the fig.3.7 the obtained graph shows the system response:

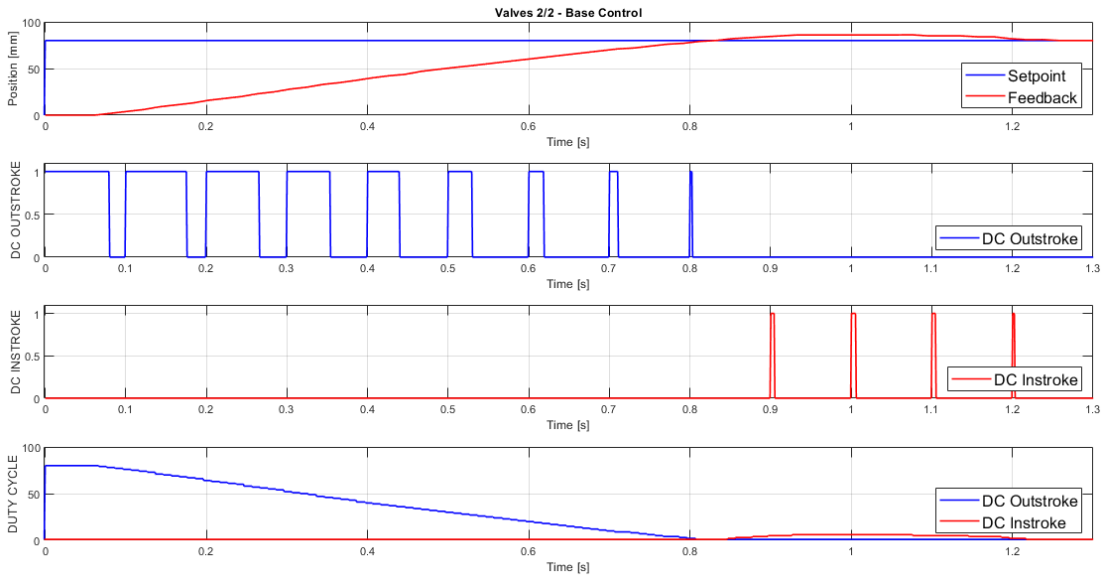


Figure 3.7: Dynamic response - oil-pneumatic cylinder

This lead to show that the initial assumptions are correct. Indeed, the overshoot is only one and the undershoot is not present. So, the system behaviour is more regular, thanks to the help of the damper.
Also, the settling time is more or less the same, but the rise time is greater than the rise time of the pneumatic cylinder. This problem, with the same pneumatic circuit, is given by the fact that the oil-pneumatic cylinder has a constant velocity, while the pneumatic cylinder has no damping and therefore reaches higher speeds.

3.1.2 Control with forced duty cycle

In the following section, the control system is based on the forced DC, in order to improved the system response. By the using of the TIA portal, to achieve the desired position (*Setpoint*), the used signals are:

NAME	USE
FEEDBACK	Int variable used to manage the stroke position
ABS_ERROR	Int variable used to carry out the converted value of the error coming from the ABS block
SETPOINT	Int variable used as reference
NORMX_MIN	Predetermined Int variable used as minimum element on the normx block
NORMX_MAX	Predetermined Int variable used as maximum value on the normx block
SCALEX_MIN	Predetermined Int variable used as minimum element on the scalex block
SCALEX_MAX	Predetermined Int variable used as maximum value on the scalex block
NORMX_VALUE	Int variable used as input value of the normx block
NORMX_OUTPUT	Real variable used to carry out the output value of the normx block
ERROR	Int variable calculated as difference between SETPOINT and FEEDBACK
INCREASE	Bool variable equal to TRUE when the system is in increasing phase
DECREASE	Bool variable equal to TRUE when the system is in decreasing phase
DC_POS_0_100	Int variable used to carry out the integer value of the Duty cycle in increasing phase
DC_NEG_0_100	Int variable used to carry out the integer value of the Duty cycle in decreasing phase
EMERGENCY/STOP	Bool variable used to manage the stopping phase
START	Bool variable used to start the system
INRANGE_POS	Int variable used as upper threshold of the INRANGE block
INRANGE_NEG	Int variable used as lower threshold of the INRANGE block
OUTPUT_IN_POS	Bool variable used to manage the use of the forced duty cycle in increasing phase. It's setted when the DC is forced.
OUTPUT_IN_NEG	Bool variable used to manage the use of the forced duty cycle in decreasing phase. It's setted when the DC is forced.
PWM_POS	Int variable used to manage the PWM in increasing phase
PWM_NEG	Int variable used to manage the PWM in increasing phase
OUTLET_INSTROKE	Bool variable used to manage the output port Q0.3 of the PLC
OUTLET_OUTSTROKE	Bool variable used to manage the output port Q0.0 of the PLC
OUTSTROKE	Bool variable used to manage the output port Q0.1 of the PLC
INSTROKE	Bool variable used to manage the output port Q0.2 of the PLC
MERKER_INCREASE	Bool variable equal to TRUE when the system is in increasing phase
MERKER_DECREASE	Bool variable equal to TRUE when the system is in decreasing phase
MERKER_EMERGENCY	Bool variable equal to TRUE when the system is in emergency case
IW66	Int variable used to read the value coming from the input port IW66

Figure 3.8: Data block and PLC variables - DC forced control - USE - oil-pneumatic cylinder

So, the used algorithm, is designed to improve the control. The reason why is to exploit to the maximum the DC. Before, with the base control, the maximum of the reached DC is around the 80%, this value depends by the error.

Now, the final DC achieve the 100% in a forced way. In particular, it is use a threshold, which allows to use a proportional control between a particular range. So, thanks to the use of a OUT_RANGE and a IN_RANGE blocks, on the TIA portal, the control is made.
The control is made in three different cases:

- Threshold = ± 15
- Threshold = ± 20
- Threshold = ± 30

These three experiments are made in order to have a comparison with the use of proportional control.
From the previous experiments, made with the pneumatic cylinder, the expected results are the improving of overshoot and settling time, and the worsening of the rise time, with the increase of the range.

The used segments are the same of the base control, but with the addition of some new blocks, in particular:

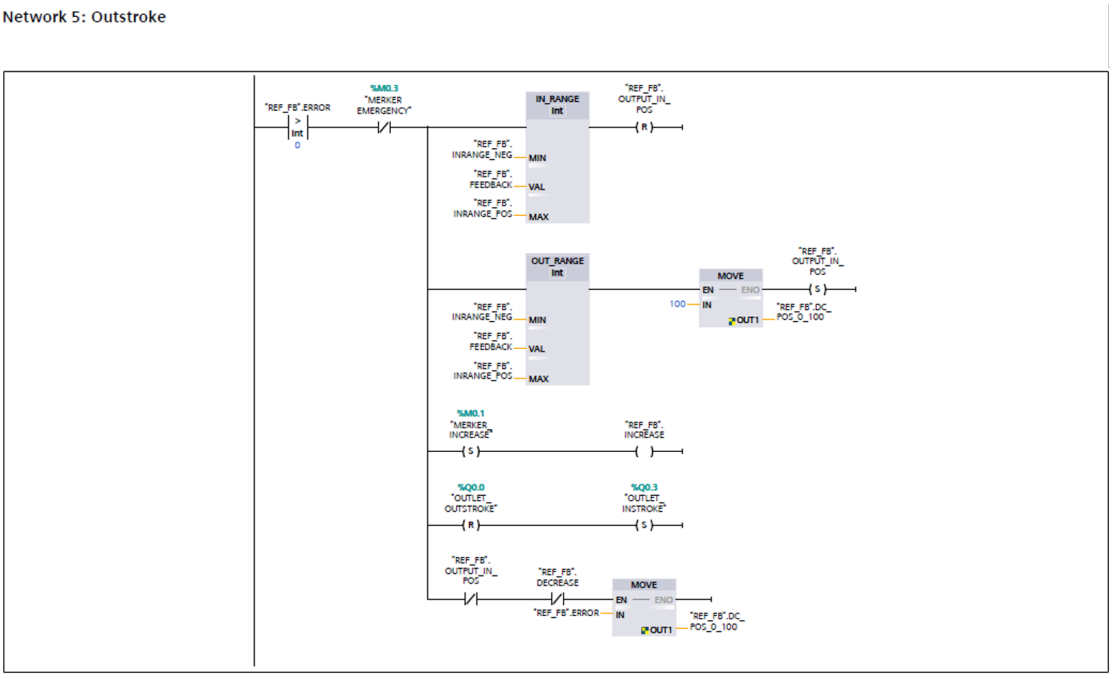


Figure 3.9: Outstroke - DC forced control - oil-pneumatic cylinder

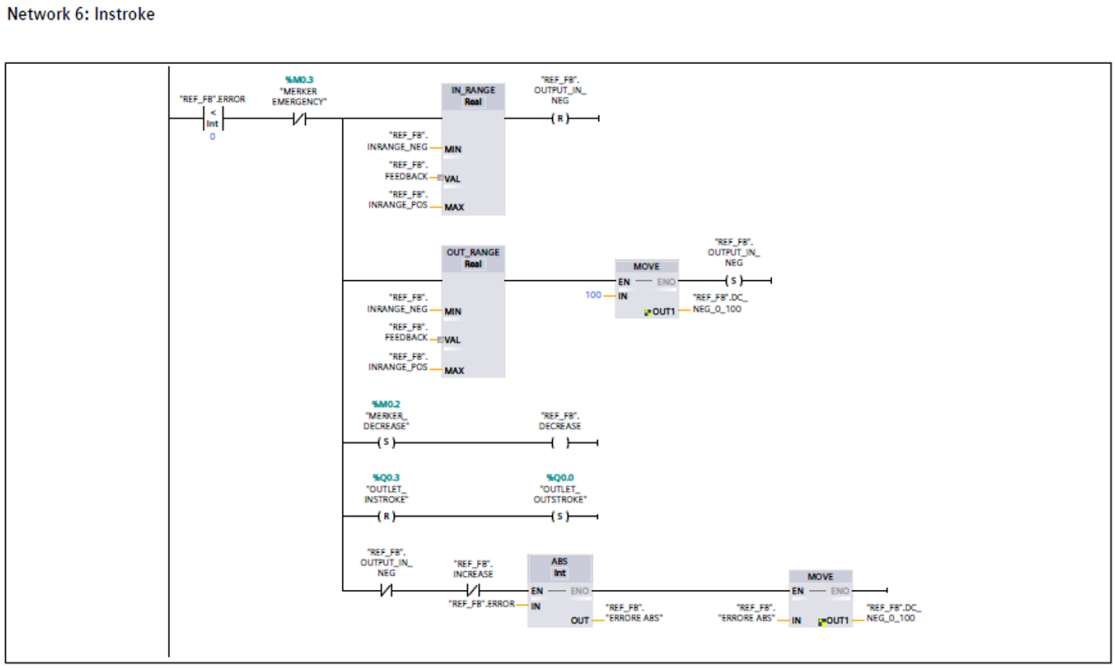


Figure 3.10: Instroke - DC forced control - oil-pneumatic cylinder

Moreover, in order to calculate the difference and the sum between *Setpoint* and threshold , the used segment is:

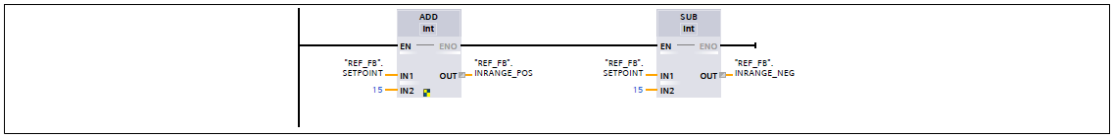


Figure 3.11: Limitation range - oil-pneumatic cylinder

In this case, the maximum opening time value of the valves decrease. As in the table below, it possible to note the made different experiments results, based on the use of the different threshold listed before:

Range [mm]	Overshoot %	Rise Time [s]	Settling Time [S]
±15	13,75	0,451	1,26
±20	11,25	0,596	1,267
±30	10,5	0,61	1,255

Table 3.2: System response - Forced DC - 2/2 Valves - oil-pneumatic cylinder

The system response is shown in fig.3.12, by the figure is possible to note the behaviour of the system. So, the initial hypothesis were correct. In fact, the rise time is worsted with respect the base control behaviour, while the overshoot and the settling time are better.

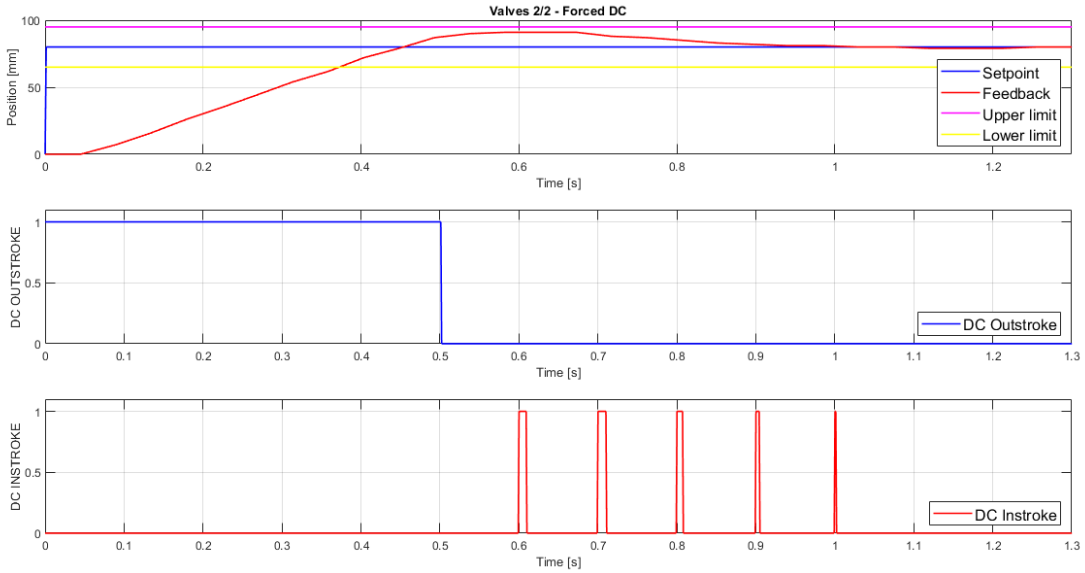


Figure 3.12: Response with a range ±15 - oil-pneumatic cylinder

The response is characterized by an overshoot and a small undershoot. In particular, between the pink line and the yellow line the control is completely proportional. While, out of these lines, the used algorithm is the forced one.
The comparison between the two used algorithms: Base control and Forced DC Control is shown as follows:

Algorithm	Overshoot %	Rise Time [s]	Settling Time [S]
Base Control	7,5	0,83	1,26
Forced DC Control	13,75	0,451	1,267

Table 3.3: Comparison algorithms oil-pneumatic cylinder - 2/2 Valves

As shown in table 3.3 the DC forced algorithm leads, obviously, to better results. The valves works at 100%, in a small amount of time.
So, it is possible to note a decrease of rise time, therefore, settling time is comparable between the two algorithms, and regards the overshoot is better in the base control, as expected.

3.1.3 Evaluation of tracking error

The tracking error is analyze in this section. By the use of the oscilloscope (fig.1.5), the tracking error is analyzed for both cases: the base control case and the forced DC one.
The evaluation of the error is made with a sinusoidal input, in order to obtain a sine wave that the PLC's input ports accept, so an offset is introduced.
On the TIA portal the segment used to recognize the error is:

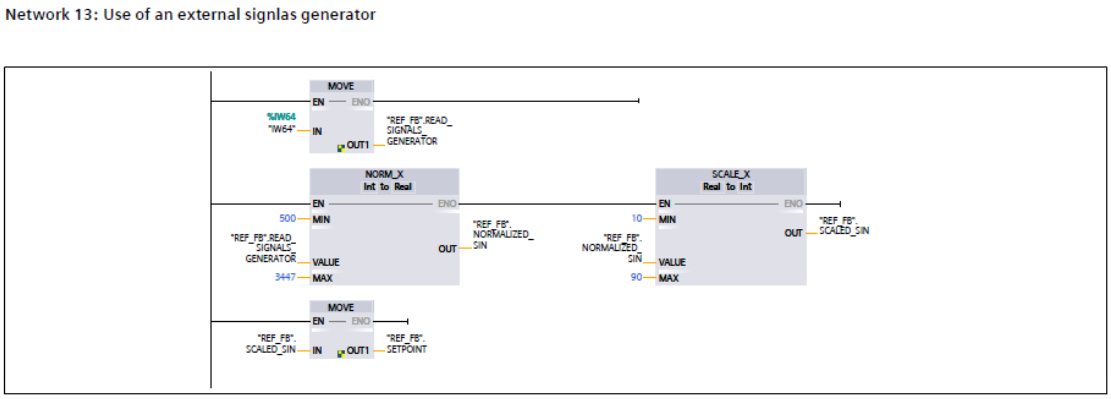


Figure 3.13: External signals generator - oil-pneumatic cylinder

In this way the input port IW64 is normalized and scaled. After that, changing the sine wave frequencies different experiments are made.

In particular, the used frequencies are:

- 0.5Hz
- 1Hz
- 1.5Hz
- 2Hz

For the base control the tracking error response is shown in fig.3.14, while for the forced DC the response is shown in fig.3.15 .
Analyzing these different behaviour the tracking error lead to better results with the Forced DC control.
Also, it is possible to note that the tracking error is appreciable around 0.6 Hz. So, after this frequency, the error is very huge. Moreover, having a damper, this cylinder lead to a worst work frequency with respect the pneumatic cylinder.

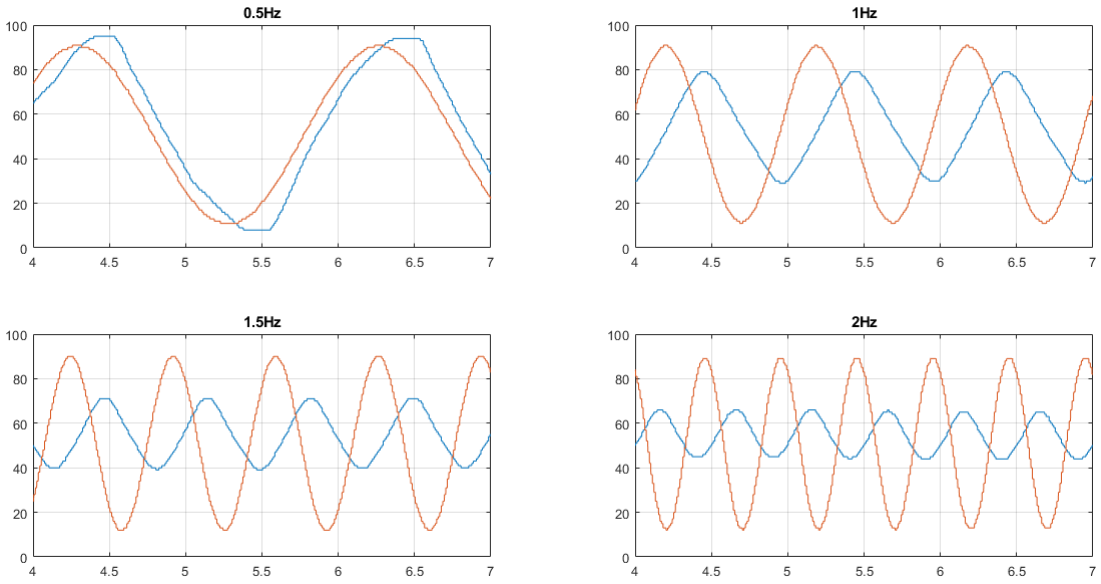


Figure 3.14: Tracking error - Sine wave - Base Control - oil-pneumatic cylinder

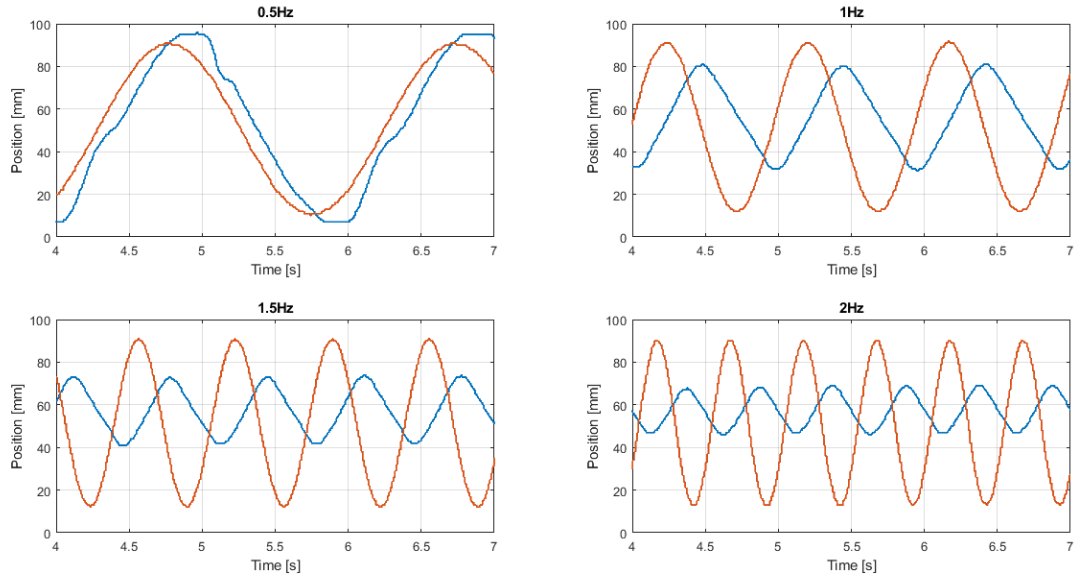


Figure 3.15: Tracking error - Sine wave - Forced DC Control - oil-pneumatic cylinder

Moreover, using not a sine wave, but with a saw-tooth wave, the best performances are once again obtained with the forced DC.

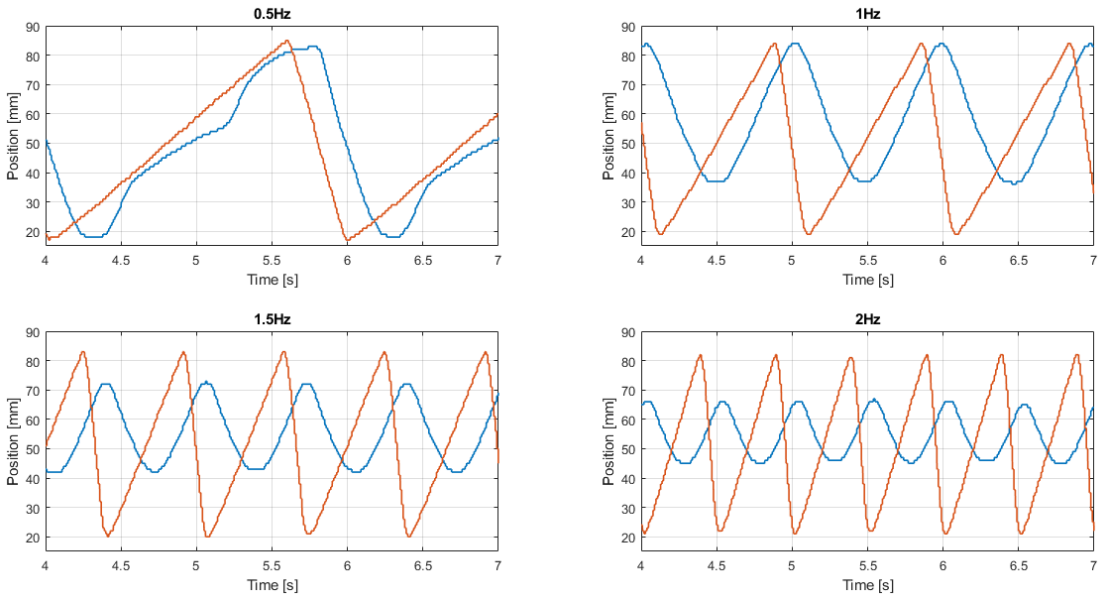


Figure 3.16: Tracking error - Saw-tooth wave - Base Control - oil-pneumatic cylinder

The response of the tracking error obtained using base and forced DC control are shown in the fig.3.16 and in fig.3.17

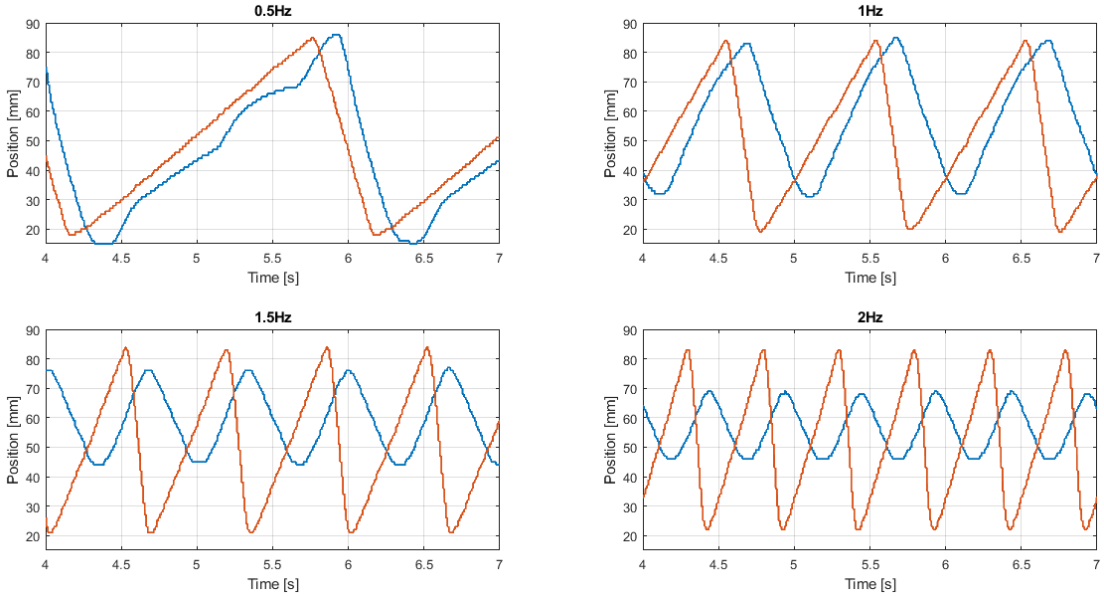


Figure 3.17: Tracking error - Saw-tooth wave - Forced DC Control - oil-pneumatic cylinder

3.2 Proportional control using 3/2 NC valves

In this section, using the oil-pneumatic cylinder, the controls which are used are: a base control with a proportional algorithm and a forced DC control, but using the 3/2 NC valves.

Figure 3.18: Oil-pneumatic cylinder

3.2.1 Base Control

Now, the cylinder presents the hydraulic damper so the expected results are different with respect the using of 2/2 valves. By the figure below, it is possible to see the used variables:

DATA BLOCK

Static		
FEEDBACK	Int	0
SETPOINT	Int	0
NORMX_MIN	Int	6680
NORMX_MAX	Int	27000
SCALEX_MIN	Int	0
SCALEX_MAX	Int	100
NORMX_VALUE	Int	0
NORMX_OUTPUT	Real	0.0
ERROR	Int	0
ERROR_ABS	Int	0
INCREASE	Bool	false
DECREASE	Bool	false
DC_POS_0_100	Int	0
DC_NEG_0_100	Int	0
EMERGENCY/STOP	Bool	false
START	Bool	false

PLC VARIABLES

IW66	Word	%IW66
PWM_POS	Int	%QW1000
PWM_NEG	Int	%QW1002
OUTLET_INSTROKE	Bool	%Q0.3
OUTLET_OUTSTROKE	Bool	%Q0.0
OUTSTROKE	Bool	%Q0.1
INSTROKE	Bool	%Q0.2
MERKER_INCREASE	Bool	%M0.1
MERKER_DECREASE	Bool	%M0.2
MERKER EMERGENCY	Bool	%M0.3

Figure 3.19: Data block and PLC variables - Base control - 3/2 valves

NAME	USE
FEEDBACK	Int variable used to manage the stroke position
ERROR_ABS	Int variable used to carry out the converted value of the error coming from the ABS block
SETPOINT	Int variable used as reference
NORMX_MIN	Predetermined Int variable used as minimum element on the normx block
NORMX_MAX	Predetermined Int variable used as maximum value on the normx block
SCALEX_MIN	Predetermined Int variable used as minimum element on the scalex block
SCALEX_MAX	Predetermined Int variable used as maximum value on the scalex block
NORMX_VALUE	Int variable used as input value of the normx block
NORMX_OUTPUT	Real variable used to carry out the output value of the normx block
ERROR	Int variable calculated as difference between SETPOINT and FEEDBACK
INCREASE	Bool variable equal to TRUE when the system is in increasing phase
DECREASE	Bool variable equal to TRUE when the system is in decreasing phase
DC_POS_0_100	Int variable used to carry out the integer value of the Duty cycle in increasing phase
DC_NEG_0_100	Int variable used to carry out the integer value of the Duty cycle in decreasing phase
EMERGENCY/STOP	Bool variable used to manage the stopping phase
START	Bool variable used to start the system
PWM_POS	Int variable used to manage the PWM in increasing phase
PWM_NEG	Int variable used to manage the PWM in increasing phase
OUTSTROKE	Bool variable used to manage the output port Q0.1 of the PLC
INSTROKE	Bool variable used to manage the output port Q0.2 of the PLC
MERKER_INCREASE	Bool variable equal to TRUE when the system is in increasing phase
MERKER_DECREASE	Bool variable equal to TRUE when the system is in decreasing phase
MERKER_EMERGENCY	Bool variable equal to TRUE when the system is in emergency case
IW66	Int variable used to read the value coming from the input port IW66

Figure 3.20: Data block and PLC variables - Base control - 3/2 valves - USE

By the using of adder and subtract the error is calculated. Based on the results, different segments are activated. If the error is higher than zero, the piston reached the outstroke position, otherwise the piston reached the instroke one.
In particular, to see in details the designed segments , in order to avoid repetition, please refer to the paragraph above, 3.1.1. The dynamic response of this system is:

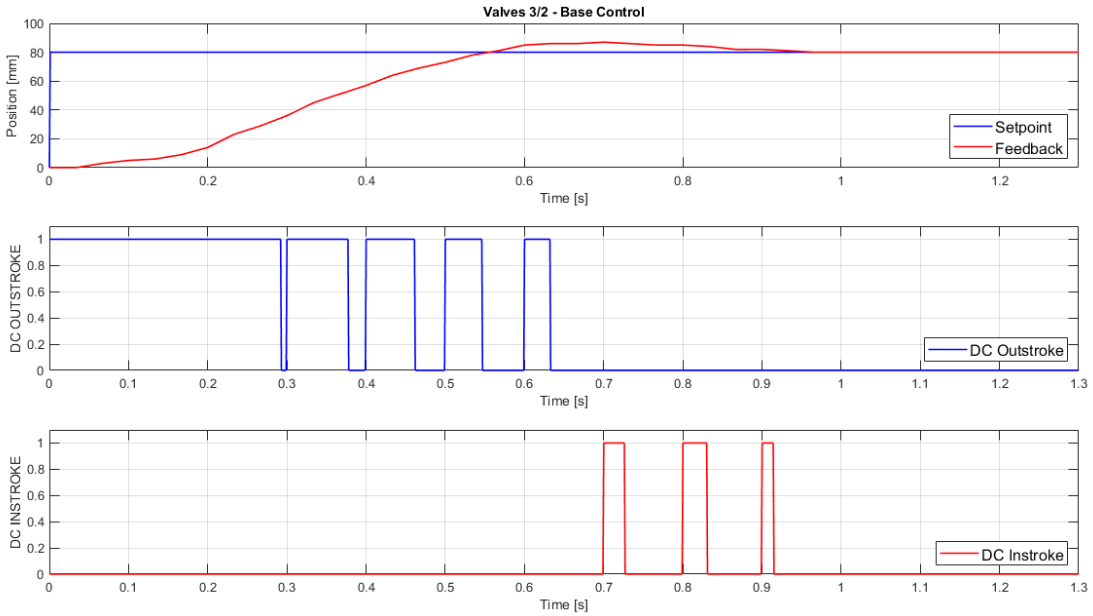


Figure 3.21: Dynamic response - Base control - 3/2 Valves - oil-pneumatic cylinder

So, the obtained results are:

Overshoot %	Rise Time [s]	Settling Time [s]
8,75	0,227	0,537

Table 3.4: System response - Base control - 3/2 Valves - oil-pneumatic cylinder

The system is very fast, the response is very good, the overshoot is only one and undershoots are not presents.

3.2.2 Control with forced duty cycle

In this section, using the oil-pneumatic cylinder, the forced DC control is analyzed. The used variables are in the following table and are also explained in the fig.3.26:

DATA BLOCK

Static		
FEEDBACK	Int	0
SETPOINT	Int	0
NORMX_MIN	Int	6780
NORMX_MAX	Int	26930
SCALEX_MIN	Int	0
SCALEX_MAX	Int	100
NORMX_VALUE	Int	0
NORMX_OUTPUT	Real	0.0
ERROR	Int	0
ERROR_ABS	Int	0
INCREASE	Bool	false
RANGE_INCREASE	Bool	false
RANGE_DECREASE	Bool	false
DECREASE	Bool	false
DC_POS_0_100	Int	0
DC_NEG_0_100	Int	0
EMERGENCY/STOP	Bool	false
START	Bool	false
RANGE_MIN	Int	0
RANGE_MAX	Int	0

PLC VARIABLES

IW66	Word	%IW66
PWM_POS	Int	%QW1000
PWM_NEG	Int	%QW1002
OUTSTROKE	Bool	%Q0.1
INSTROKE	Bool	%Q0.2
MERKER_INCREASE	Bool	%M0.1
MERKER_DECREASE	Bool	%M0.2
MERKER EMERGENCY	Bool	%M0.3

Figure 3.22: Data block and PLC variables - Forced DC control - 3/2 valves

The main configurations are the same. So, the difference between *Setpoint* and *Feedback* is done.

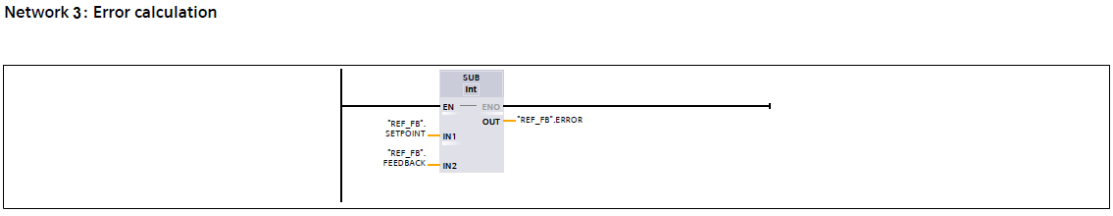


Figure 3.23: Network 3: Error calculation

Network 4: OUTSTROKE

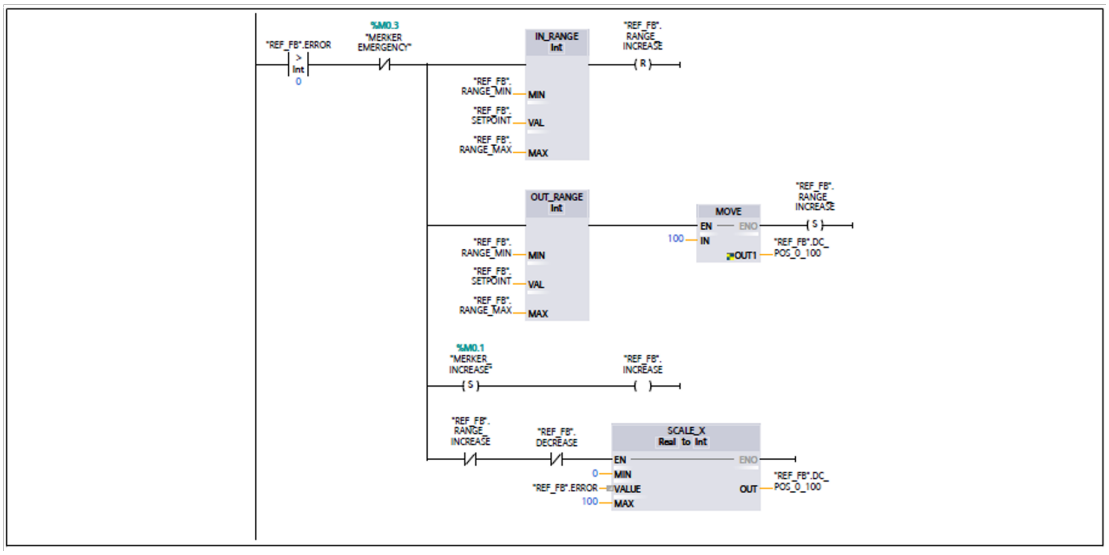


Figure 3.24: Network 4: Outstroke

The used transducer is the IW66 port of the PLC, as for the base control. The designed algorithm is as before.
But, now, two square waves are created, using pulse generators. In this way, it is possible to reach 100mm and on the other hand 0mm.

Network 5: INSTROKE

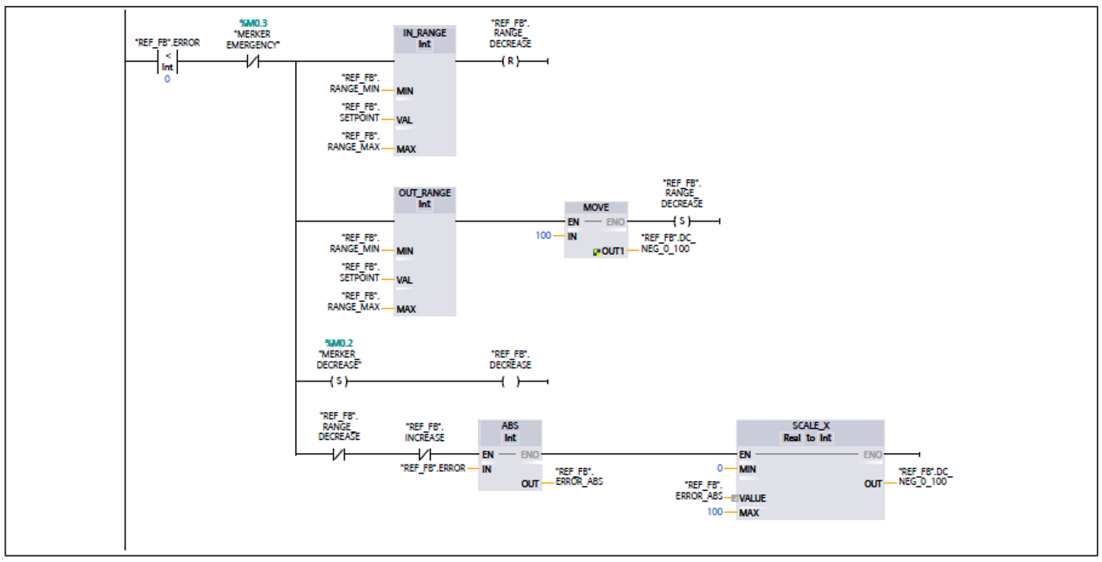


Figure 3.25: Network 5: Instroke

NAME	USE
FEEDBACK	Int variable used to manage the stroke position
ERROR_ABS	Int variable used to carry out the converted value of the error coming from the ABS block
SETPOINT	Int variable used as reference
NORMX_MIN	Predetermined Int variable used as minimum element on the normx block
NORMX_MAX	Predetermined Int variable used as maximum value on the normx block
SCALEX_MIN	Predetermined Int variable used as minimum element on the scalex block
SCALEX_MAX	Predetermined Int variable used as maximum value on the scalex block
NORMX_VALUE	Int variable used as input value of the normx block
NORMX_OUTPUT	Real variable used to carry out the output value of the normx block
ERROR	Int variable calculated as difference between SETPOINT and FEEDBACK
INCREASE	Bool variable equal to TRUE when the system is in increasing phase
DECREASE	Bool variable equal to TRUE when the system is in decreasing phase
RANGE_INCREASE	Bool variable used to manage the use of the forced duty cycle in increasing phase. It's setted when the DC is forced.
RANGE_DECREASE	Bool variable used to manage the use of the forced duty cycle in decreasing phase. It's setted when the DC is forced.
RANGE_MIN	Int variable used as lower threshold of the INRANGE block
RANGE_MAX	Int variable used as upper threshold of the INRANGE block
DC_POS_0_100	Int variable used to carry out the integer value of the Duty cycle in increasing phase
DC_NEG_0_100	Int variable used to carry out the integer value of the Duty cycle in decreasing phase
EMERGENCY/STOP	Bool variable used to manage the stopping phase
START	Bool variable used to start the system
PWM_POS	Int variable used to manage the PWM in increasing phase
PWM_NEG	Int variable used to manage the PWM in increasing phase
OUTSTROKE	Bool variable used to manage the output port Q0.1 of the PLC
INSTROKE	Bool variable used to manage the output port Q0.2 of the PLC
MERKER_INCREASE	Bool variable equal to TRUE when the system is in increasing phase
MERKER_DECREASE	Bool variable equal to TRUE when the system is in decreasing phase
MERKER_EMERGENCY	Bool variable equal to TRUE when the system is in emergency case
IW66	Int variable used to read the value coming from the input port IW66

Figure 3.26: Data block and PLC variables - Forced DC control - 3/2 valves - USE

After the experiments the obtained results are:

Overshoot %	Rise Time [s]	Settling Time [s]
10	0,206	0,655

Table 3.5: System response - Forced DC - 3/2 Valves - oil-pneumatic cylinder

So, the dynamic response is:

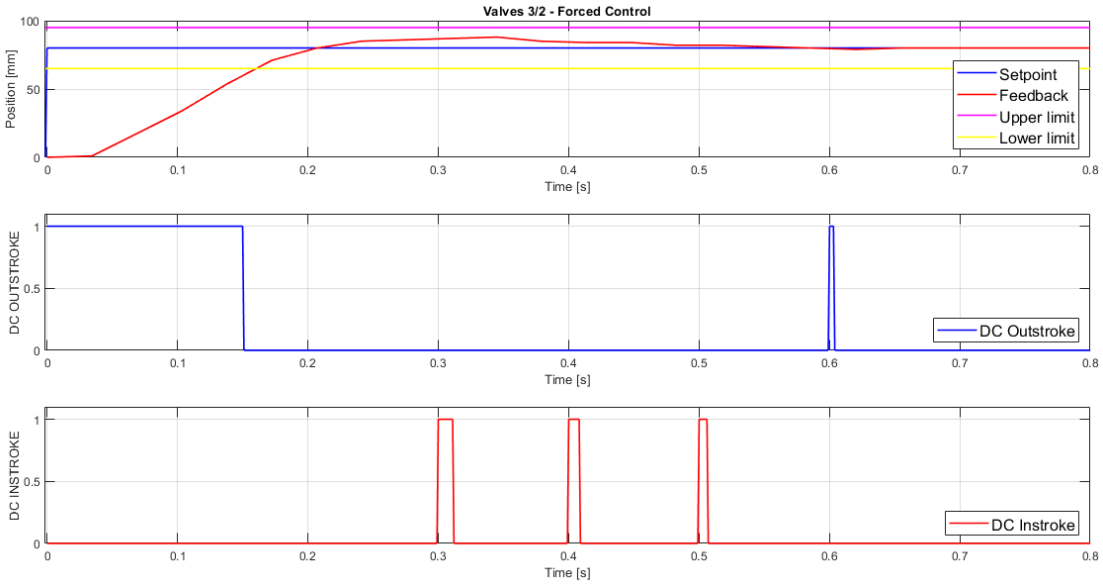


Figure 3.27: Dynamic response - Forced DC - 3/2 Valves - oil-pneumatic cylinder

3.3 Comparison between 2/2 NC valves and 3/2 NC valves

The comparisons using 2/2 Valves and 3/2 Valves with both controls in this section are analyzed.
The comparison is made with a sinusoidal input as error.
By using signals generator and the input port IW64, the *Setpoint* is obtained. As shown in figure 3.28 the used segment is:

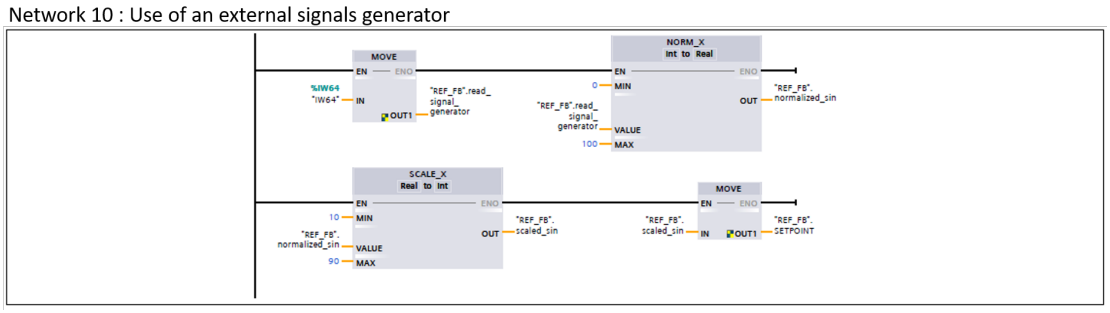


Figure 3.28: Tracking error - 3/2 Valves - Network

The dynamic responses of the tracking error using 3/2 Valves for the base control, using oil-pneumatic cylinder are:

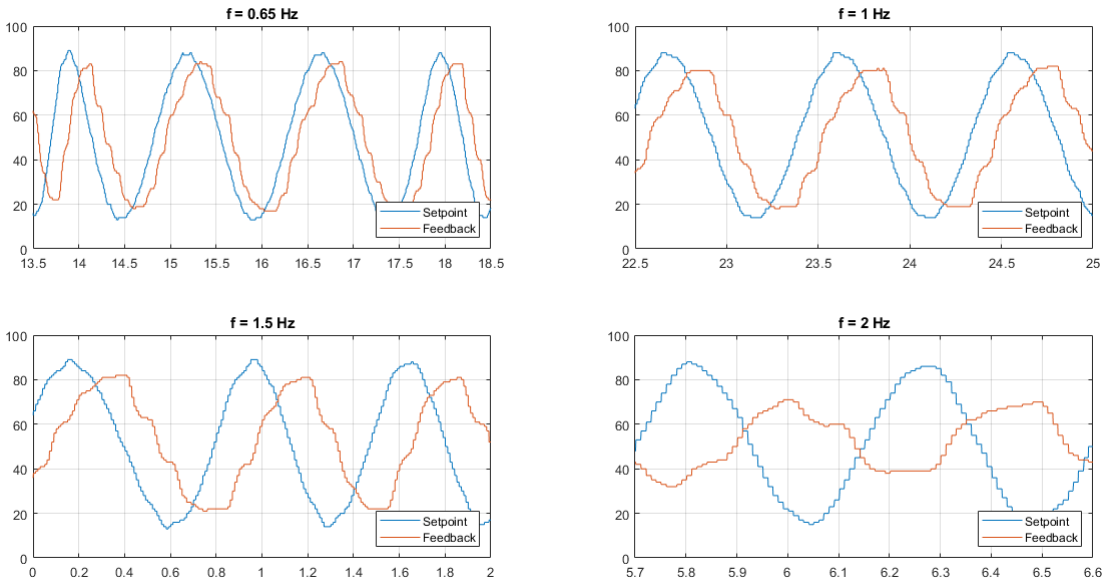


Figure 3.29: Tracking error - 3/2 Valves - Base Control - oil-pneumatic cylinder

By using different frequencies the tracking error is evaluated. So, the error is only acceptable with a frequency equal to or less than 0.65 Hz.
All the other tests show a decrease in the position reached and a considerable delay. The different behaviours are shown below:

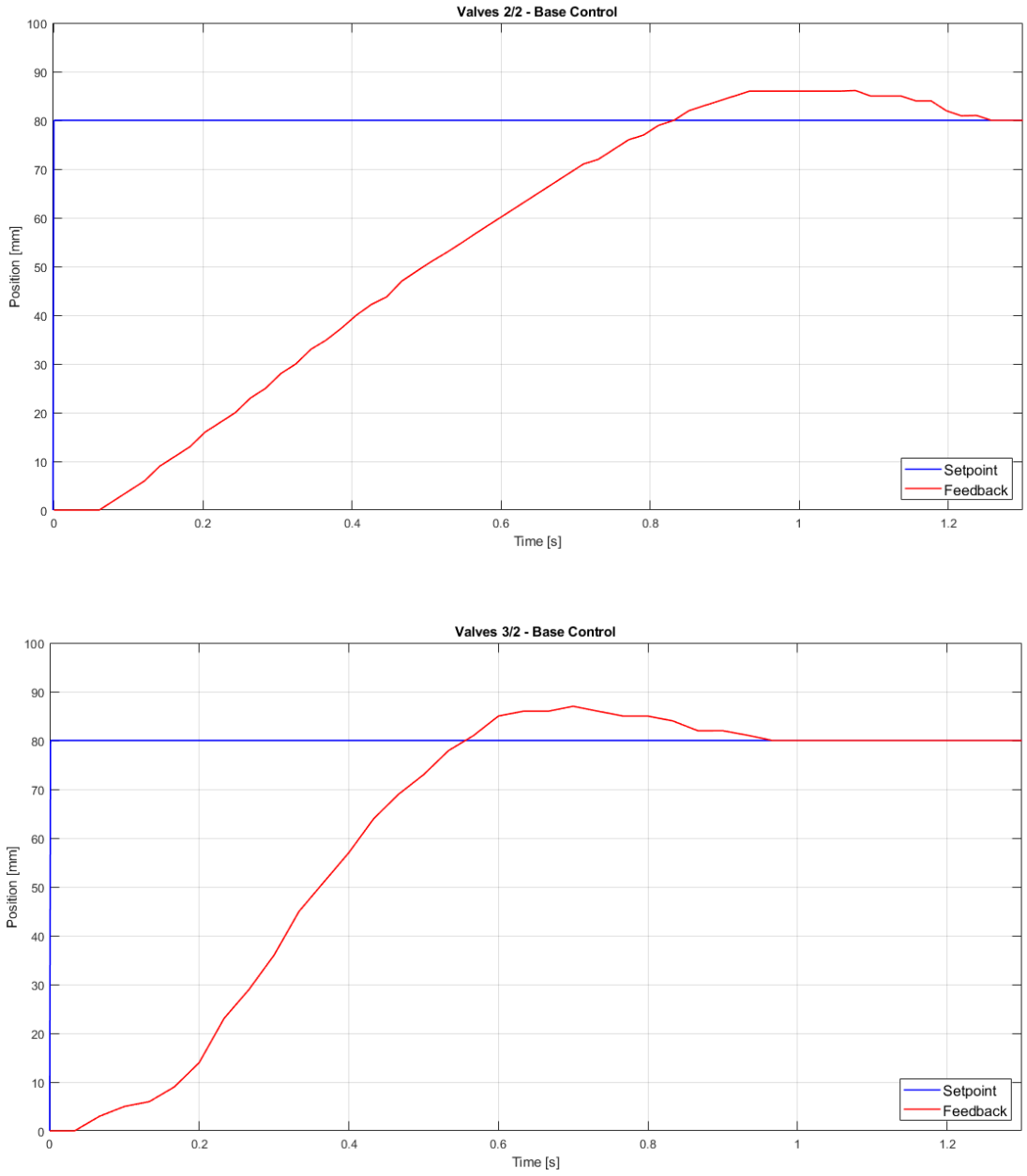


Figure 3.30: Base control comparison - oil-pneumatic cylinder

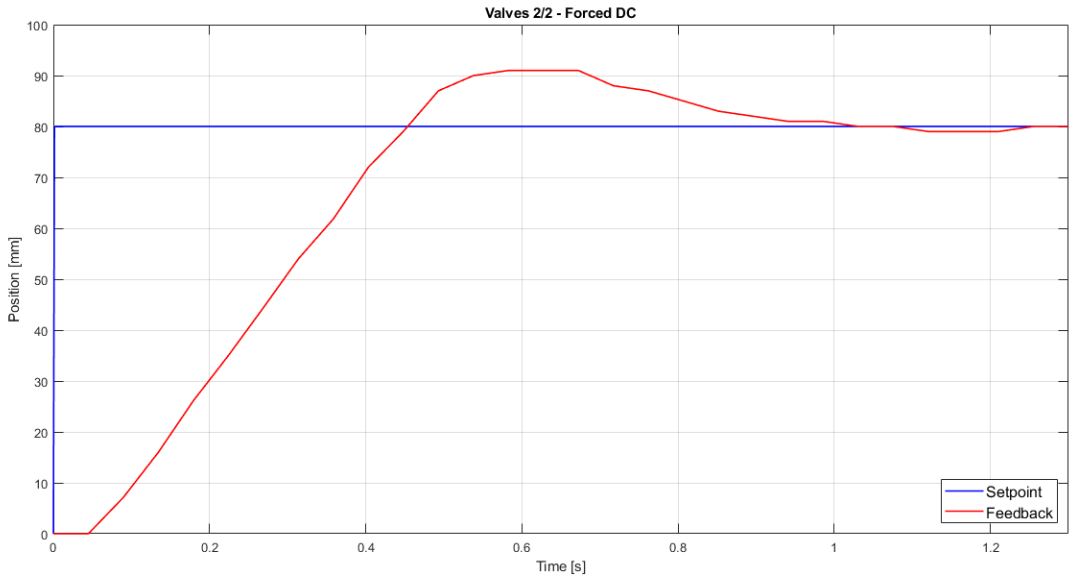
So, it is possible to note that the obtained shapes are similar. Using 3/2 valves

the resulting base control is more or less better. Counterwise, the only thing that is slightly worsened is the overshoot.

Algorithm	Overshoot %	Rise Time [s]	Settling Time [s]
2/2 Base Control	7,5	0,83	1,26
3/2 Base Control	8,75	0,227	0,537

Table 3.6: Base control comparison - oil-pneumatic cylinder

As regards the using of forced DC control, the resulting comparison is made with these responses:



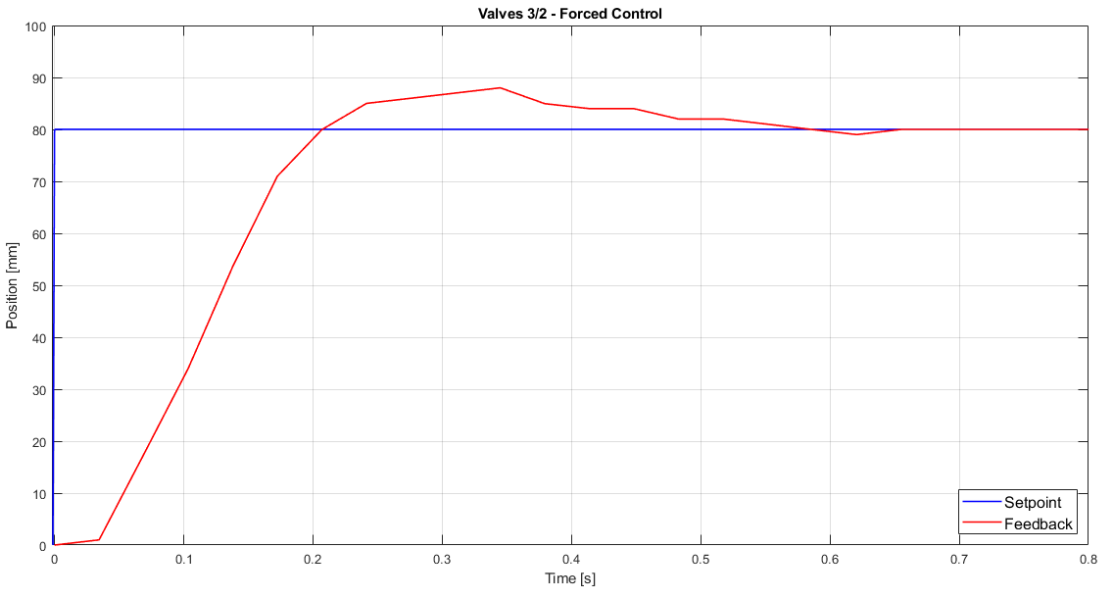


Figure 3.31: Forced DC comparison - oil-pneumatic cylinder

As the previous comparison for the forced DC control using the pneumatic cylinder, here the obtained performance are more efficient using the 3/2 valves. The numeric data are shown below:

Algorithm	Overshoot %	Rise Time [s]	Settling Time [s]
2/2 Forced DC	13,75	0,451	1,267
3/2 Forced DC	10	0,206	0,655

Table 3.7: Forced DC comparison - oil-pneumatic cylinder

So, by combining these comparisons, what it is possible to deduce is that by using the base control with the 3/2 valves you get improvements in the settling time and also, good rise times and overshoot.

Algorithm	Overshoot %	Rise Time [s]	Settling Time [s]
2/2 Base Control	7,5	0,83	1,26
3/2 Base Control	8,75	0,227	0,537
2/2 Forced DC	13,75	0,451	1,267
3/2 Forced DC	10	0,206	0,655

Table 3.8: Final comparison - oil-pneumatic cylinder

Moreover, if is highlighted the behavior between 2/2 base and forced DC controls and then between 3/2 base and forced DC controls, there is no deep diversity as before, using the pneumatic cylinder.
This is because this time the cylinder is damped.

Chapter 4

PID controller using pneumatic cylinder and oil-pneumatic cylinder

In this section, the position control is obtained using a PID controller. In particular, this control is chosen because its implementation improved the results of the previous control. On the TIA portal, it is present a block, called *PID Compact*, which fits with the control problem. If before the designed algorithm were based on the proportional control, so by using the difference between *Setpoint* and *Feedback*; now, the error is Proportional-Integrative-Derivative (PID). On the TIA portal, the *PID Compact* is a PID with a anti-windup regulator, and present a Proportional and Derivative weighting. The algorithm is based on the following equation:

$$y = K_p[(b \cdot w - x) + \frac{1}{T_I \cdot s} \cdot (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1}(c \cdot w - x)] \qquad (4.1)$$

The terms are listed in the table below (4.1)

y = Output of the PID algorithm
Kp = Proportional gain
w = Setpoint
x = Feedback
T_I = Integration time
T_D= Derivative time
s = Laplace operator
b = weighting of P component
c = weighting of D component
a = coefficient for the derivative delay

Table 4.1: Equation variables

Observing the table 4.1, it is possible to note the weighting of P and D terms. These terms are indicated with a b and c. Except for these terms, then the equation is normally a PID controller equation, b and c can be imposed equal to 1. The acceptable range which allows to have a correct behaviour by the datasheet is between 0 and 1. If the chosen value is 1, the equation becomes:

$$y = K_p[(w - x) + \frac{1}{T_I \cdot s} \cdot (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1}(w - x)] \tag{4.2}$$

In particular, the relationship between the variables used on TIA portal and the real variables used on the PID's equation are:

$$Kp = kp \tag{4.3}$$

$$Ti = \frac{kp}{ki} \tag{4.4}$$

$$Td = \frac{kp}{kd} \tag{4.5}$$

The *PID Compact* has many In/Out ports, in the following figure (4.1) the main block. Also, it is important to place this block inside the *Cyclic interrupt* (OB30), in order to allow a scan with a predetermined frequency.

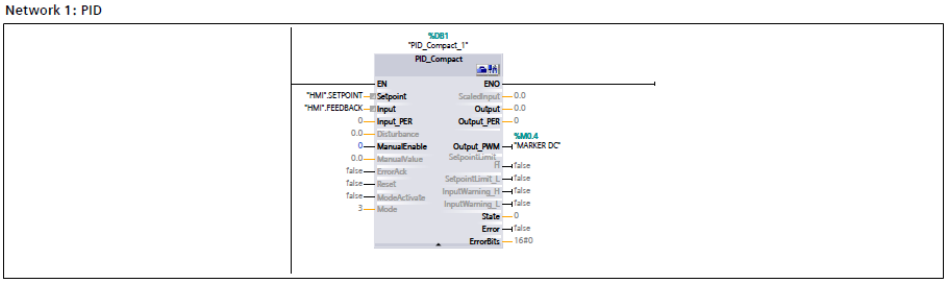


Figure 4.1: PID Compact

In order to use the designed algorithm the used ports are:

- Setpoint
- Input
- Output_PWM

The input port *Setpoint* is linked with the Block variable *"HMI".SETPOINT*, the second one is linked with the Block variable *"HMI".FEEDBACK*, and the last one is linked with a merker, called *merker_DC*.
Before using the *PID Compact* block in the algorithm on the TIA portal, some configurations are required.
In particular, subsequently, each setting is described.

- Controller type: is imposed the type of the controller (length, pressure, temperature, ...) and it's scale (mm, Pa, °C,...). Moreover, the set mode is set to Automatic Mode.

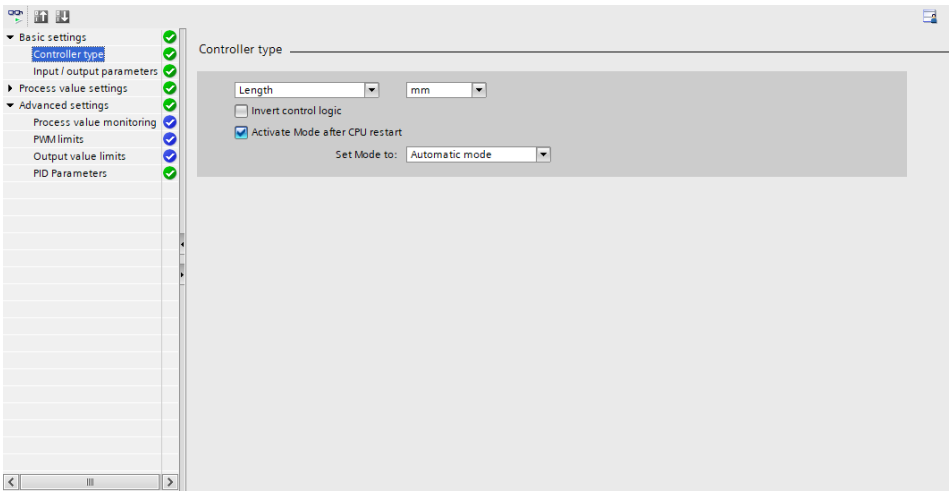


Figure 4.2: Controller type

- Input/output parameters: is set the type of input and the output port

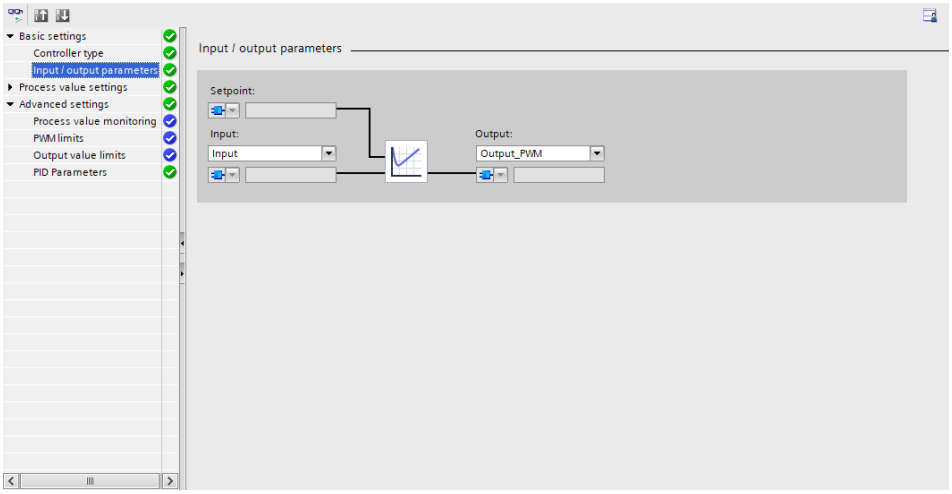


Figure 4.3: Input/output parameters

- Process value limits: are set the limits of the inputs (mm)

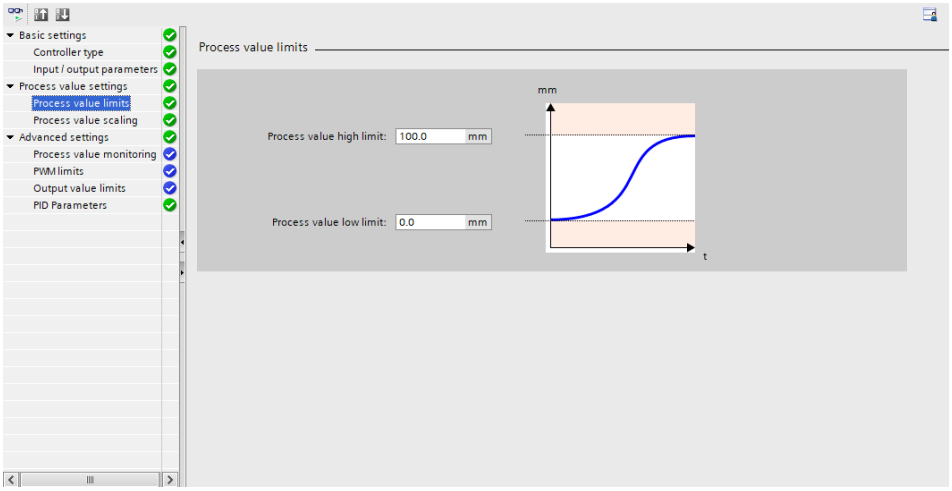


Figure 4.4: Process value limits

- Process value scaling: this setting allows to normalize and scale an analog signal, in this case the command is disabled, because it is exploited outside the PID.

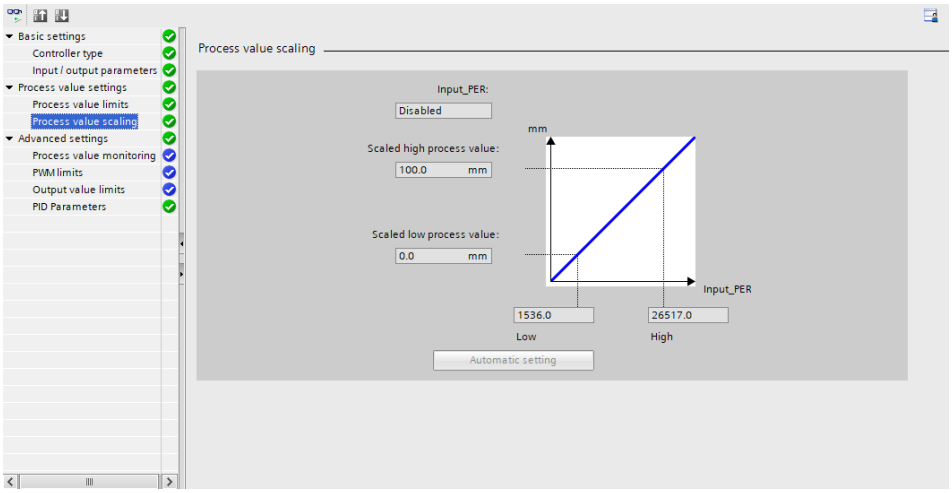


Figure 4.5: Process value scaling

- Process value monitoring: this setting allows to monitoring and rise up a flag in case of limit crossing, in this case the command is disabled.

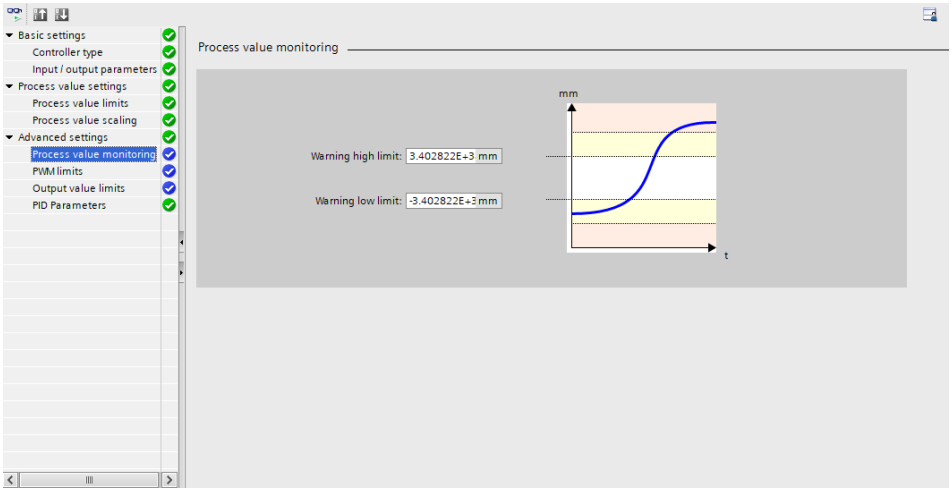


Figure 4.6: Process value monitoring

- PWM limits: used in order to set the minimum time where the PWM is ON or OFF

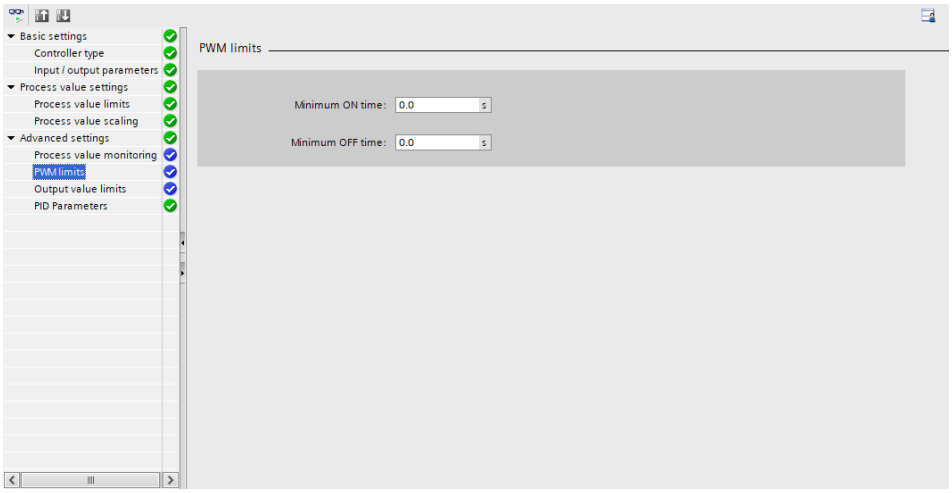


Figure 4.7: PWM limits

- Output value limits: this setting is automatically configured, as follows, if the output selected in "*Input/output parameters*" is OUTPUT PWM

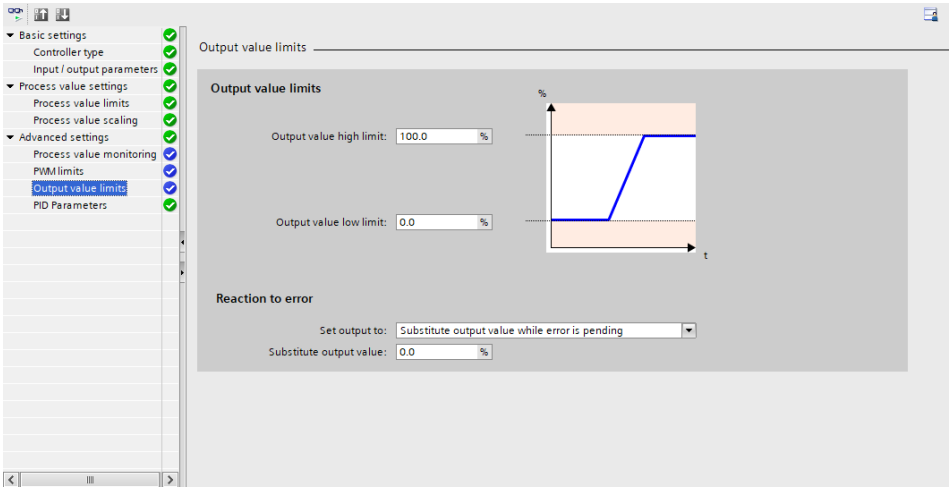


Figure 4.8: Output value limits

- PID Parameters: in this section it is possible to show the controller structure and the other parameters. In particular, after, the settings are chosen in different way, in order to made different experiments.

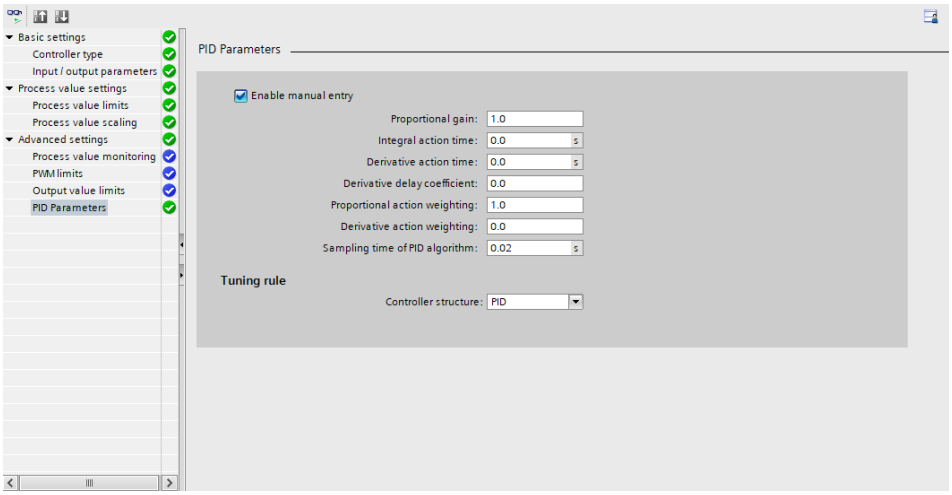


Figure 4.9: PID parameters

Based on the type of the cylinder the following configurations are used. The minimum *PID Compact* sampling time which is possible to choose is 1ms. Otherwise, by the datasheet the possible PID algorithm sampling time it should be at least ten times the OB30 sampling time . This ratio is chosen as the PWM output is used.

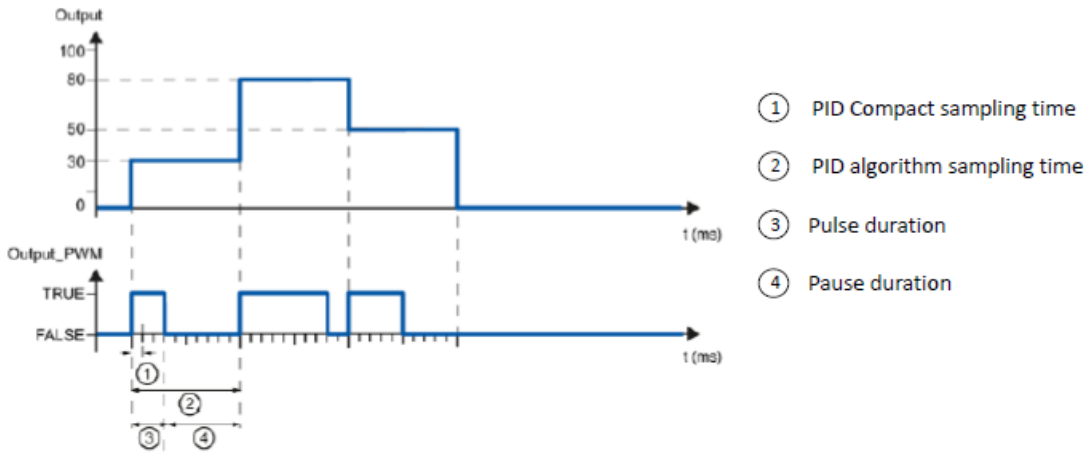


Figure 4.10: Datasheet PID Compact

Moreover, this allows to have a signal output with a high precision.
For what concerned the pneumatic cylinder, various launches are performed in order to obtain the correct ratio between the cycle of the OB30 and the sampling time of the PID algorithm, as shown in the fig.4.11:

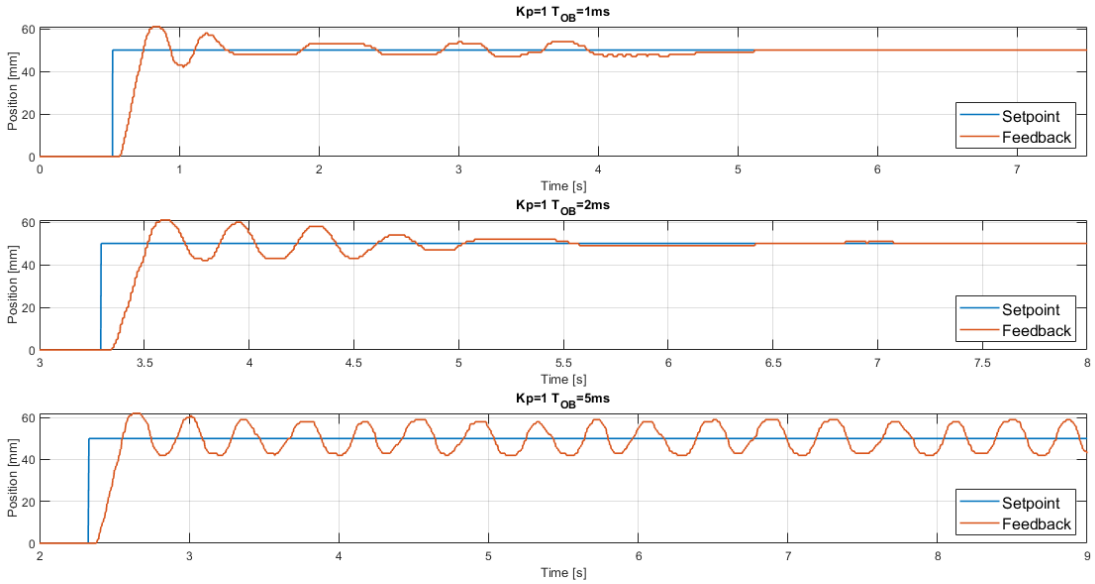


Figure 4.11: Variables PID Compact sampling time - pneumatic cylinder

In this way, the sampling time of the PID Compact is chosen equal to 2 ms. By the fig.4.11, it is possible to note that the 5ms has an unstable behaviour, while using 1ms the shape is worse.
Consequently, choosing 2 ms the ratio for the sampling time of the PID algorithm must be 10 times greater, so equal to 20 ms. In order to demonstrate the choice, several experiments, are performed, varying the sampling time of the OB30, as shown in the fig.4.12

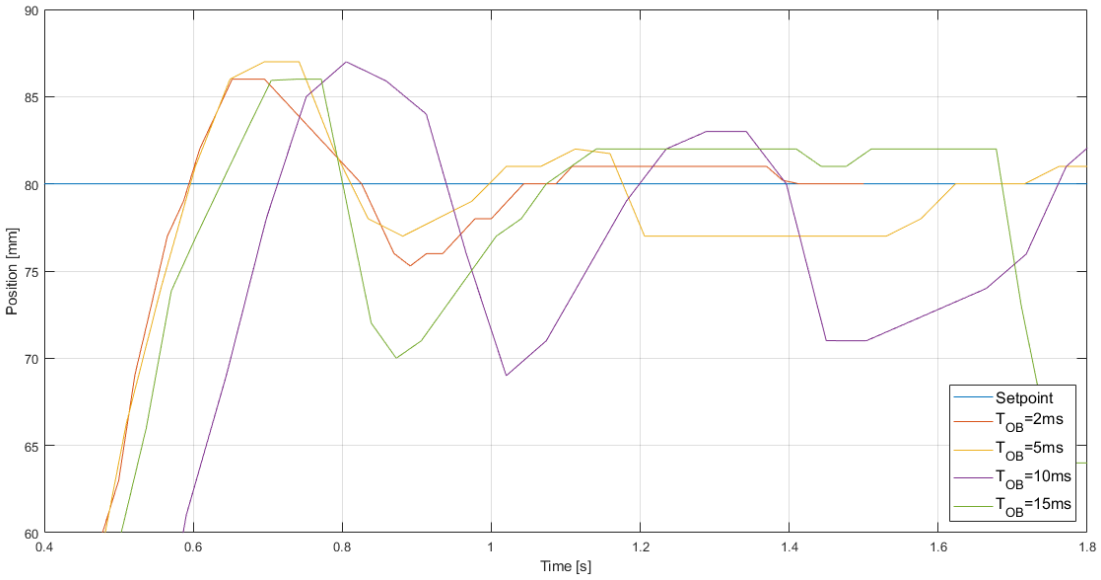


Figure 4.12: Variables OB30 sampling time - pneumatic cylinder

PID algorithm sampling time	20ms	20ms	20ms	20ms
PID compact sampling time	2ms	5ms	10ms	15ms
Accuracy ratio	10	4	2	1,33

Table 4.2: Table variables OB30 sampling time - pneumatic cylinder

So, choosing these sampling times, 2 ms for OB30 and 20 ms for the PID algorithm, an accuracy ratio equal to 10 is obtained.

The same procedure is made for the oil-pneumatic cylinder. So, in order to choose the sampling time of the *PID Compact* many experiments are done.

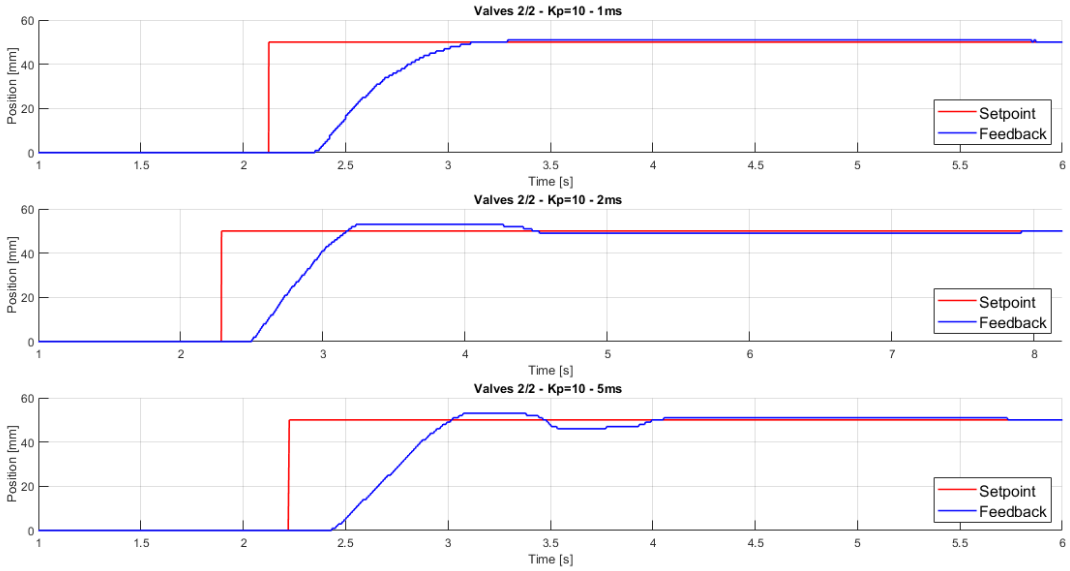


Figure 4.13: Variables PID Compact sampling time - oil pneumatic cylinder

After that, is possible to note that the curve with the best performance is associated with the OB30 smapling time equal to 1 ms. So, in order to respect the recommended ratio, the PID algorithm sampling time is choose equal to 10 ms. Now, after these configurations the segments used into the cyclic interrupts are:

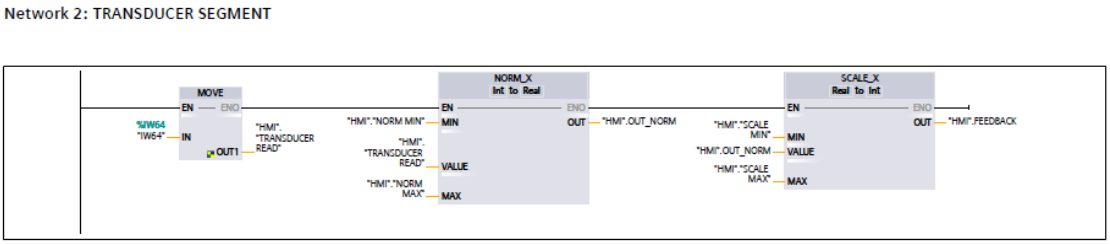


Figure 4.14: Network 2: Transducer segment PID

So, in fig.4.14 the transducer segment is shows, where scale and norm blocks are used, as before.
As regards the emergency and the error calculation, the first one is performed with a N/O contact that activate a "Merker emergency" in order to stop the system. While, to restart the system, the N/O contact "Restart" is used.
For what concerned the error calculation, the segment is equal as the previous projects.

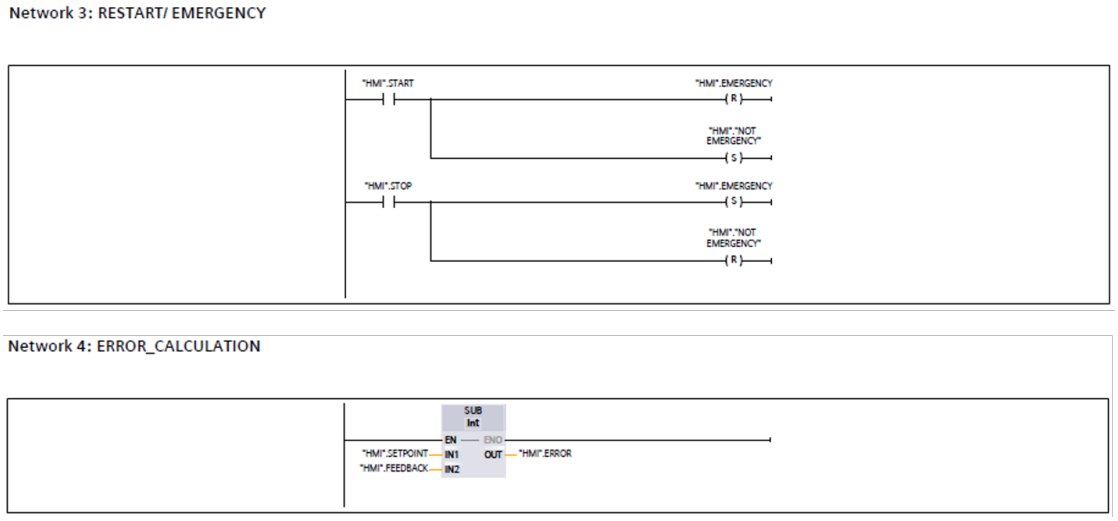


Figure 4.15: Network 3 and 4: Restart/Emergency and calculation PID

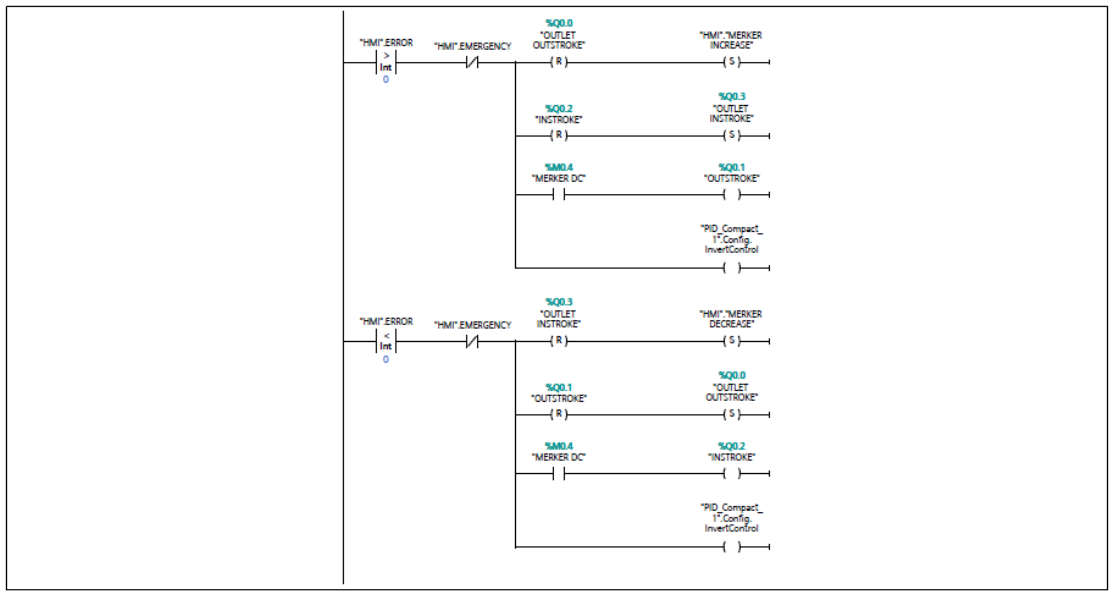


Figure 4.16: Network 5: Outstroke/Instroke PID

The "Error" variable is used as known the system's phase. If the error is greater than 0, the "Merker DC" variable is used to performed the outstroke, otherwise the instroke is asserted.

In this particular case, using the 2/2 valves, when the outstroke happens, the output "Q0.3" is activate in order to connect the front chamber with the environment. Counterwise, the output "Q0.0" is asserted if the instroke phase in order to accomplish the same function.
As shown in fig.4.16, in the last rung for each phase, another contact is used. The contact is the **Invert Control** used to performed stroke's inversion. Without these rungs, the piston is block, so the Setpoint is not achieved.

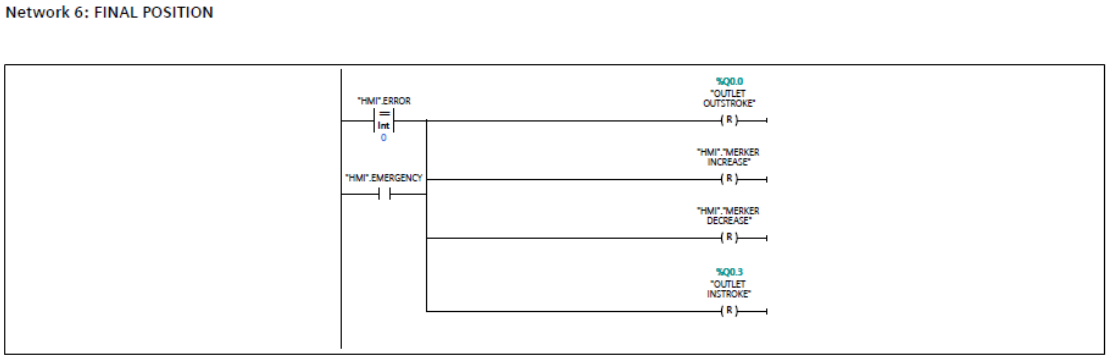


Figure 4.17: Network 6: Final position PID

In the last segment (fig.4.17) the *Setpoint* achieved the *Feedback* so all ports are deactivated. The same function is used in the Emergency case, through the "Emergency" contact.

4.1 PID control using valve 2/2 NC

By different PID parameters, various experiments are shown below, three types of configuration are used:

- Pure Proportional
- Pure Integrative
- Pure Derivative

4.1.1 Pure Proportional

In order to use a single pure control, settings are changed. In particular, to made a pure proportional control the parameters are configured as shown in following fig.4.18

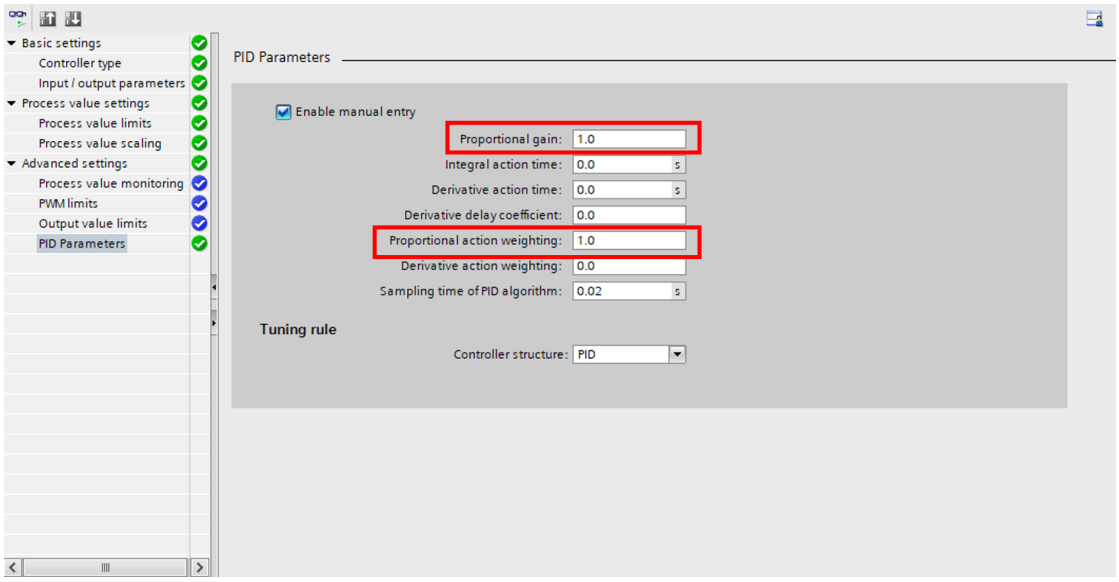


Figure 4.18: Pure proportional control - pneumatic cylinder

This allows to have in the equation all the terms equal to zero, and the gain of the pure porportional control set in different way. As shown in the fig.4.19 , the gain is configured with different values for the pneumatic cylinder:

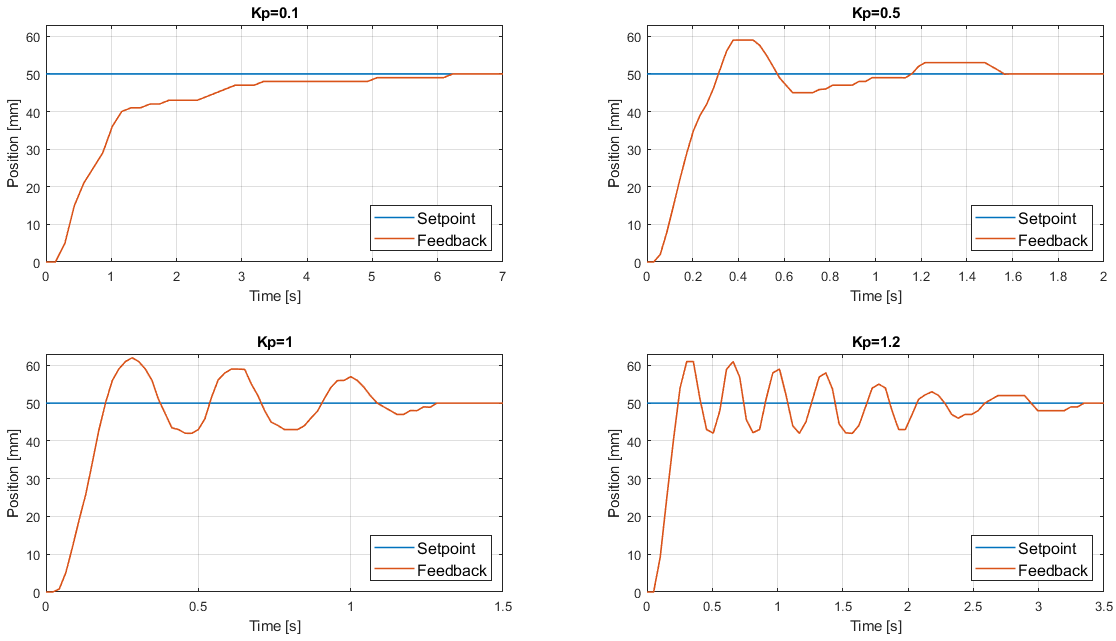


Figure 4.19: Pure proportional control - Gain - pneumatic cylinder

It is possible to observe, that with Kp=0.1 the response has a rise time too big, while with Kp=0.5 are presented two overshoot and one undershoot with rise time and settling time appreciable. Moreover, if the gain is equal to 1, the dynamic response is better, but there are several overshoots and undershoots. Instead, with Kp=1.2, the settling time is not comparable with respect the other results. By these launches the chosen gain is: Kp=1.

Kp	Rise time [s]	Settling time [s]	Overshoot %
0,1	6,1	6,1	0
0,5	0,3	1,57	18
1	0,195	1,28	24
1,2	0,238	3,3	22

Table 4.3: Gain Kp - pneumatic cylinder

Moreover, though the same procedure, the results obtained for oil-pneumatic cylinder as shown in fig.4.20

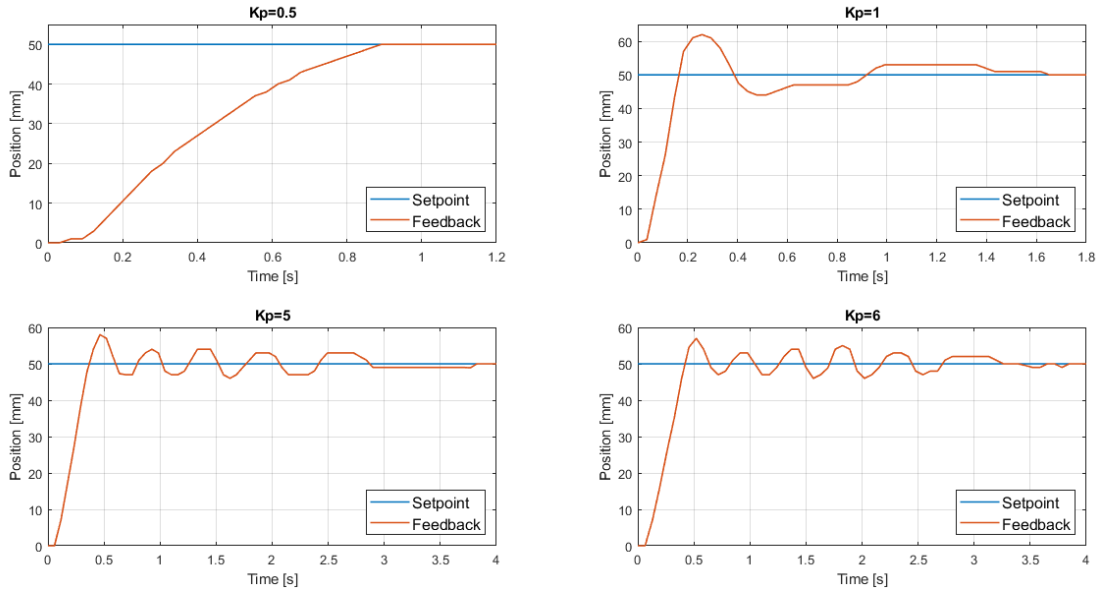


Figure 4.20: Pure proportional control - oil pneumatic cylinder

It's possible to note, which for higher value of Kp, the behaviour shows several

overshoot and undershoot. The better response is obtained with Kp equal to 0.5, there is no overshoot and time characteristics are appreciable.

Kp	Rise time [s]	Settling time [s]	Overshoot %
0,5	0,872	0,872	0
1	0,162	1,63	2,4
5	0,349	3,826	16
6	0,395	3,847	14

Table 4.4: Gain Kp - oil-pneumatic cylinder

So, now it is possible to proceed with the other controls.

4.1.2 Pure Integrative

In this section, the behaviour analyzed is pure integrative. In order to obtain this configuration needed to set only integral action time at the desired value. Also, it is important that the proportional gain coefficient remained set equal to 1.

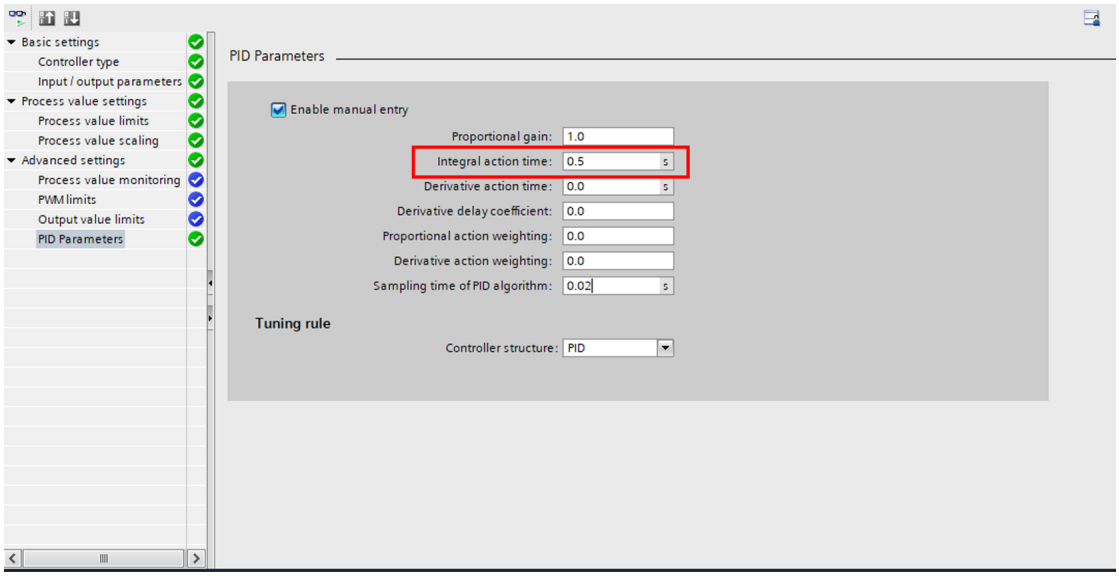


Figure 4.21: Pure integrative control - pneumatic cylinder

By using different values, the obtained results are shown in the following figure.

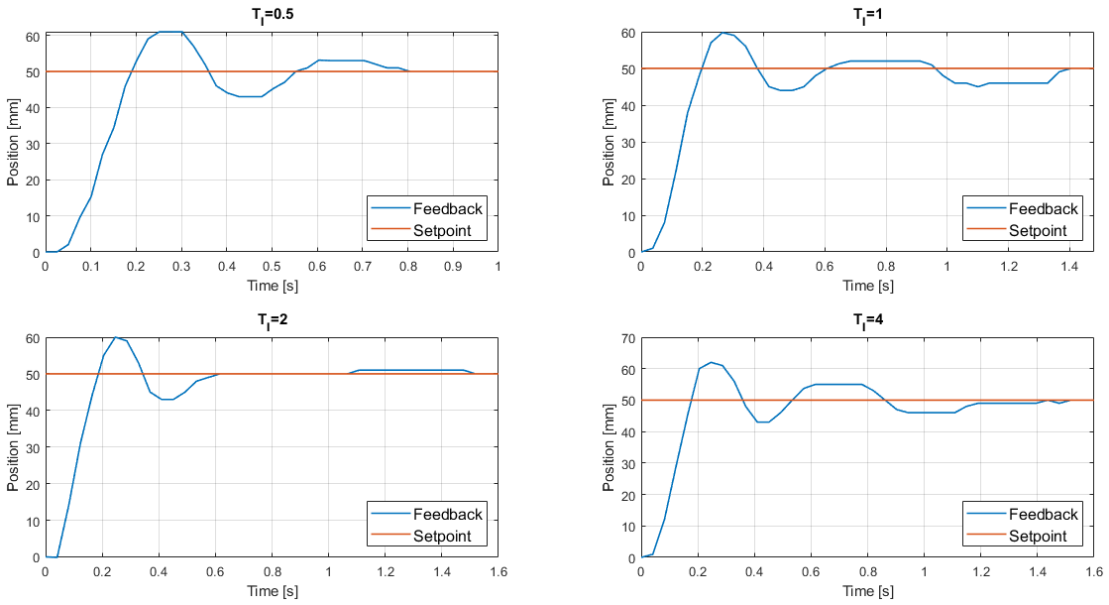


Figure 4.22: Pure integrative control - Ti - pneumatic cylinder

Therefore, it is possible to note that as the coefficient Ti increases, all the dynamic characteristics worsen.

Ti [s]	Rise time [s]	Settling time [s]	Overshoot %
0,5	0,183	0,804	2,2
1	0,1966	1,44	2
2	0,1823	1,52	2
4	0,166	1,487	2,4

Table 4.5: Ti - pneumatic cylinder

While, in the oil-pneumatic cylinder the obtained results are:

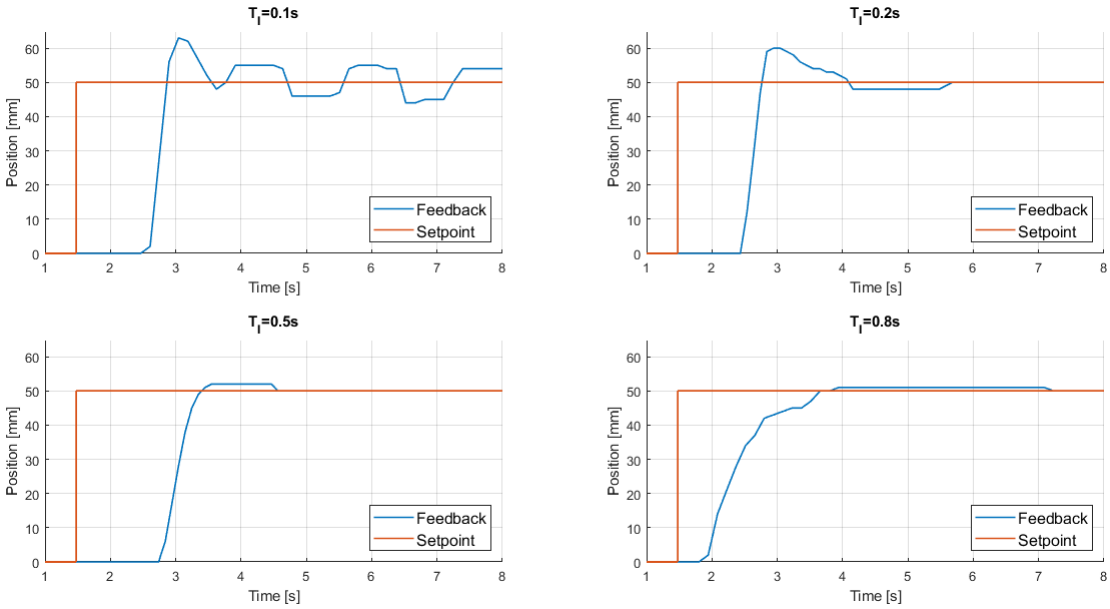


Figure 4.23: Pure integrative control - Ti - oil-pneumatic cylinder

Ti	Rise Time [s]	Settling Time [s]	Overshoot %
0,1	2,2	7,38	26
0,2	2,11	5	20
0,5	2,73	3,91	4
0,8	2,17	5,73	2

Table 4.6: Ti - oil-pneumatic cylinder

So, the chosen T_I is equal to 0.5s, because its response has the best compromise between rise time, settling time and overshoot with respect to the other experiments.

4.1.3 Pure Derivative

In this section, the PID controller is set as Proportional-Derivative Controller. All the parameters are imposed as shown in the following fig4.24

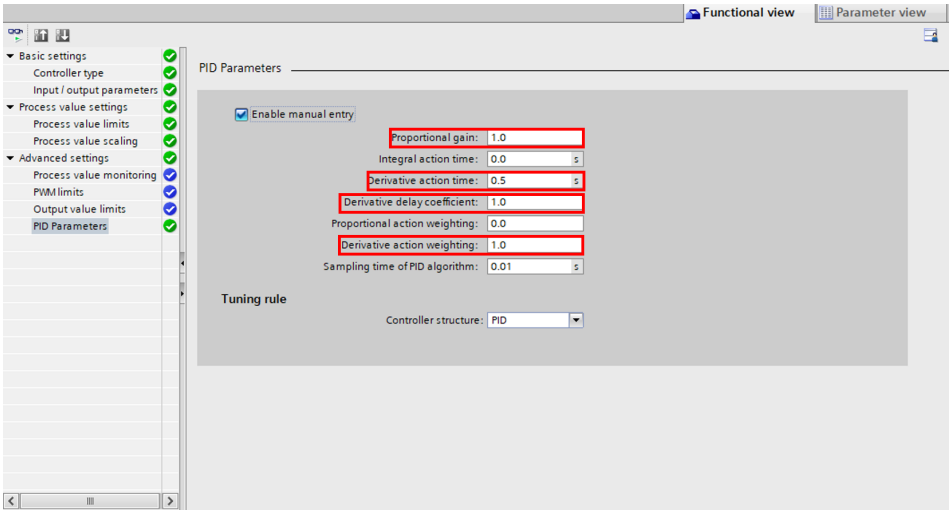


Figure 4.24: Pure derivative control - pneumatic cylinder

In particular, the Kp and the c are set equal to 1, instead the T_D is changed in order to obtain the correct value. To get its value different experiments are done. In order to made the Derivative Control the filter is switch on, its activation depends on the a parameter, which is set to one.

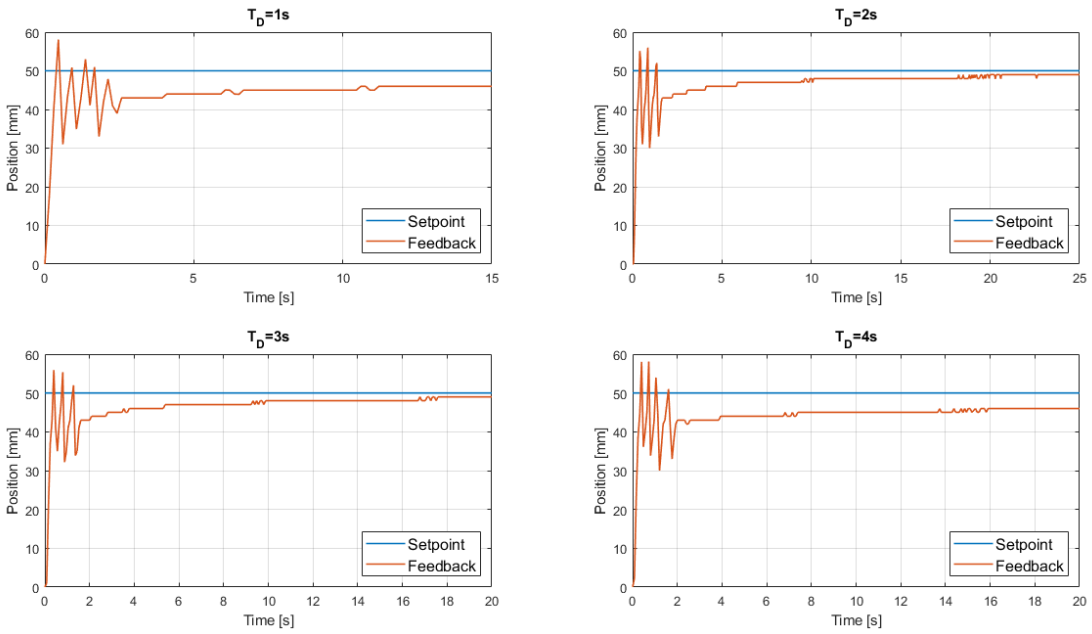


Figure 4.25: Proportional derivative control - pneumatic cylinder

From the figures it is possible to see that the dynamic response is correct only after a certain period of time. The chosen value is $T_D = 2$ s, as the response has less overshoot than the others. For what concern the oil-pneumatic cylinder the best behaviour is obtained set $T_D = 0.5$ s, in order to avoid errors, in terms of undershoot response.

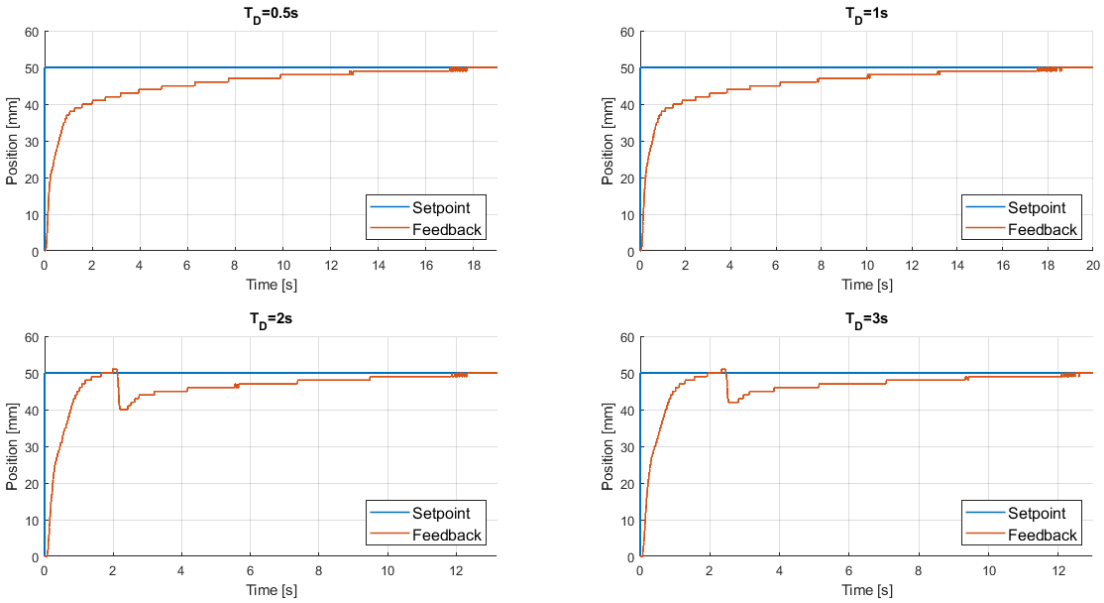


Figure 4.26: Proportional derivative control - oil-pneumatic cylinder

Based on results of both cylinders, overshoots and undershoot responses presents in the pneumatic cylinder are linked to the absence of the damper, which appears in the oil-pneumatic cylinder.

4.2 PID control using valve 3/2 NC

Now, using the 3/2 valves same experiments are repeated for both cylinders: pneumatic cylinder and oil-pneumatic cylinder. Using the OB30 sampling time equal to 2 ms for the first cylinder and 1 ms for the second one. Now, the used 3/2 valves are different with respect the valves used before, in particular are the 3/2 valves OX 821 103C2KK, and also, in order to connect them, is used a driver. The connections in fig.4.27 is made:

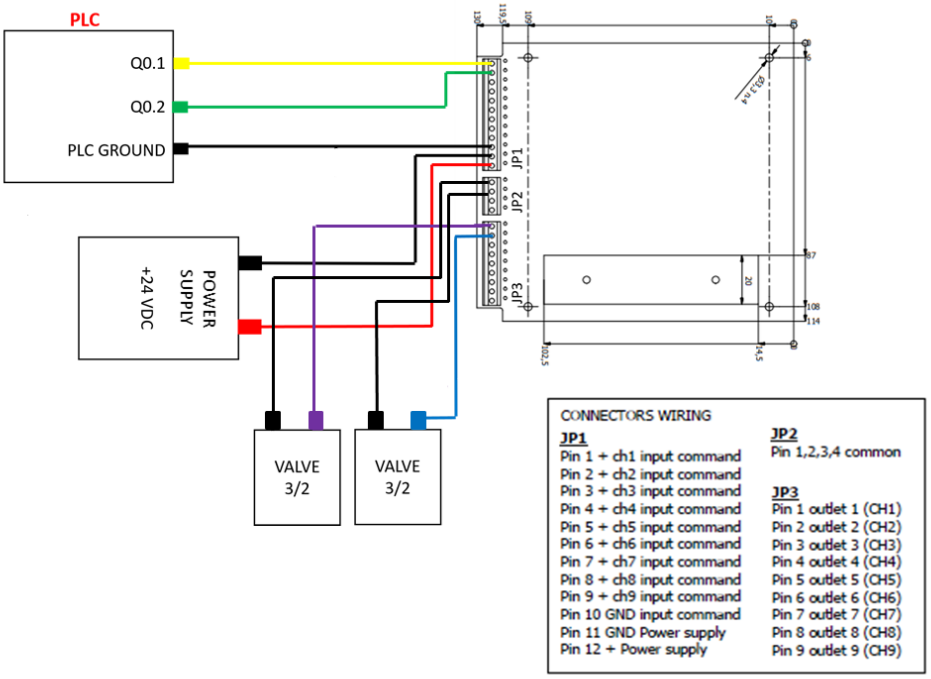


Figure 4.27: Driver and Valves connections

By the datasheet it is made the connection as follows: the PLC's outputs Q0.1, Q0.2 are connected in the pin1 and pin2 of the JP1 part of the driver, while the ground on pin10. Power supply is connected at the pin11 and pin12 of JP1, instead the valves have the common ground with the JP2, and the outlet connect in the pin1 and pin2 of the JP3, respectively.
The made experiments are:

- Pure Proportional
- Pure Integrative
- Pure Derivative

4.2.1 Pure Proportional

The settings on TIA portal are:

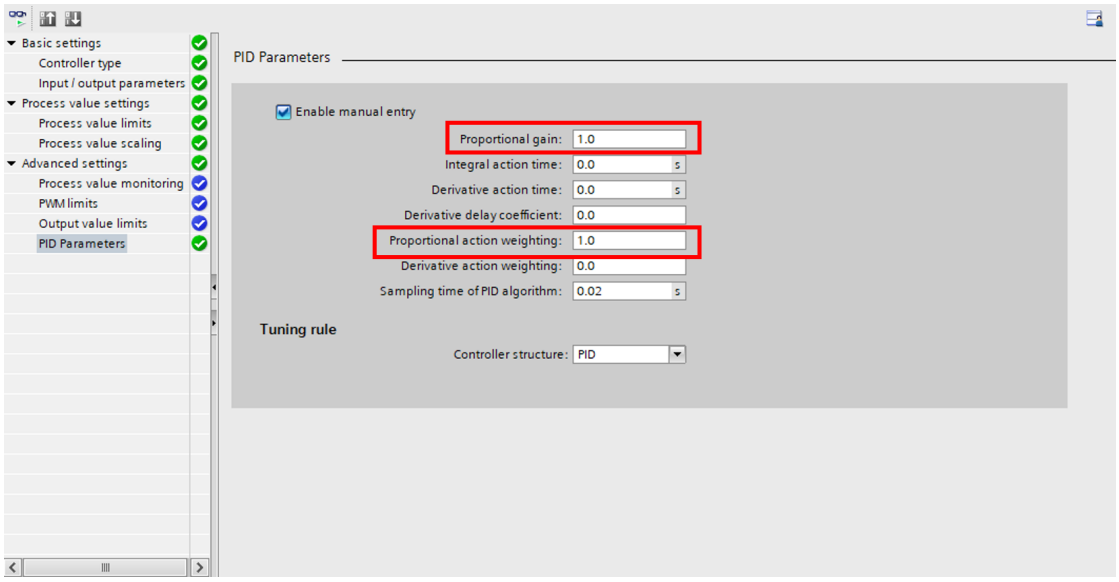


Figure 4.28: Pure proportional control - Pneumatic cylinder - 3/2 Valves

Thanks these configurations the pure proportional control is made with different parameters. For the pneumatic cylinder the obtained responses are:

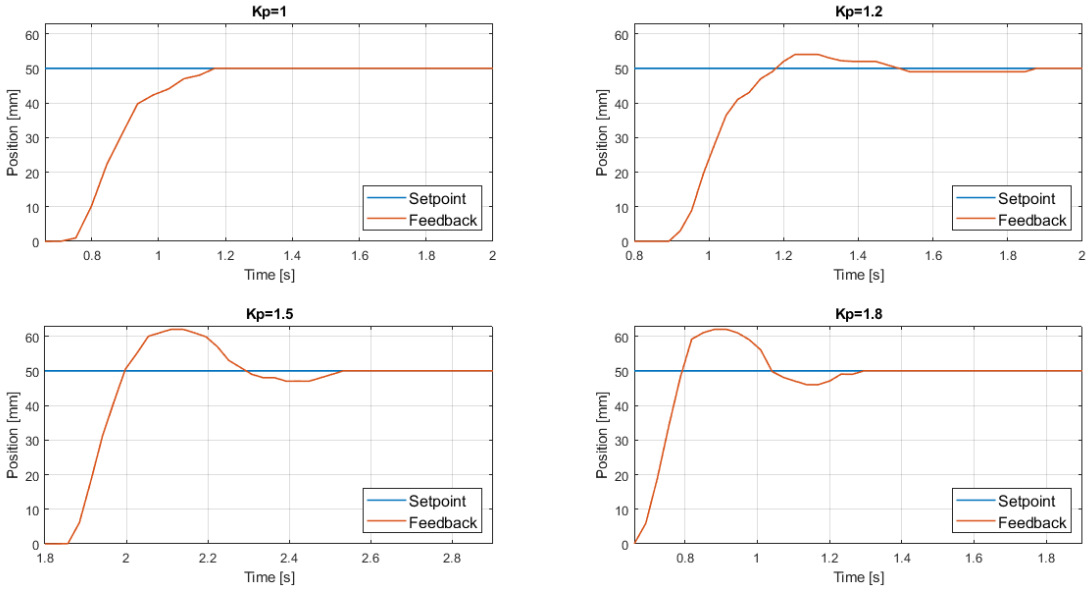


Figure 4.29: Pure proportional control - pneumatic cylinder - 3/2 Valves

So, it is possible to note that with Kp=1 the response is better than using Kp=1.2.

The best overshoot is obtained with Kp=1, which is absent. Also, the response presented has a rise time and settling time equal to 1.169 s.

Kp	Rise Time [s]	Settling Time [s]	Overshoot %
1	1,168	1,168	0
1,2	1,169	1,877	8
1,5	1,997	2,533	24
1,8	0,7872	1,96	24

Table 4.7: Gain Kp - pneumatic cylinder - 3/2 Valves

As regards the oil-pneumatic cylinder the obtained results are shown in the figure below:

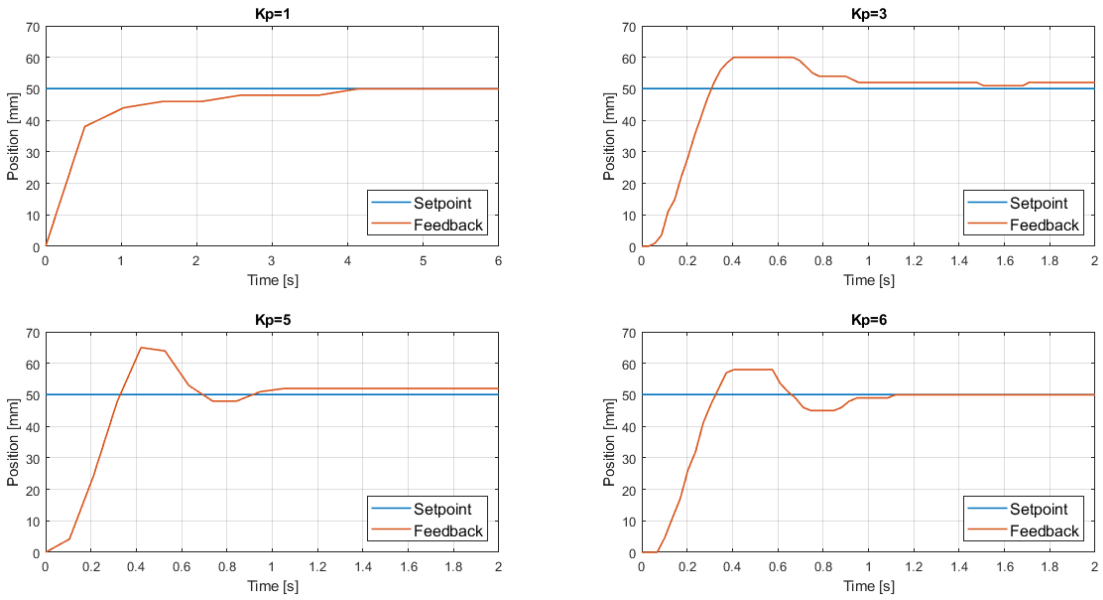


Figure 4.30: Pure proportional control - oil-pneumatic cylinder - 3/2 Valves

So, using Kp=6, is obtained a good compromise between rise time and overshoot.

Kp	Rise time [s]	Settling time [s]	Overshoot %
1	4,07	4,07	0
3	0,3	-	20
5	0,33	-	30
6	0,33	1,12	16

Table 4.8: Gain Kp - oil-pneumatic cylinder - 3/2 Valves

4.2.2 Pure Integrative

Now, the behaviour analyzed is pure integrative. So, settings only integral action time at the desired value the obtained configuration is:

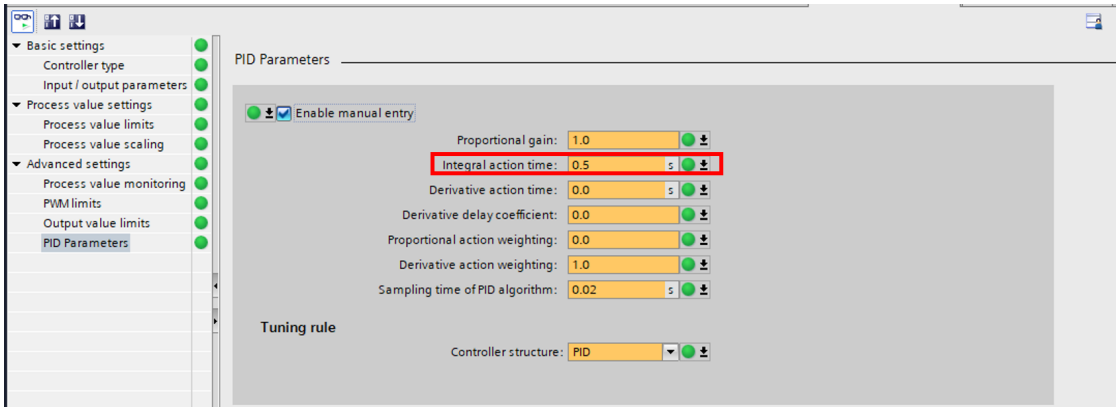


Figure 4.31: Pure integrative configuration - 3/2 Valves

By using different values, imposing the proportional gain coefficient equal to 1, the obtained results are shown in the figure below:

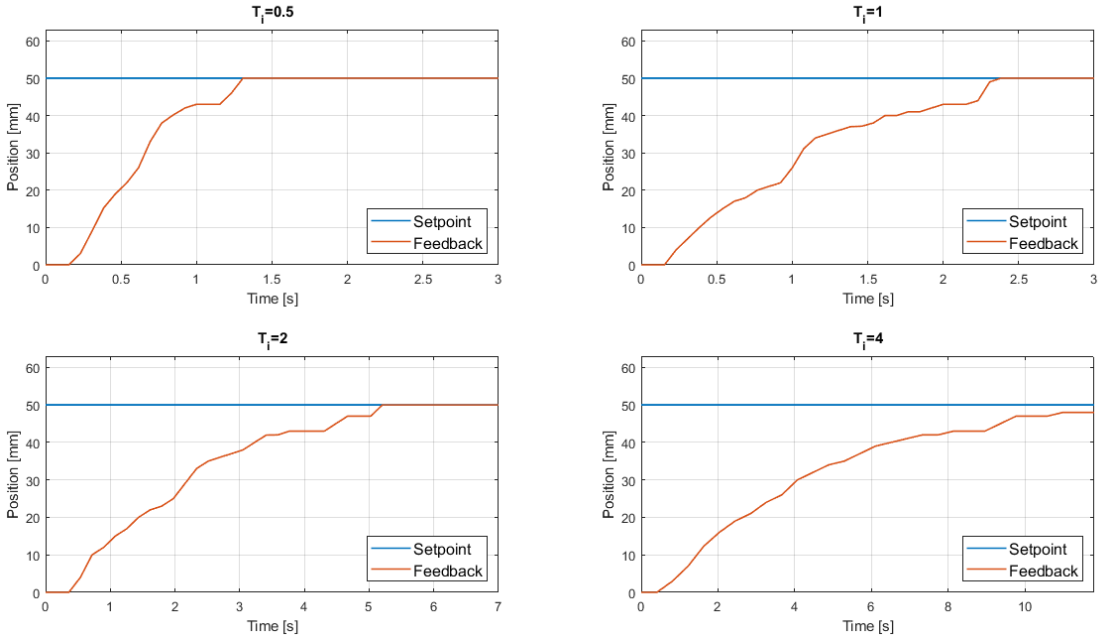


Figure 4.32: Pure integrative control - pneumatic cylinder - 3/2 Valves

Also, for what concerned the oil-pneumatic cylinder, using the pure integrative control the obtained results are:

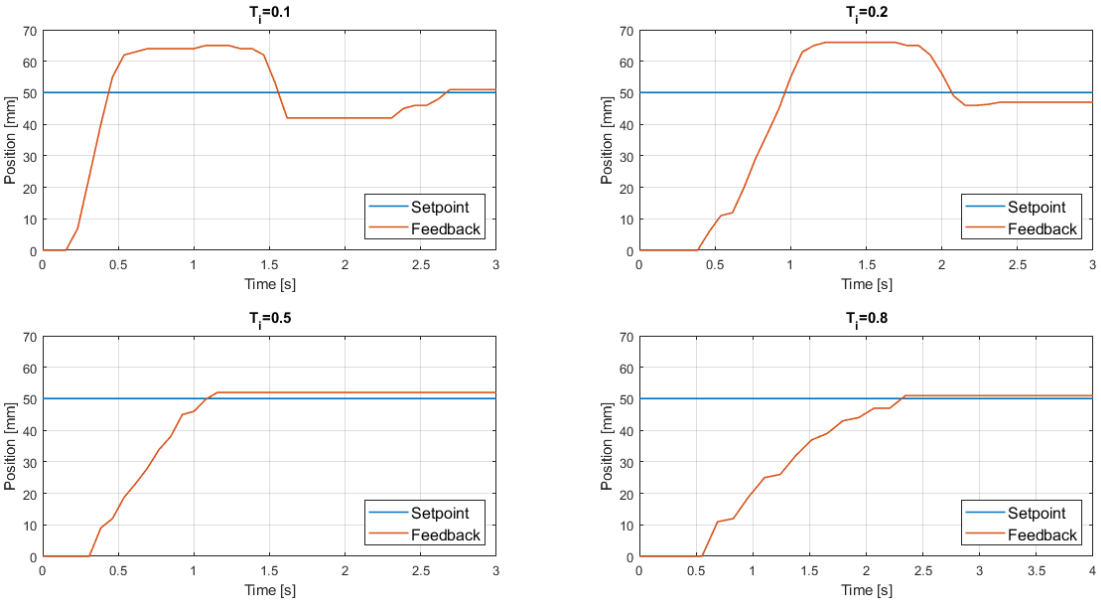


Figure 4.33: Pure integrative control - oil-pneumatic cylinder - 3/2 Valves

The tables below show the variables details, the first for the pneumatic cylinder and the second one for the oil-pneumatic cylinder:

Ti [s]	Rise time [s]	Settling time [s]	Overshoot %
0,5	1,3	1,3	0
1	2,38	2,38	0
2	5,20	5,20	0
4	-	-	0

Table 4.9: Ti - pneumatic cylinder - 3/2 Valves

Ti [s]	Rise time [s]	Settling time [s]	Overshoot %
0,1	0,44	-	30
0,2	0,95	-	32
0,5	1,07	-	4
0,8	2,25	-	2

Table 4.10: Ti - oil-pneumatic cylinder - 3/2 Valves

4.2.3 Pure Derivative

The obtained results using a pure derivative control are not achieved good results, for this reason the pure derivative PID responses, using 3/2 valves, are not shown.

4.3 Autotuning

Using 2/2 valves, the autotuning on the Tia portal is made. By using this particular settings optimal parameters are obtained, which allows to have correct system response. In particular, the auto-tuning is possible in two different way: pretuning and fine tuning, as shown in the fig .

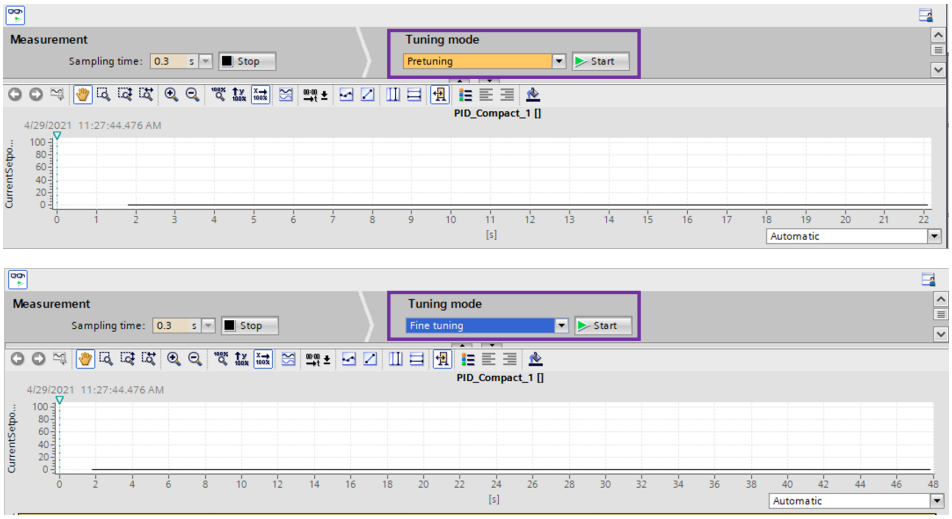


Figure 4.34: Pretuning and fine tuning

After many cycle of optimization the Tia portal returns optimal parameters. These obtained results , must be loaded on the configuration (fig.4.35), and with a simulation it is possible to simulate the system behaviour.

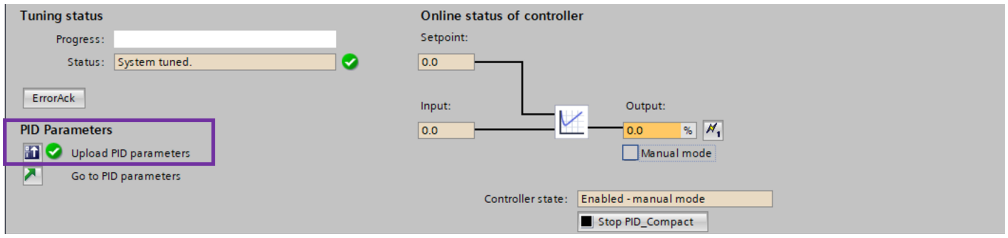


Figure 4.35: PID parameters

For both the cylinders, using the valves 2/2 NC the resulted parameters are:

Figure 4.36: Autotuning PID parameters

So, the obtained system response are:

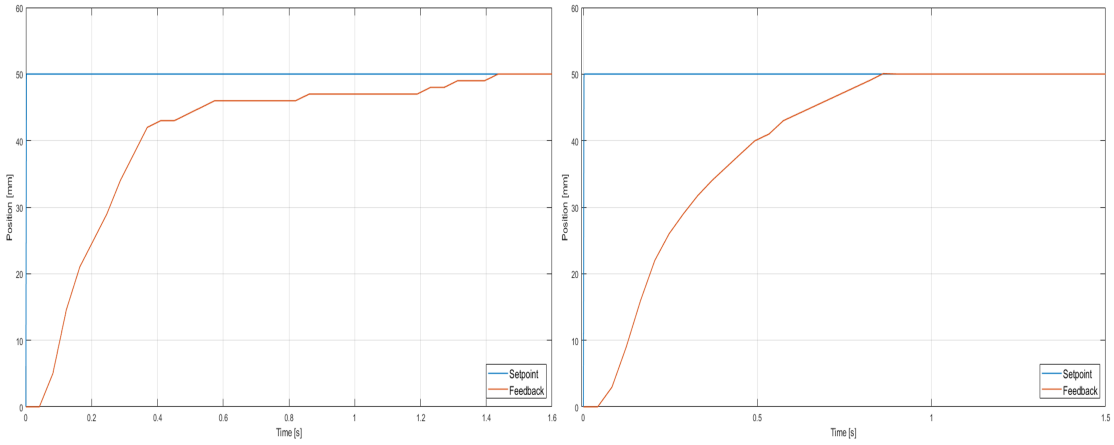


Figure 4.37: Autotuning PID response for both cylinders

As shown in figure 4.37 , it is possible to note that the rise time, for both

cylinders, is not very good. In fact, with other experiments, the obtained rise time results are better. But, on the optimization phase, it is preferred to give more importance to the overshoot, in order to avoid oscillations around the *Setpoint*. This choice is made in order to have a similar first order system characteristic. In particular, it is possible to observe that the overshoot is equal to zero, for both cylinders. Assuming a future application where it leads to minimizing overshoot, thus avoiding oscillations to minimize the impact of the system with any work object.

4.4 Comparison between integrated PWM and CTRL_PWM Block

Given the obtained results, now a comparison between the use of the PWM integrated in the PID block and the use of the CTRL_PWM block external to the PID is made.
The reason why this comparison is made, is to understand if one block is more efficient than the other.
This comparison is made with the oil-pneumatic cylinder with the use of 2/2 valves NC. Trough the measurements, using a proportional PID with Kp=0.5, it is possible to note that with the integrated control PWM, the system response is as desired. While, using the external block CTRL_PWM the valves presents an error.

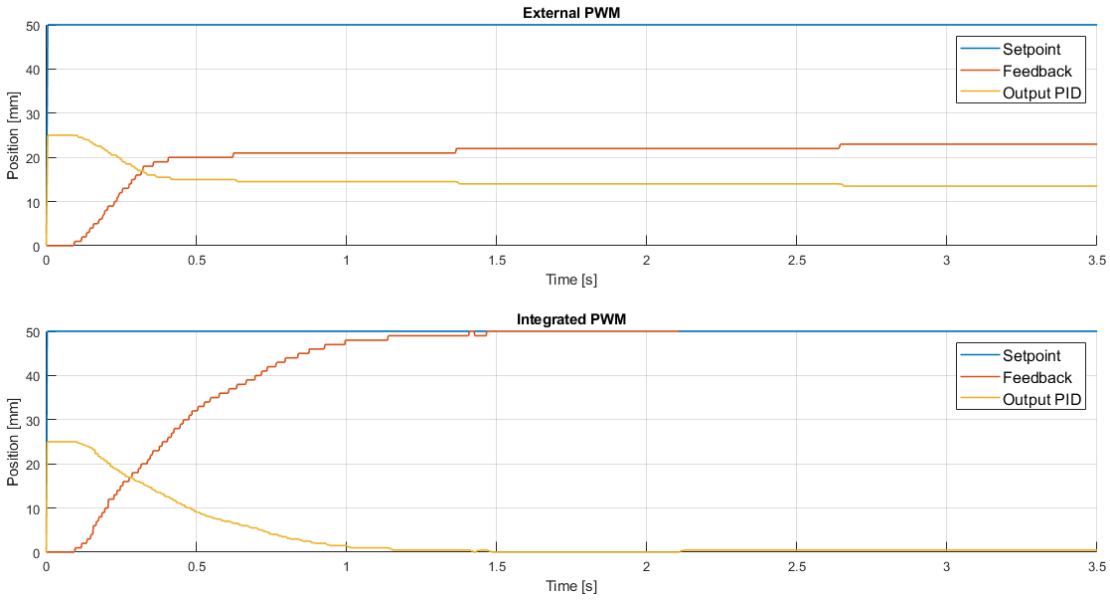


Figure 4.38: Comparison between Integrated and external PWM

With the external block, the valves with a presence of DC around 10% fail to make an opening and closing cycles. The behavior is analyzed using an oscilloscope. The DC is rounded to the its lower decade, so if the output is 35%, there is a DC equal to 30 at the input to the valves.
In particular, if the error is less than 20%, the real value at the PLC output is 10%. In the figure below, the problem is shown.

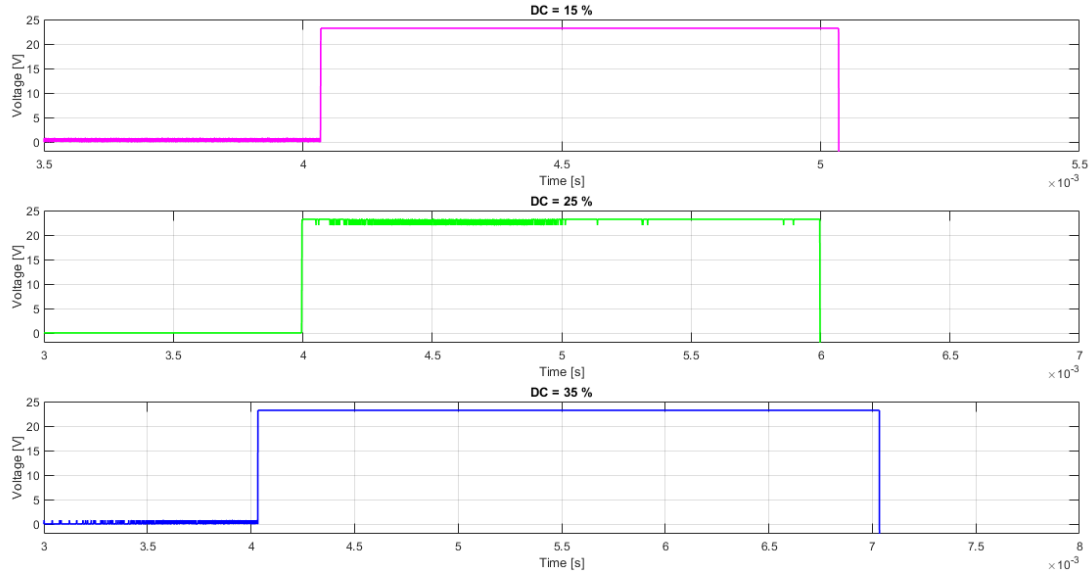


Figure 4.39: Rounded DC

To demonstrate the anomalous behavior of the valves, an experiment is performed by forcing a signal equal to 15% of DC in both configurations.

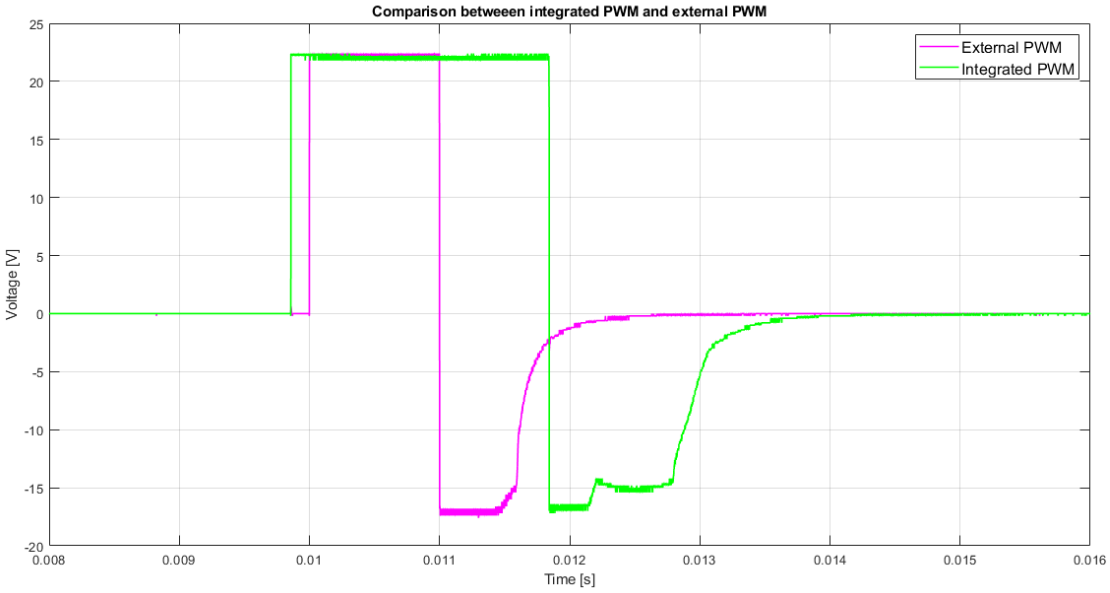


Figure 4.40: Integrated PWM and External PWM - Comparison

It is noted that the curve assigned to the integrated PWM shows a DC equal to

15%, while the external PWM shows the problem highlighted above, so that the DC is equal to 10.
Therefore, as the valves fail to perform their activity, the piston remains stationary, so using the PID on the TIA portal, the best way to control the digital valves is the use of the *Output_PWM* integrated on the *PID_Compact* block.

Chapter 5

Matlab Controls

5.1 SCL generation using PLC Coder

The PLC Coder is a MATLAB's Toolbox used to create a Structured Control Language (SCL) file. This file is created as a text file, that in this case , after a configuration, shows in fig. 5.2, convert a Simulink's Subsystem in a textual language. This file is used inside a segment in the TIA portal. Before using the file, the segment must be converted from the ladder diagram network to SCL segment (fig.5.1).

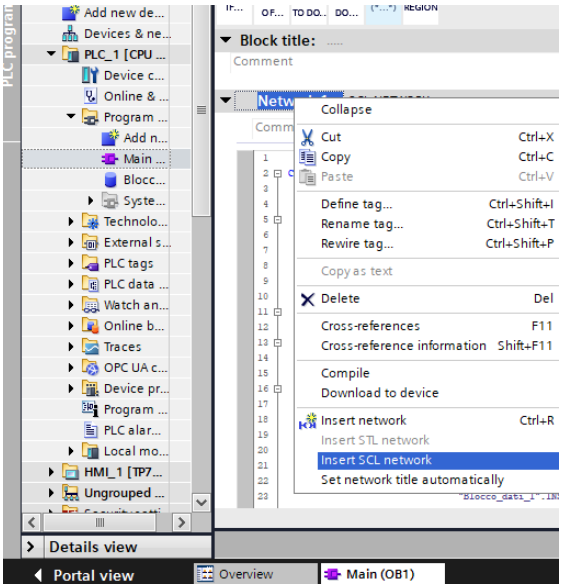


Figure 5.1: New SCL segment

After, simply, the code is copy and paste on the network and with the used of

the Data Block variables is possible to run the project and have the same behaviour as a ladder segment.

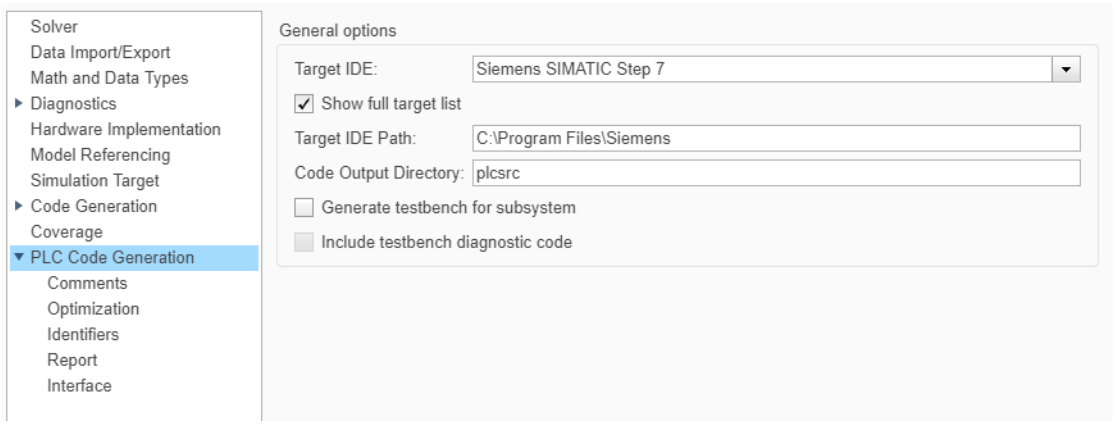


Figure 5.2: Simulink configuration for TIA Portal

Also, it's possible to create a SCL file derived from Simulink and State Flow, as explored below.
A first example, in order to become familiar with this configuration, is done.

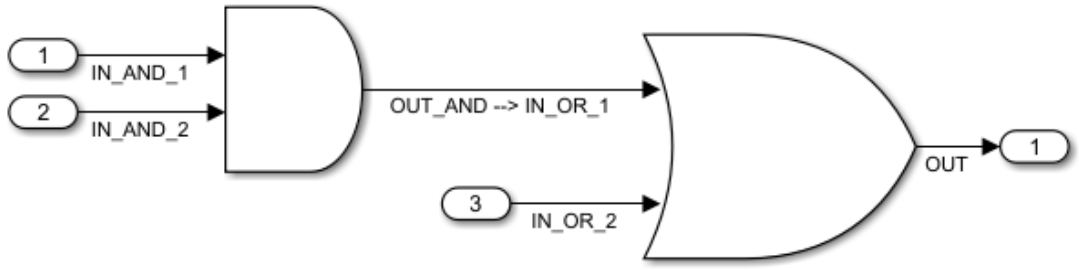


Figure 5.3: Simulink configuration - without Subsystem

In particular, a basic combinatorial circuit is designed. By two different inputs, and AND is performed, from the AND output both with another input, an OR is done. The outputs can be four, as shown below:

AND

IN AND 1	IN AND 2	OUT AND
0	0	0
0	1	0
1	0	0
1	1	1

OR

OUT AND == IN OR 1	IN OR 2	OUT OR
0	0	0
0	1	1
0	0	0
1	1	1

Figure 5.4: Combinatorial Outputs

So, now, creating a Subsystem, is possible to have the SCL file output.

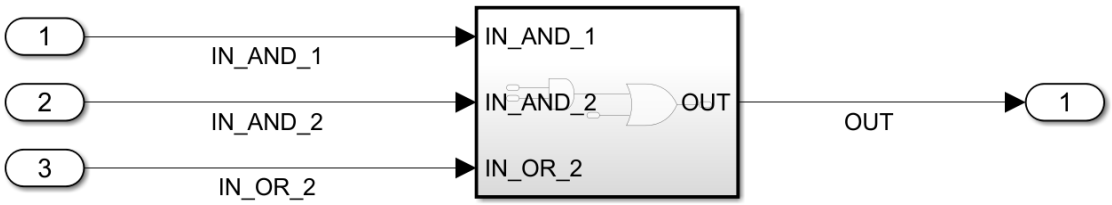


Figure 5.5: Simulink configuration - with Subsystem

So, by right clicking on the Subsystem and set into settings the Siemens SIMATIC Step 7 (fig.5.2), it is possible to do the code generation.

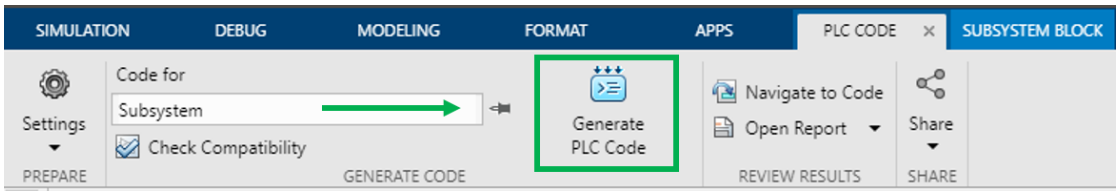


Figure 5.6: Code generation - Subsystem

The SCL file is obtained, so it is possible to copy and paste the code on TIA portal, in a new segment.

```
FUNCTION_BLOCK Subsystem
--the Subsystem is the function
VAR_INPUT -- the inputs variables are three
    IN_AND_1: BOOL;
    IN_AND_2: BOOL;
    IN_OR_2: BOOL;
END_VAR
VAR_OUTPUT --the output variable is one
    OUT: BOOL;
END_VAR
VAR
END_VAR
OUT := (IN_AND_1 AND IN_AND_2) OR IN_OR_2;
--calculation of the output, until is performed the AND , after the OR
END_FUNCTION_BLOCK
```

5.1.1 Simulink

For simplicity, in order to explain the Simulink behaviour it's chosen to implement the base control, with the use of 2/2 valves, algorithm.

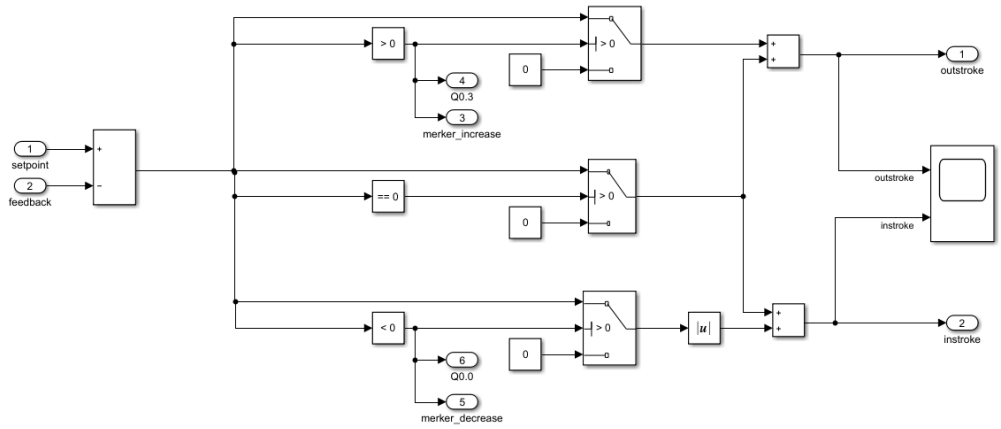


Figure 5.7: Simulink scheme of the base control with 2/2 valves

The implemented algorithm presented in the previous fig.5.7, is the same with respect to the Base Control explained in the second chapter.
In particular, in the following figures the scheme is explained more accurately.

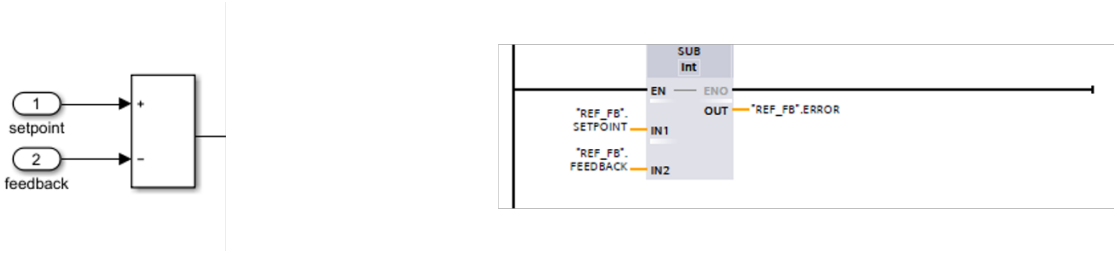


Figure 5.8: Error segments comparison

In fig.5.8 is shown the comparison between the error calculation network on TIA portal with respect the subtract block used in Simulink, them perform the same function, so the different between Setpoint and Feedback.

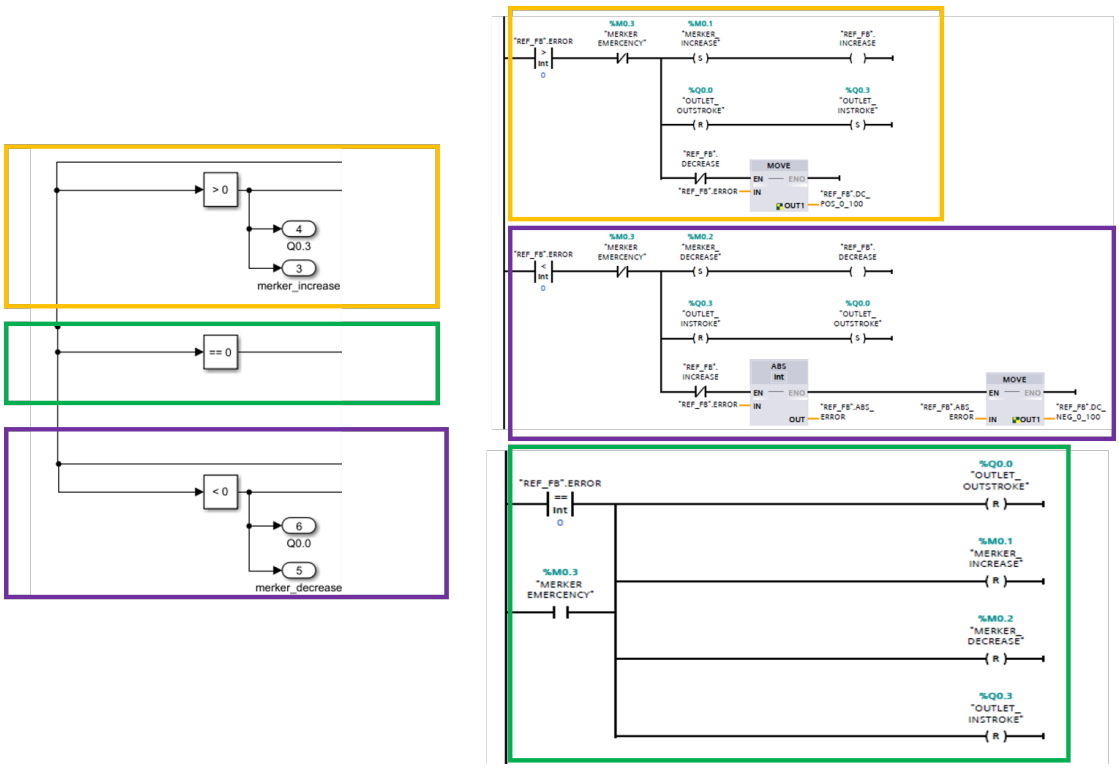


Figure 5.9: Instroke/Outstroke/Equal segments comparison

The calculated error it's compared with the zero value. Depends on the output of the difference, the Simulink implemented, allows to have a direct activation of the output port of the PLC and a proportional control, as in the previous segments used in the chapter 2, and shown in fig.5.9.
After, the outputs of the comparators are used to drive different switches, based on

the results, instroke (with abs block), outstroke or equal are performed on Simulink.

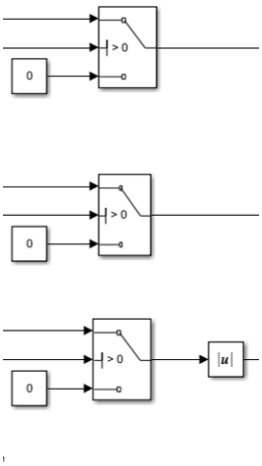


Figure 5.10: Switches

The adders used at the end on Simulink, allow to merge the three signals obtained, in two signals. Thus allowing two distinct phases: outstroke and equal, instroke and equal phases.

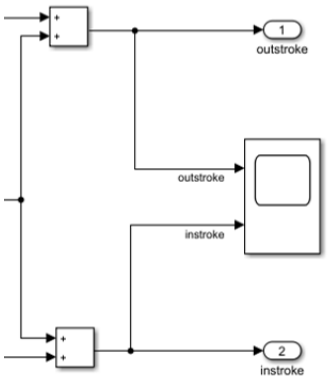


Figure 5.11: Adders

In the following fig.5.12 is shown the Simulink algorithm response implemented on TIA Portal. Once the *Setpoint* arrives, the Simulink algorithm starts to work.

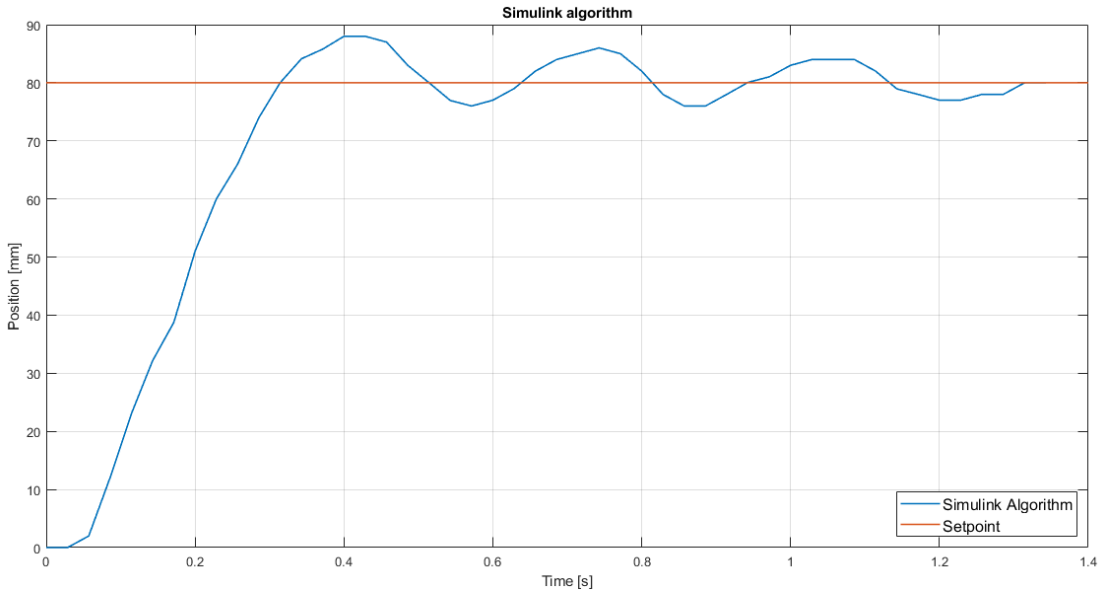


Figure 5.12: Simulink algorithm implemented on TIA portal

The corresponding measures are reported in the table below:

Algorithm	Overshoot %	Rise Time [s]	Settling Time [s]
Simulink Algorithm	6,25	0,4	1,2

Table 5.1: PLC and Simulink data comparison

At the end, is shown the SCL file generated from PLC Coder and used on a TIA Portal SCL segment.

```
CASE "HMI".method_type OF
0:
  "HMI".integrator := 0.5;--integrator is a local variable
  "HMI".filter := 0.0;--filter is a local variable
1:
  "HMI".add := "HMI".set - "HMI".feed;
  --variable add contains the difference between set and feed
  (* IF "HMI".add > 0.0 THEN--check if add is grather than zero
          "merker_dc" := 1;--flag up merker_dc
  ELSIF "HMI".add < 0.0 THEN--check if add is lesser than zero
          "merker_dc" := 1;--flag up merker_dc
```

```
12    ELSIF "HMI".add = 0.0 THEN--check if add is equal to zero
13            "merker_dc" := 0;--in this case merker_dc is 0
14    END_IF;*)
15    "HMI".filter_coeff := (0.0 - "HMI".filter) * 100.0;
16    --filter_coeff is a local variable
17    "HMI".output := ("HMI".add + "HMI".integrator) + "HMI".filter_coeff;
18    --output is a globale variable that contains the information about the error
19    IF "HMI".output >= 100.0 THEN
20    --this section is used for safety, output variable can has value [0.0,100.0]
21            "HMI".output := 100.0;
22            --in order to saturate the maximum possible value equal to 100.0
23    ELSIF "HMI".output <= 0.0 THEN
24            "HMI".output := 0.0;
25            --in order to saturate the maximum possible value equal to 100.0
26    END_IF;
27  --Upgrade of the varible integrator and filter
28  "HMI".integrator := (0.001 * "HMI".add) + "HMI".integrator;
29  "HMI".filter := (0.0 * "HMI".filter_coeff) + "HMI".filter;
30  END_CASE;
```

5.1.2 Stateflow

In order to see the same system in another way, it is chosen to use the Stateflow Toolbox.
Stateflow Toolbox, with respect the Simulink, avoid the use of many blocks (i.e. adder, switch, comparator), but with a simple logic it is possible to manage many signals, with mathematical rules and conditions (as in C language).
In order to create a Stateflow, is necessary to install the corresponding package in MATLAB. After that, the algorithm created is based on the base control of 2/2 valves, designed with a proportional control, as the Simulink project seen before (5.1.1).
So, using basic convention of Stateflow is a more friendly language with respect to the Simulink schemes. So, as regards the code, the difference between *Setpoint* and *Feedback* is calculated. After, based on the result, comparing with the zero value, the resulting cases are three:

- Increase $\rightarrow Error > 0 \rightarrow Outstroke$
- Decrease $\rightarrow Error < 0 \rightarrow Instroke$
- Equal $\rightarrow Error = 0 \rightarrow Feedback = Setpoint$

These cases can be accessed by comparing the difference with zero. In particular, each state is connect to each other. In this way, the system can achieve possible

state from any previous point. As shown in the fig.5.13, it is possible to see the complete Stateflow of the system:

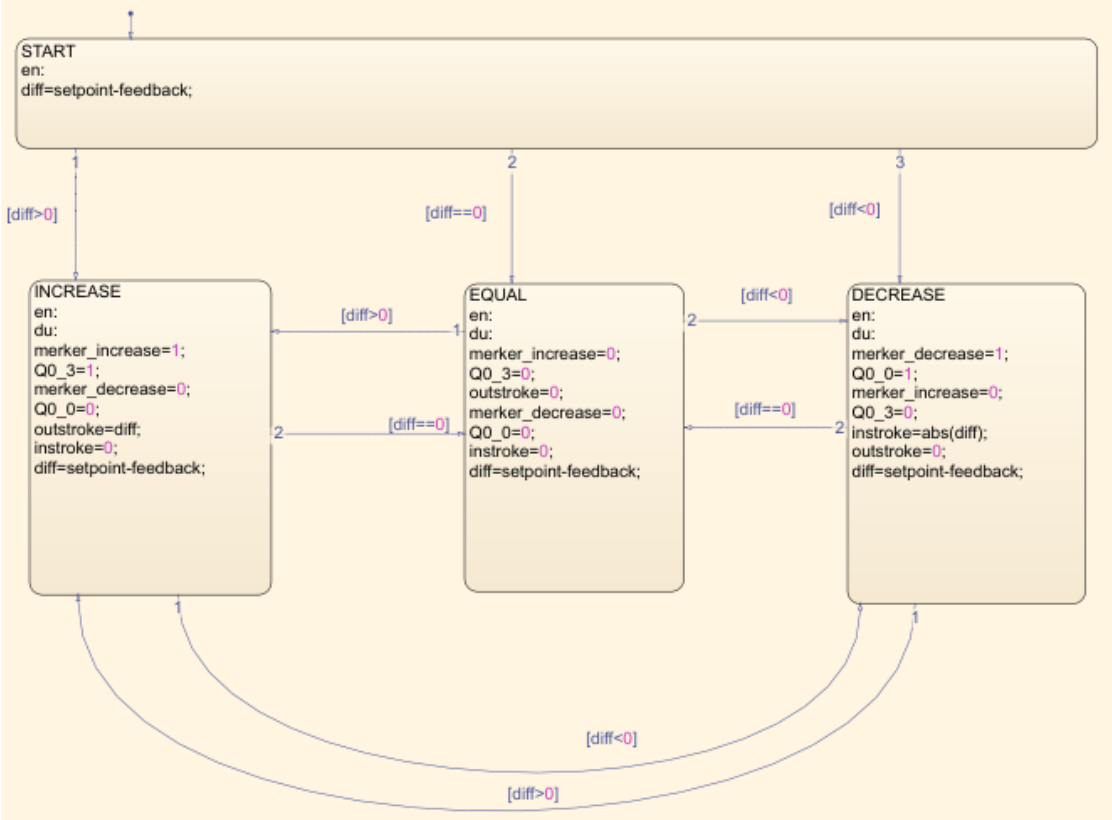


Figure 5.13: Stateflow scheme

The first state is the *Start*, this state is the **default transition**. By definition it has an arrow with a circle, which indicates the starting condition of the chart. Indeed, after this first block, with different arrows you connect the different states with a condition. This condition is included with a square brackets. It's important to point attention on the arrow's direction: the arrow goes from the origin state to the final destination state. For example, if a state has just an arrow pointing to itself, if the condition is verified, the algorithm stuck in a loop on the same state with no way out. Each state are carried out by specific action, by Stateflow's operations, shown in the table below:

Action	Abbr.	Meaning
entry	en	Performs operations when the state becomes active
during	du	Performs actions when the state is active and a specific event arrives

Table 5.2: States operations

For instance, in the *Increase* state, the action **during** in performed, after that the **entry** is activated. So, the boolen functions are updated, and the new difference is calculated. Based on this result, the condition **diff** proceeds forward in the other states. By the using of the Subsystem, using references and scopes the evaluation of the system is performed (fig.5.14)

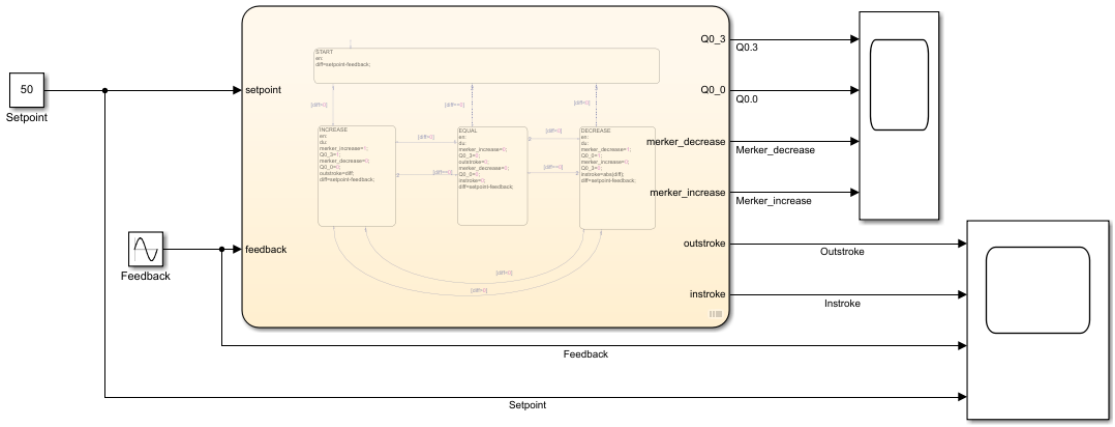


Figure 5.14: Stateflow simulation

The used inputs and outputs are 2 and 6, respectively. The inputs are *Setpoint* and *Feedback* as usual, while the outputs are the four boolean variables, which are Q0.0, Q0.3, merker increase and decrease and two integer values, which are used as the PWM control, as shown in fig.2.16. In order to verify the correctness of the algorithm, it's decided to vary the *Feedback*, keeping the *Setpoint* constant. This choice is made, since the real transducer does not communicate with the Simulink environment. The obtained results in simulation are:

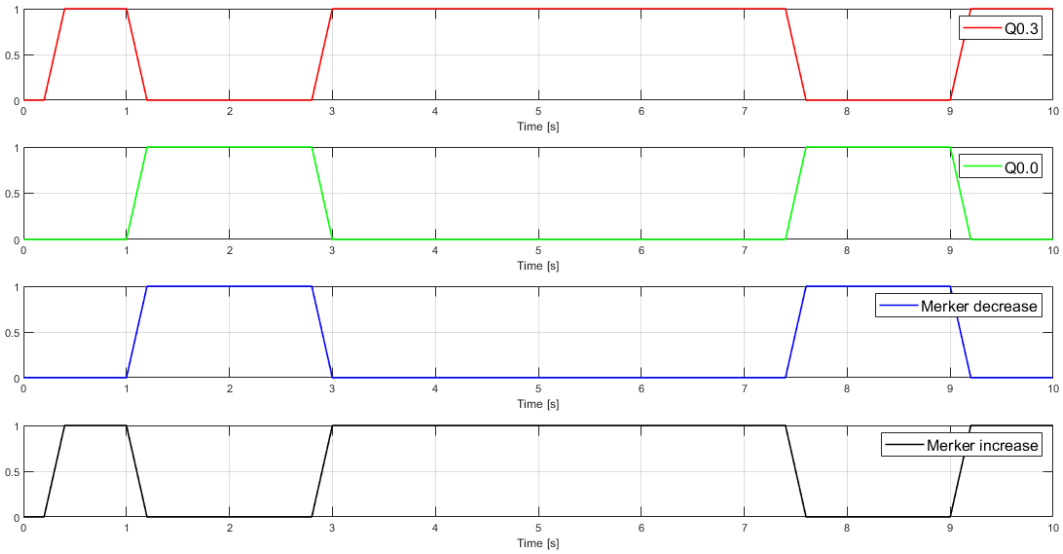


Figure 5.15: Simulink simulation of the Stateflow

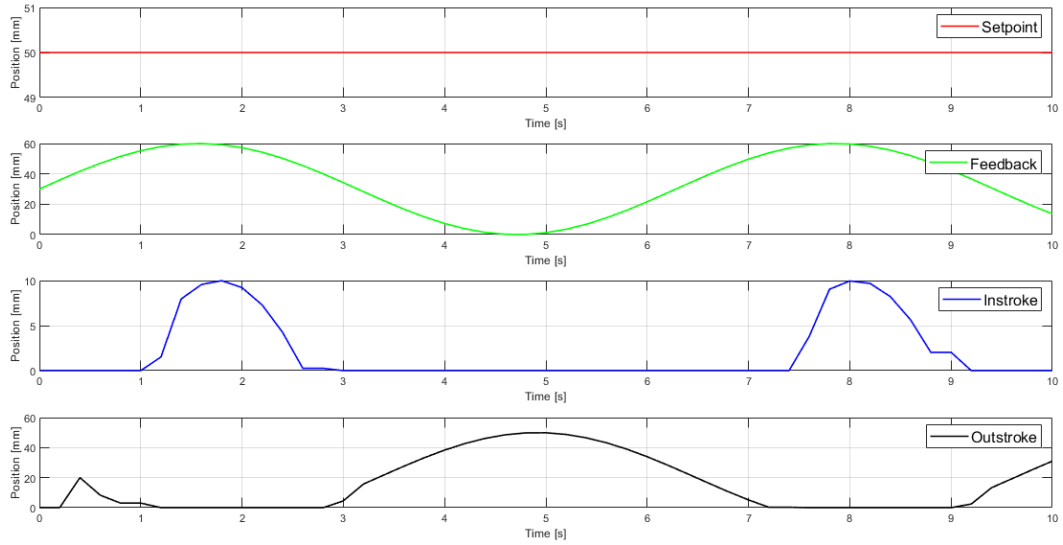


Figure 5.16: Simulink simulation: boolen output

It is possible to note, that the algorithm works in a efficient way. Thanks to the use of a sinusoidal signal in the feedback, a test for each case is verified.
By the SCL code generation on the TIA portal it is imported the file:

```
1 CASE "Blocco_dati_1".ssMethodType OF
2 0:
```

```
3    "Blocco_dati_1".is_active_c2_Chart := 0;
4    "Blocco_dati_1".is_c2_Chart := 0;
5    --these two local variables are inizialized equal to 0
6  1:
7  IF "Blocco_dati_1".is_active_c2_Chart = 0 THEN
8     "Blocco_dati_1".is_active_c2_Chart := 1;
9     "Blocco_dati_1".is_c2_Chart := 4;
10    "Blocco_dati_1".DIFF:="Blocco_dati_1".SETPOINT-"Blocco_dati_1".FEEDBACK;
11    --DIFF contains the difference between SETPOINT and FEEDBACK
12 ELSE
13   CASE "Blocco_dati_1".is_c2_Chart OF
14   1:
15   IF "Blocco_dati_1".DIFF > 0.0 THEN--check if DIFF is grather than zero
16         "Blocco_dati_1".is_c2_Chart := 3;
17   ELSE
18   IF "Blocco_dati_1".DIFF = 0.0 THEN--check if DIFF is equal to zero
19          "Blocco_dati_1".is_c2_Chart := 2;
20   ELSE
21   --code run inside this ELSE only if the previous twp IF are false and so
22   --the DIFF variable is negative.En fact merker DECREASE and OUTLET
23   --OUTSTROKE are imposed equal to 1 in order to perform the decreasing phase.
24          "Blocco_dati_1"."merker DECREASE" := 1;
25          "Blocco_dati_1"."OUTLET OUTSTROKE":= 1;
26          "Blocco_dati_1"."merker INCREASE" := 0;
27          "Blocco_dati_1"."OUTLET INSTROKE" := 0;
28          "Blocco_dati_1".INSTROKE:=ABS("Blocco_dati_1".DIFF);
29          --since is a proportional control INSTROKE is imposed
30          --equal to the absolute value of DIFF
31          "Blocco_dati_1".DIFF :="Blocco_dati_1".SETPOINT-"Blocco_dati_1".FEEDBACK;
32          --DIFF contains the difference between SETPOINT and FEEDBACK
33    END_IF;
34   END_IF;
35   2:
36   IF "Blocco_dati_1".DIFF > 0.0 THEN--check if DIFF is grather than zero
37      "Blocco_dati_1".is_c2_Chart := 3;
38   ELSE
39   IF "Blocco_dati_1".DIFF < 0.0 THEN--check if DIFF is lesser than zero
40      "Blocco_dati_1".is_c2_Chart := 1;
41   ELSE
42   --code run inside this ELSE only if the previous twp IF are false and so
43   --the DIFF variable is equal to 0. En fact al the variables are imposed
44   --equal to 0 in order to perform the stopping phase.
45      "Blocco_dati_1"."merker INCREASE" := 0;
46          "Blocco_dati_1"."OUTLET INSTROKE" := 0;
47          "Blocco_dati_1".OUTSTROKE := 0.0;
```

```
48            "Blocco_dati_1"."merker DECREASE" := 0;
49            "Blocco_dati_1"."OUTLET OUTSTROKE" := 0;
50            "Blocco_dati_1".INSTROKE := 0.0;
51            "Blocco_dati_1".DIFF:="Blocco_dati_1".SETPOINT-"Blocco_dati_1".FEEDBACK;
52     END_IF;
53    END_IF;
54    3:
55    IF "Blocco_dati_1".DIFF < 0.0 THEN--check if DIFF is lesser than zero
56       "Blocco_dati_1".is_c2_Chart := 1;
57    ELSE
58    IF "Blocco_dati_1".DIFF = 0.0 THEN--check if DIFF is equal to zero
59       "Blocco_dati_1".is_c2_Chart := 2;
60    ELSE
61    --code run inside this ELSE only if the previous twp IF are false and so
62    --the DIFF variable is positive.En fact merker INCREASE and OUTLET
63    --INSTROKE are imposed equal to 1 in order to perform the increasing phase.
64       "Blocco_dati_1"."merker INCREASE" := 1;
65           "Blocco_dati_1"."OUTLET INSTROKE" := 1;
66           "Blocco_dati_1"."merker DECREASE" := 0;
67           "Blocco_dati_1"."OUTLET OUTSTROKE" := 0;
68           "Blocco_dati_1".OUTSTROKE :="Blocco_dati_1".DIFF;
69           "Blocco_dati_1".DIFF:="Blocco_dati_1".SETPOINT-"Blocco_dati_1".FEEDBACK;
70           --DIFF contains the difference between SETPOINT and FEEDBACK
71     END_IF;
72    END_IF;
73    ELSE
74    --this section allow the correct upgrade of the "is_c2_Chart" local
75    --variable and upgrade the entire code
76    IF "Blocco_dati_1".DIFF > 0.0 THEN
77       "Blocco_dati_1".is_c2_Chart := 3;
78    ELSE
79    IF "Blocco_dati_1".DIFF = 0.0 THEN
80           "Blocco_dati_1".is_c2_Chart := 2;
81    ELSE
82    IF "Blocco_dati_1".DIFF < 0.0 THEN
83           "Blocco_dati_1".is_c2_Chart := 1;
84    END_IF;
85    END_IF;
86    END_IF;
87   END_CASE;
88  END_IF;
89  END_CASE;
```

Otherwise, simulating the Stateflow algorithm on TIA portal the following results are obtained:

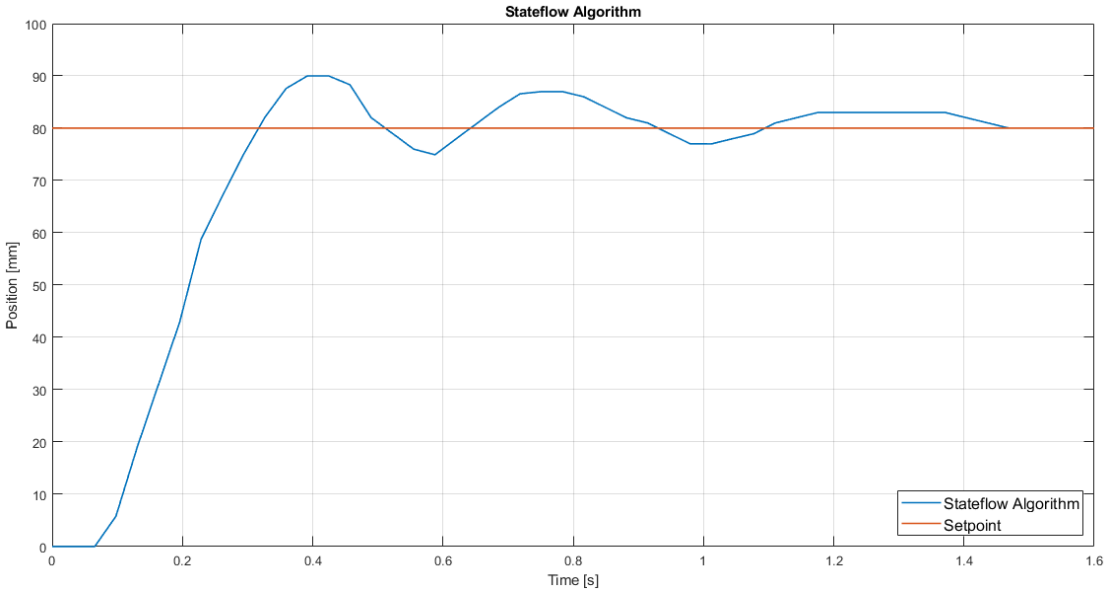


Figure 5.17: Stateflow algorithms on TIA portal

So, the obtained results are:

Algorithm	Overshoot %	Rise Time [s]	Settling Time [s]
Stateflow Algorithm	12,5	0,317	1,47

Table 5.3: Stateflow results

After these experiments, the conclusion are that this PLC Coder is very useful and has a greater potential with respect the ladder diagram made on the TIA portal. Because by a block scheme, which in a simply way, made the designed algorithm. Moreover, once the text file is created the reliability of the SCL code is quite null and also bit intelligible .

5.2 Comparison between PLC - Simulink - Stateflow algorithms

Based on the results a comparison between these different algorithms is made.

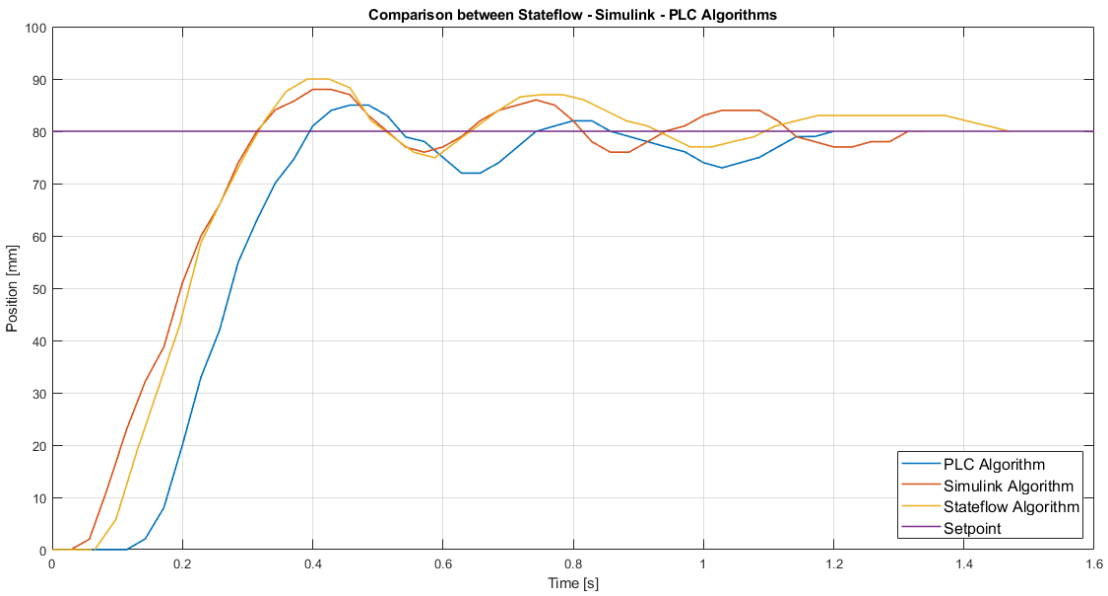


Figure 5.18: Comparison PLC-Simulink-Stateflow algorithms

So, the comparison of the final results are:

Algorithm	Overshoot %	Rise Time [s]	Settling Time [S]
PLC	6,25	0,4	1,2
Simulink	10	0,3142	1,3142
Stateflow	12,5	0,317	1,47

Table 5.4: Final results

By the table 5.4, it is possible to note that the Simulink response is better than the system designed by the PLC. This because the implementation with the SCL created by the code generation with the PLC Coder is shorter with respect the created ladder diagram created on the TIA portal. Instead, the settling time and

the overshoot are better in the simulation obtained by the PLC algorithm.
For what concern the Stateflow algorithm with respect the Simulink algorithm, it is possible to note that the rise time of the Stateflow algorithm is better than the PLC algorithm. Instead, the overshoot and the settling time are better with the PLC control.
In general, the PLC code have better stability, but worst rise time response with respect the MATLAB's controls.

5.3 Simscape Toolbox

With Simscape it is possible to create models of physical systems within the Simulink environment.
Simscape allows to build link-based physical component models that integrate directly with block diagrams and other modeling paradigms. Also, it is possible to model hydraulic actuators and pneumatic systems by assembling the fundamental components in a schematic. Simscape add-on products provide more complex analytics components and capabilities.
Simscape helps to develop control systems and test system-level performance.
The Simscape language is based on MATLAB, which enables text-based authoring of physical modeling components, domains and libraries. Moreover, allows to parameterize models using MATLAB variables and expressions, as well as design control systems for the physical system in Simulink.
In this project, the proposed work is to design the same panel that is used until now. So, proportional control and PID control using a pneumatic cylinder, and four valves 2/2 NC are made.

5.3.1 Simscape design

In order to design the same environment used before, for the other experiments, a Simscape and a Simscape Fluids Toolboxes are installed on MATLAB. Using the Simulink scheme designed in the chapter above, a pneumatic circuit is made. In particular, the pneumatic circuit is created.

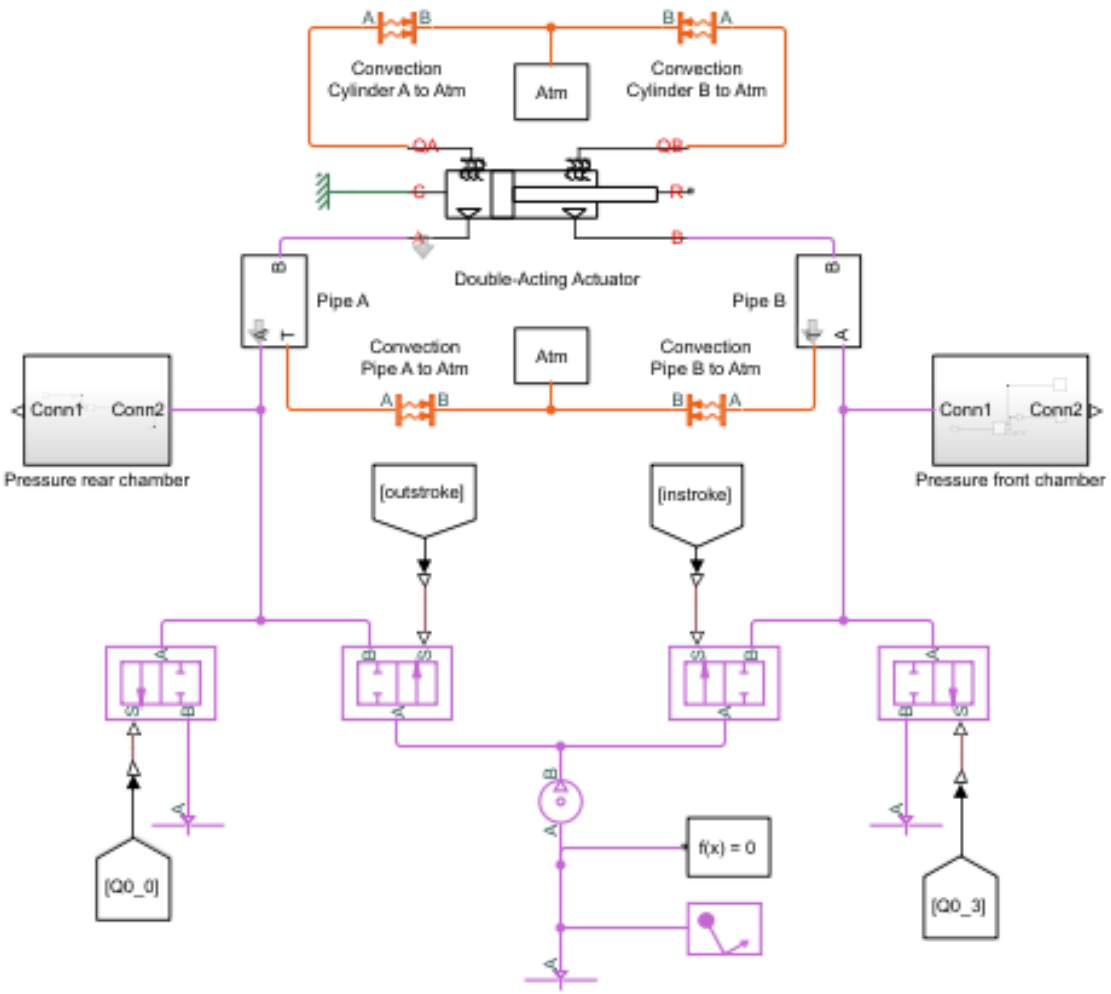


Figure 5.19: Simscape design

As shown in fig.5.19, four valves, a pneumatic cylinder, supply and exhausts are present. In order to have a similar behaviour these configurations are fix:

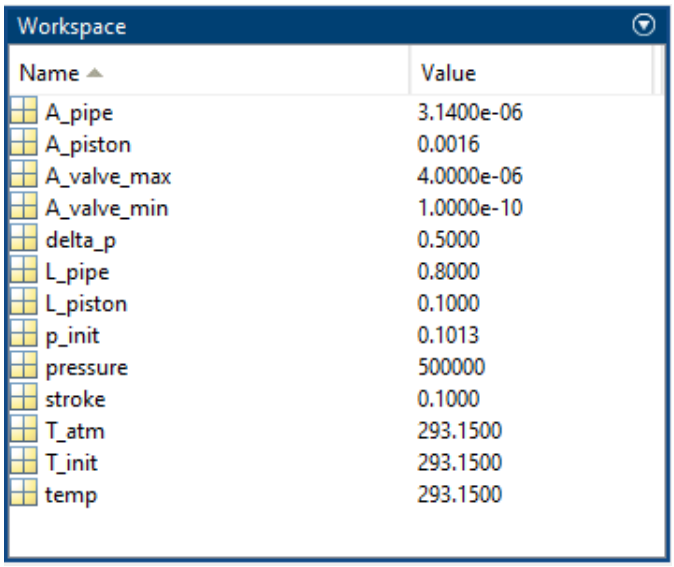


Figure 5.20: Simscape Data

These data are the parameters of the real systems, discussed above. In particular, now is analyzed each part of the Simscape design.
The main part is regards the cylinder, which is provided by MathWorks itself, by starting in the command window of MATLAB "*ssc_pneumatic_actuator*".

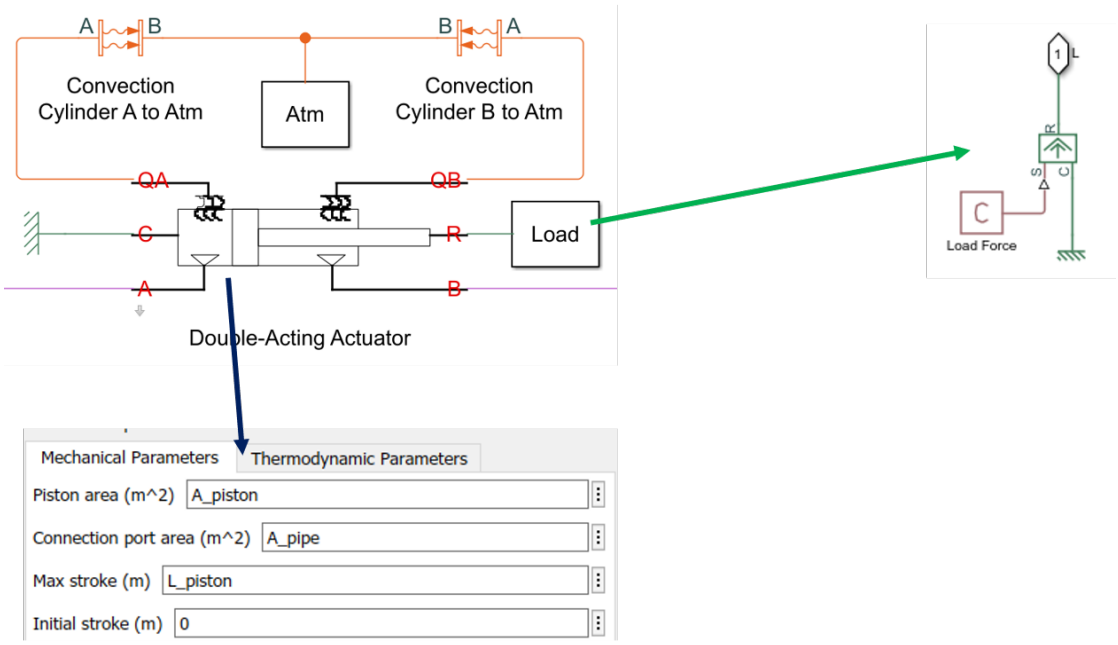


Figure 5.21: Simscape - Double-acting cylinder

Ports A and B receive pressure from the valves. Ports QA and QB model heat exchanges inside the two chambers. Ports C and R are mechanical ports.
Also, is add a LOAD, where the “Ideal Force Source” produces force proportional to the input physical signal. A positive signal in S generates a force acting from C to R.
The second characteristic that is changed , as a real system, is the pipe’s model.

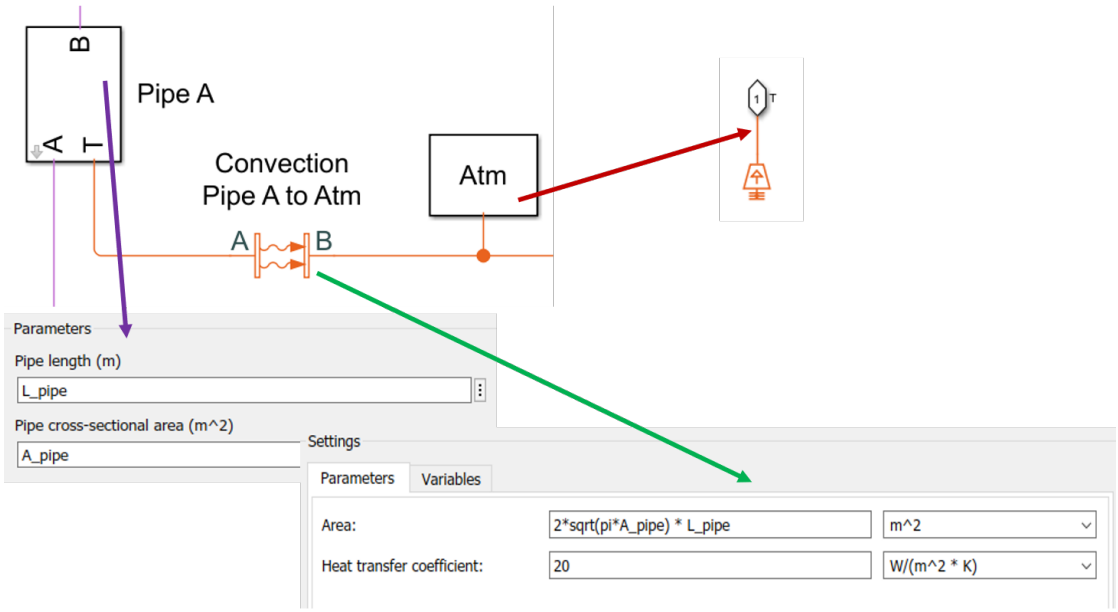


Figure 5.22: Simscape - Pipe’s model

The "Pipe A" model the characteristic of the pipe, in terms of pipe’s length and cross section’s area. As regards the block of "Convection Pipe A to Atm" , it model convective heat transfer due to fluid motion and a “Temperature Source”, with a constant absolute temperature (293,15K) at the port.
For what concern valves, the idea is the same of the previous experiments designed with a 2/2 valves. In particular, on Simscape, the valves are 2/2 valves with this configurations: Ports A and B are the gas conserving ports. When the physical signal, present on S port, is positive, the connection between A and B is opened, otherwise no. Also, an important aspect is that in these valves there are no heat exchange with the atmosphere.

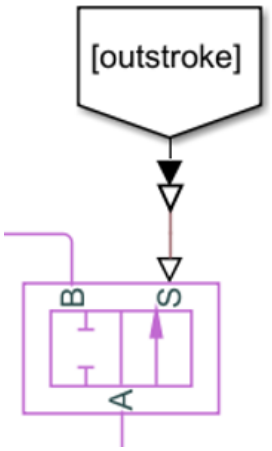


Figure 5.23: Simscape - 2/2 Valves

The Outstroke port and the other signals with "From" blocks arrives from the controller (fig. 5.25) and are the same behaviours of the previous one cases.

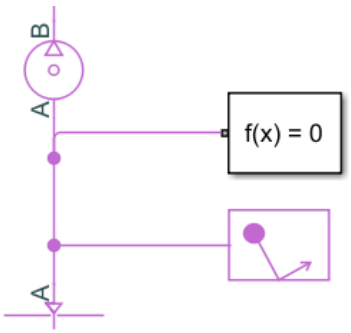


Figure 5.24: Simscape - Supply

At the end, the focus is on the supply section. The valve's port A is linked to pressure supply, solver configuration and to reservoir (G).
The "pressure source" can maintain a constant pressure. A positive differential pressure leads to a pressure port B greater then the one at port A.
This block sets constant boundary conditions in a gas network. The volume of gas inside the reservoir is assumed infinite. Instead, the last block is the "solver configuration", which is only necessary for Simscape models. It is not a physical part in the system, but it requires solver setting information for simulation.

5.3.2 Proportional control

The valves actuation is managed by a proportional control.

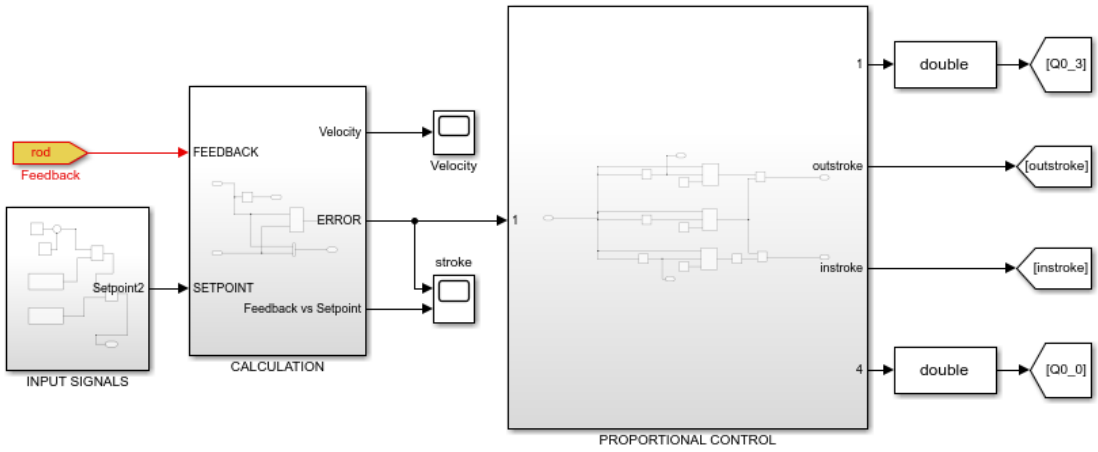


Figure 5.25: Simscape Control

By the fig.5.25 is possible to note which are present three Subsystem blocks: Proportional Control Subsystem, Input Signals Subsystem and Calculation. In the first one block there are one input e four outputs, and its the same Simulink scheme used before (fig. 5.7). Depends on input signal the outputs are activated in different ways.
In the second block, there is one output, and offers the possibility to manage three different input signals, which are used to perform the simulations. In particular, three different cases are studied: Step input, sinusoidal input and sawtooth input, as shown in figure below.

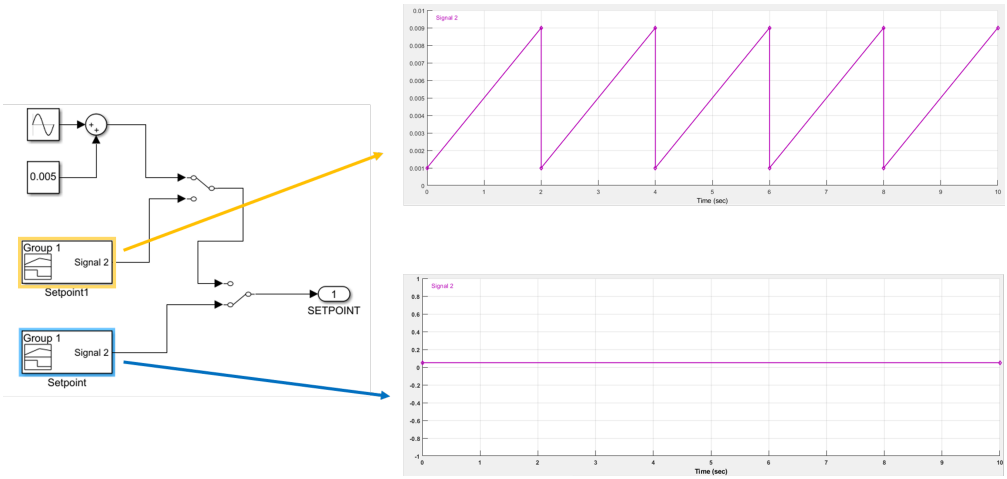


Figure 5.26: Input Signals

The third block allows to calculate four signals: velocity, error, feedback and setpoint. In particular, just to have the possibility to manage signals independently:

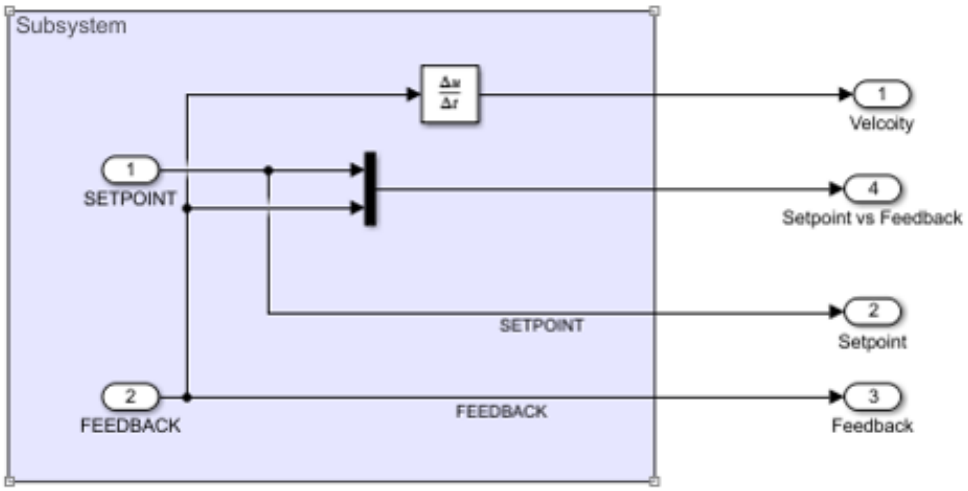


Figure 5.27: Used Signals - Third subsystem

These used signals are the same used in the real system. So, after the simulation is performed. In particular, 3 different inputs are proved (fig. 5.26).

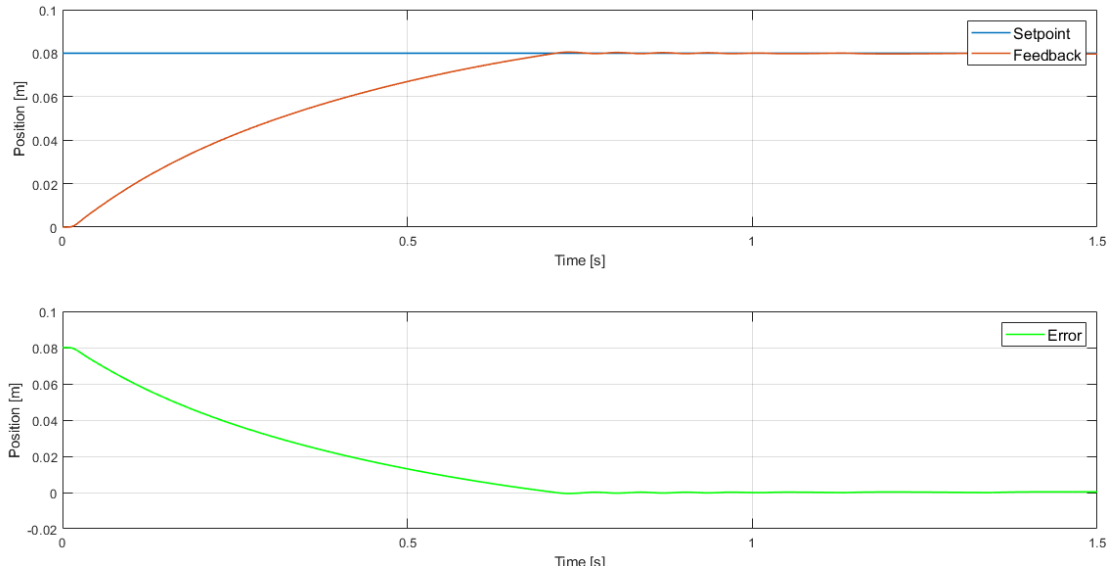


Figure 5.28: Step signal

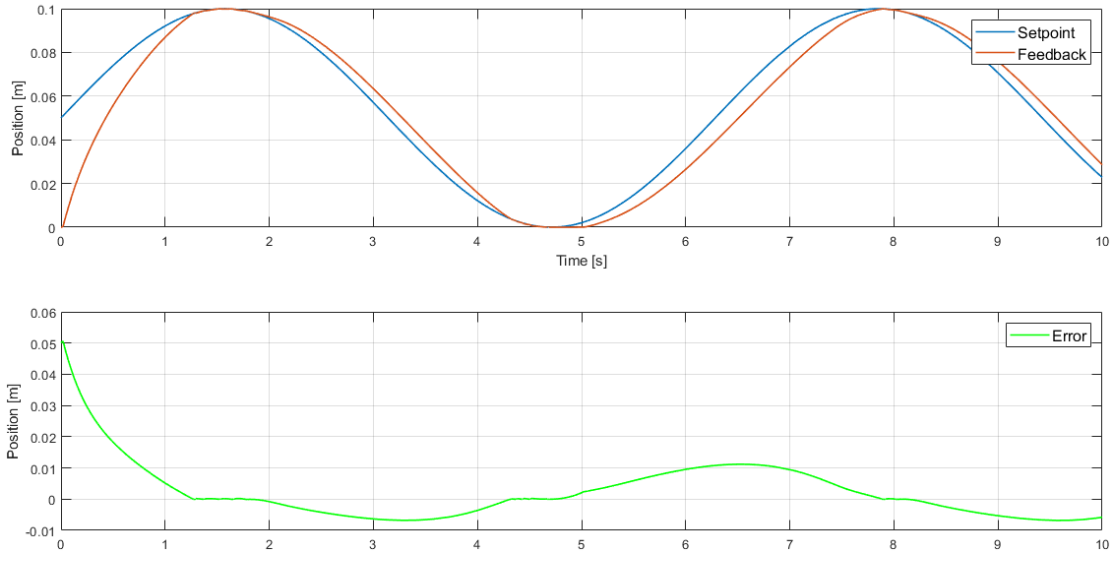


Figure 5.29: Sinusoidal signal

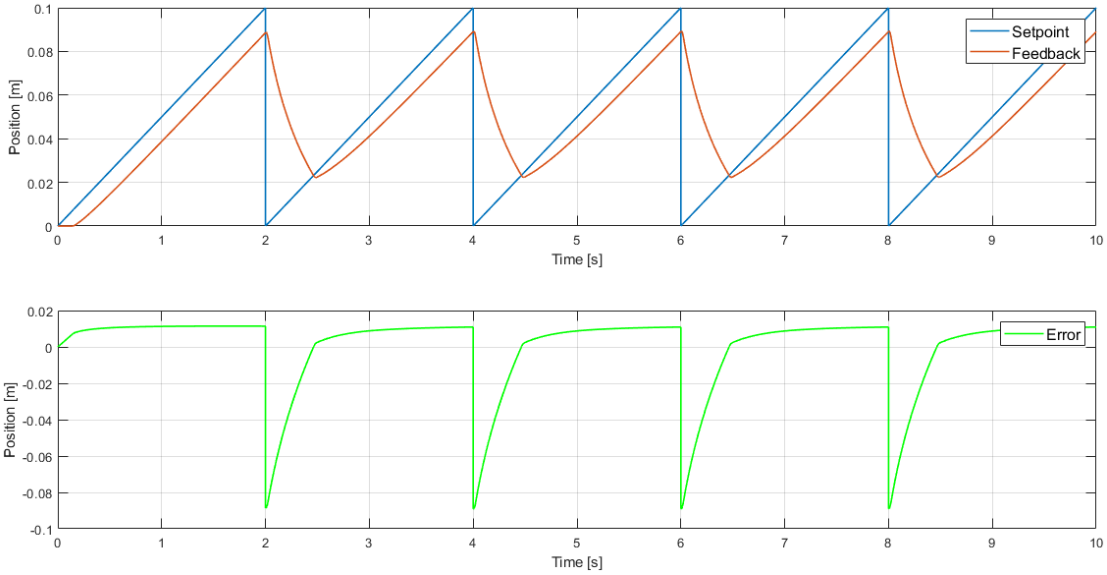


Figure 5.30: Sawtooth signal

The final outputs are the same seen until now: Q0.0, Q0.3 and markers, that have the same meaning as before.

Moreover, the behaviour of the tracking error obtained by the using of a sinusoidal input and a step input are very similar with respect to the real approach.
In the decreasing phase are present some ripples in the instroke phase. Instead, with a sawtooth signal input, the response presents a delay and also, a mismatch when setpoint is less than feedback. This phenomena prove that the model is comparable with a real system simulated on the bank.

5.3.3 PID control

In this section, a PID control technique is design on Simscape Toolbox.

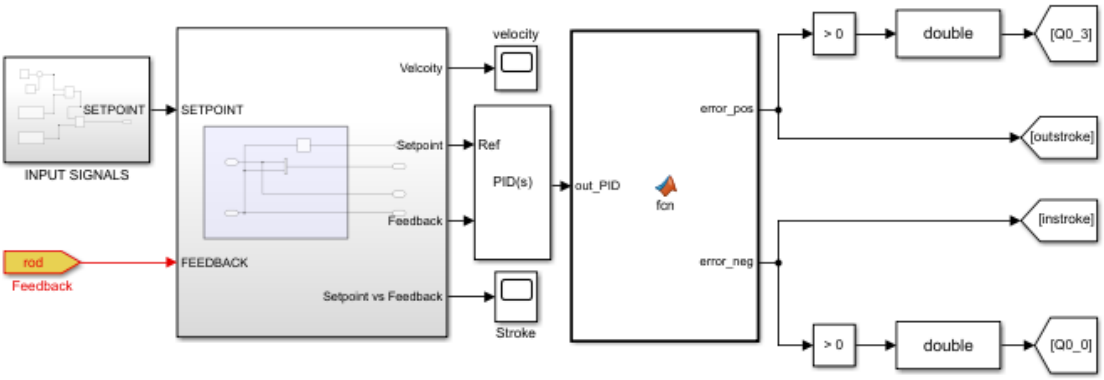


Figure 5.31: PID control - Simscape

The experiment is made with a PID in order to compare the results obtained in the chapter 4, above.
The input signal Subsystem is the same of the section above (fig.5.26), also the second Subsystem (fig. 5.27). The coefficients are imposed equal to zero. While, the used proportional coefficients are:

- Kp=0.1;
- Kp=0.5;
- Kp=1;
- Kp=1.2;

The PID's output is used as input to a function (5.32). This function is made in order to emulate the real system, so the increase phase and the decrease phase of the cylinder, so the error is created.
The algorithm is based on the output of the PID compared with the zero value, in order to create the 'error'.

```
function [error_pos,error_neg] = fcn(out_PID)
if out_PID > 0
   error_pos=out_PID;
   error_neg=0;
else
    if out_PID < 0
        error_pos=0;
        error_neg=abs(out_PID);
    else
        error_pos=0;
        error_neg=0;
    end
end
```

Figure 5.32: Function: Output PID

By using the same input signals Subsystem used in the proportional control, it is possible carry out the same experiments.

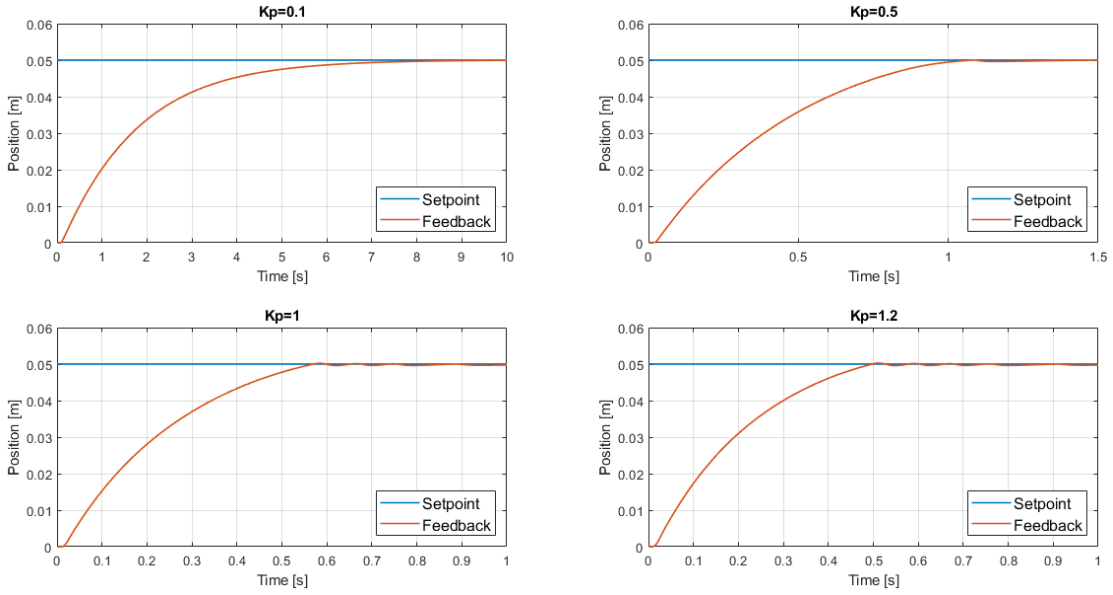


Figure 5.33: System response using PID - Simscape

By the fig.5.33 it is possible to note that with Kp=0.1 the obtained result is comparable with the real experiment, indeed its rise time is more or less 7s. Moreover, with Kp=1 and Kp=1.2 there are some overshoots and undershoots as before.

PID's simulations with varying parameters

Other experiments are made changing some parameters:

- Pipe's radius

- Valve's flow rate

This because you want to test the system under different conditions, to see the system response in case these parameters change.
All the experiments are made with a reference experiment in order to compare the results. In particular with a pipe's radius equal to 0.001 m and a valve's flow rate equal to 135 l/min, as the real system.
In particular, the first experiments are made with changing the pipe's radius as shown in the table below:

Radius [m]	Overshoot %	Rise time [s]
0,001	0,58	0,613
0,002	0,99	0,454
0,003	0,81	0,399

Table 5.5: Pipe's radius

By the simulation the obtained responses are:

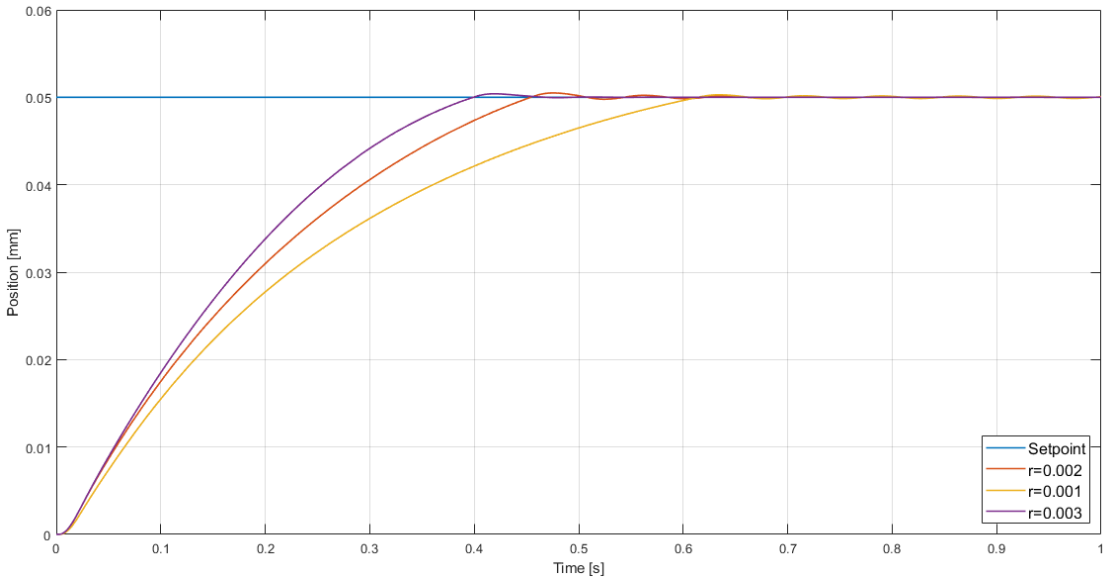


Figure 5.34: Pipe's radius responses

As shown in fig.5.34, it is possible to note that using the largest standard sizes of pipe's cross section area, the performance improved.
So, now changing the flow rates, the obtained results are:

Valve's flow rate [l/min]	Overshooot %	Rise time [s]
67,5	0,3	0,8
135	0,99	0,454
270	1,6	0,282

Table 5.6: Valve's flow rate

By the simulation the obtained responses are:

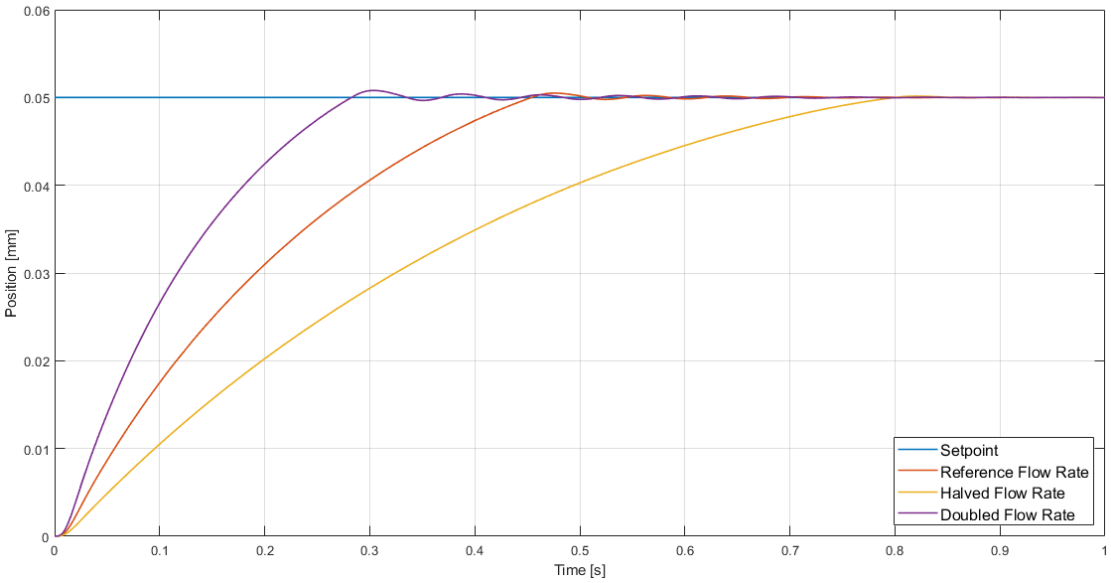


Figure 5.35: Valve's flow rate responses

So, now is possible to see that the improvement and the worsening are greater. Finally, another experiment is made changing both pipe's radius and valve's flow rates, in particular, doubling and halving at the same time valve's flow rates and increasing the pipe's radius.
By the table, the results shown are :

Radius [m]	Valve's flow rate [l/min]	Overshoot %	Rise time [s]
0,001	67,5	0,43	0,973
0,002	135	0,99	0,454
0,003	270	2,2	0,224

Table 5.7: Valve's flow rates and Pipe's radius

The responses in time are:

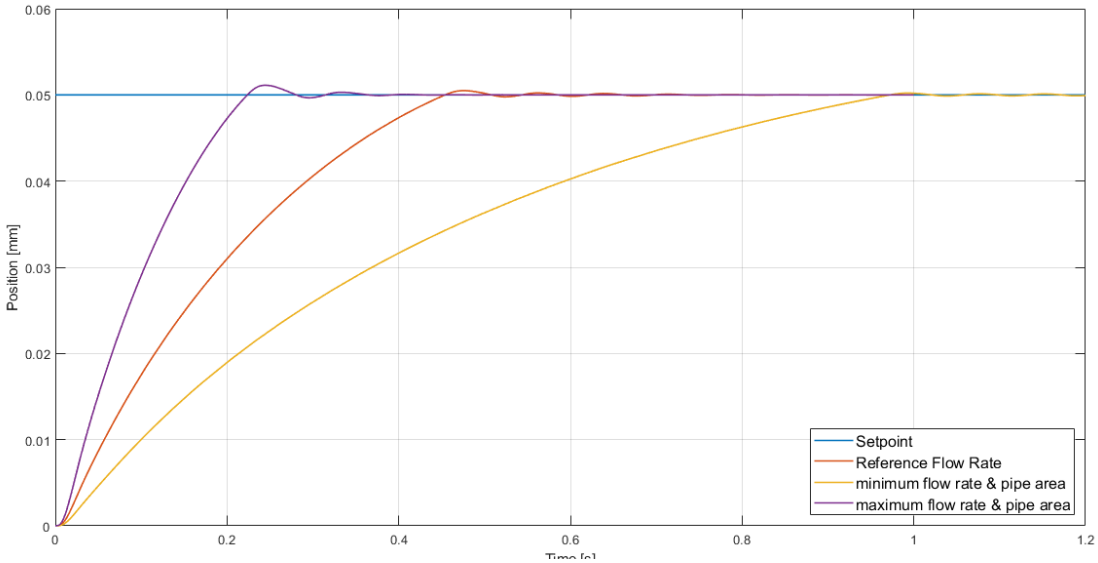


Figure 5.36: Valve's flow rates and Pipe's radius responses in time

In conclusion, it is possible to say that the better results are obtained by using the highest parameters; instead, using the the minimum parameters the obtained values are worst.

5.3.4 Autotuning

In MATLAB, also, there is a possibility to use the PID autotuning. This Toolbox allows to obtained PID parameters as a function of response time and transient behavior.
Subsequently these parameters are loaded to the PID block. To obtain the results and therefore the desired behavior it is possible to carry out the experiments with two different ways:

- using an ideal form of the PID
- using a Linear time-invariant (LTI) block of the plant

The analyzed case is the second. Starting from the PID block, it is possible to set the different gains inside the block: proportional, derivative, integrative. Once this is done, using the TUNE toolbox you can choose the response you want to get, as follows.

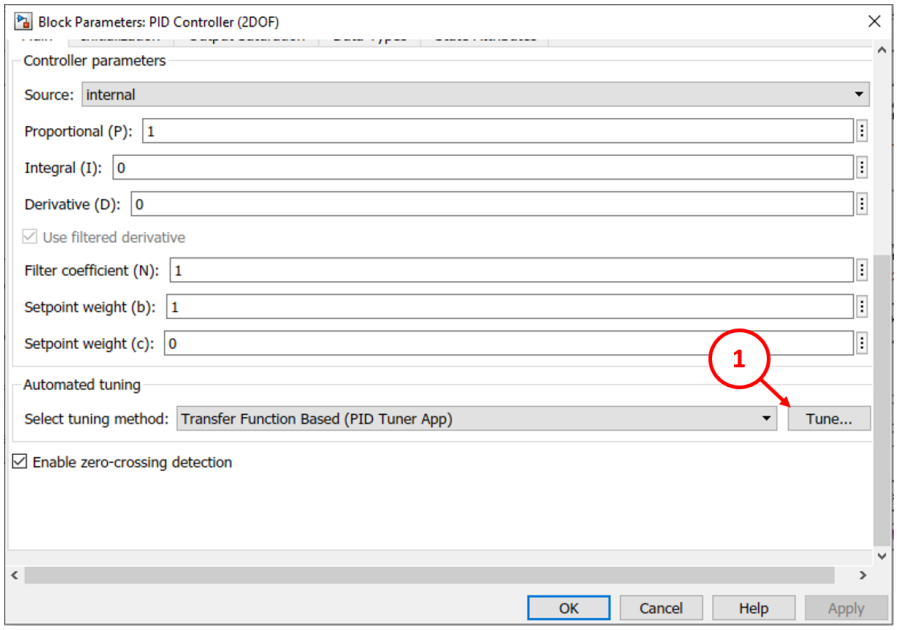


Figure 5.37: Starting Tune

So, the Toolbox is opened:

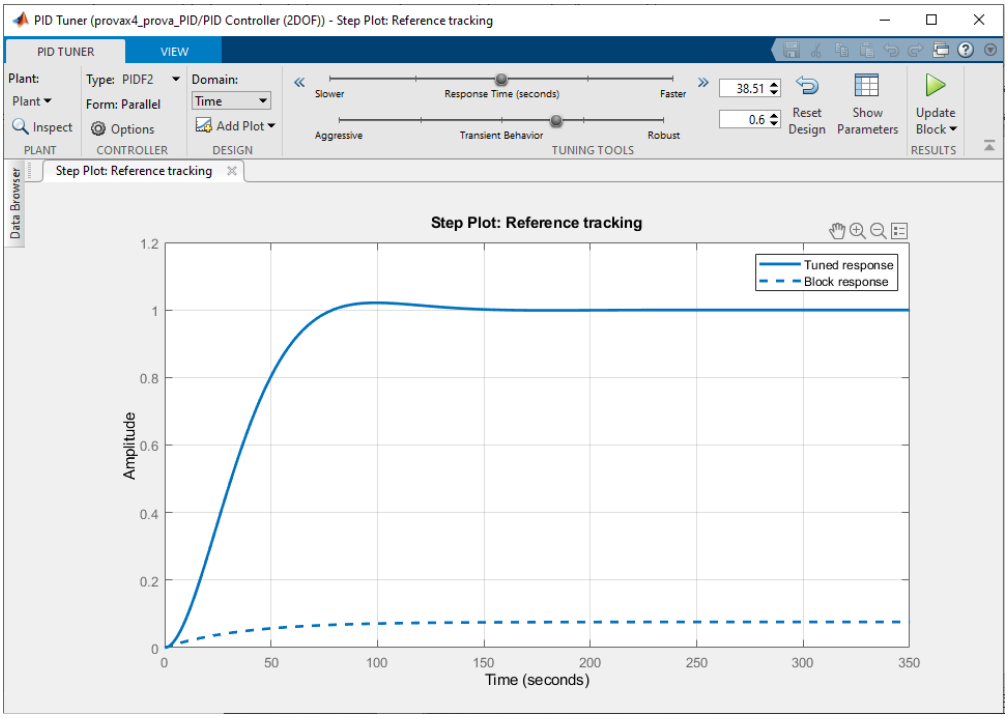


Figure 5.38: PID Tuning

So, the Step plot appears and two different curve are present: tuned response and block response. The first is the future response in case the auto tuning parameters are loaded into the block, and the block response is the current response without changing parameters, therefore with the parameters chosen by the user.

With the using of the second method,is used the Linear time-invariant transfer function (LTI TF) block, so the procedure is:

- 1. Tune (fig.5.37)
- 2. Explore data browser
- 3. Plant
- 4. Export
- 5. Update block

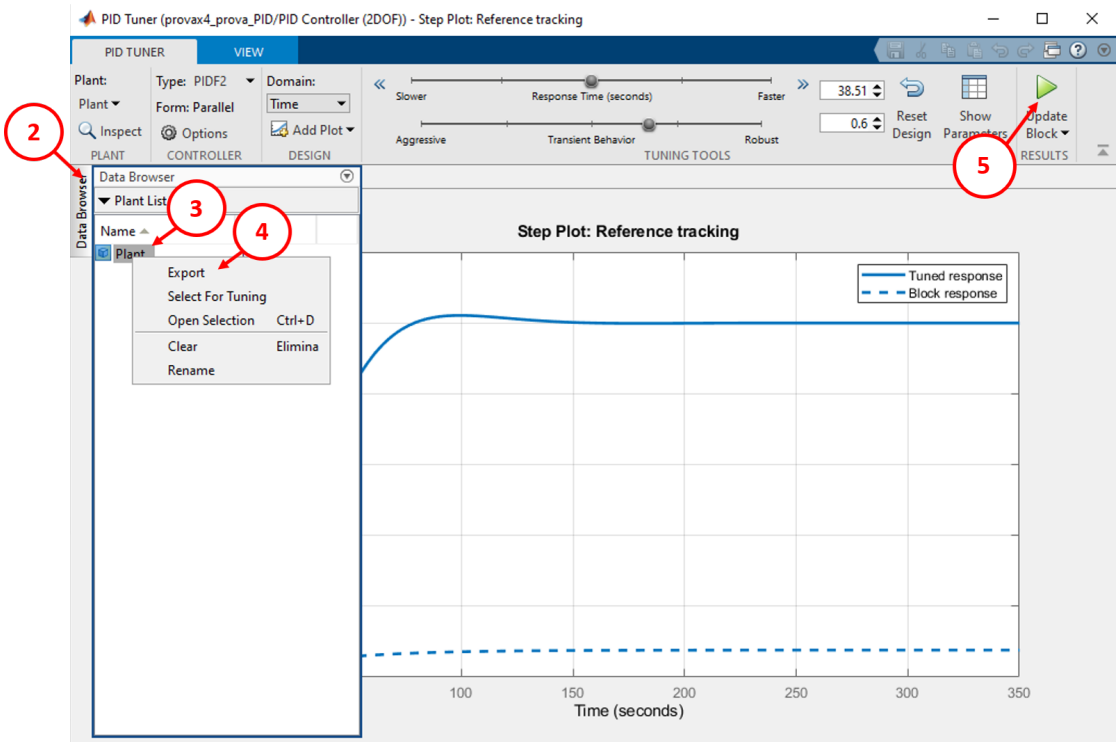


Figure 5.39: Plant - LTI block

As shown in fig.5.39 , it is possible to export into the workspace the plant TF. Using a new Simulink file, the output of the PID is sent to the plant as input.

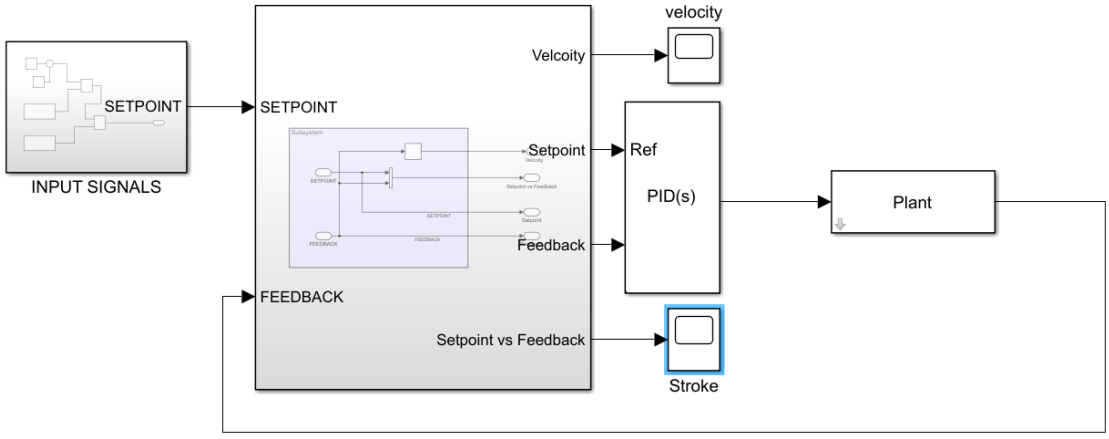
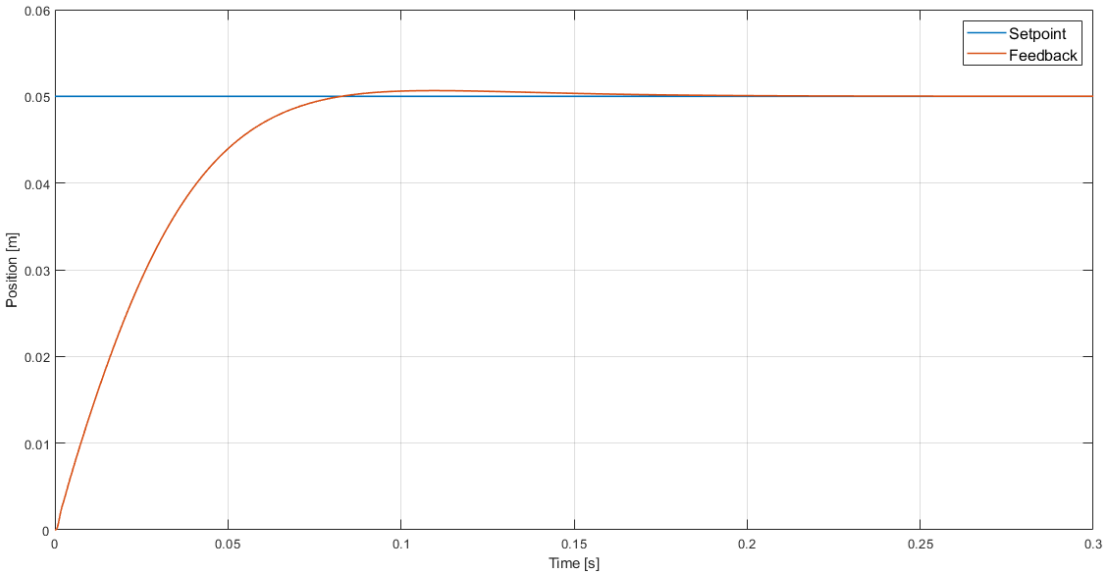
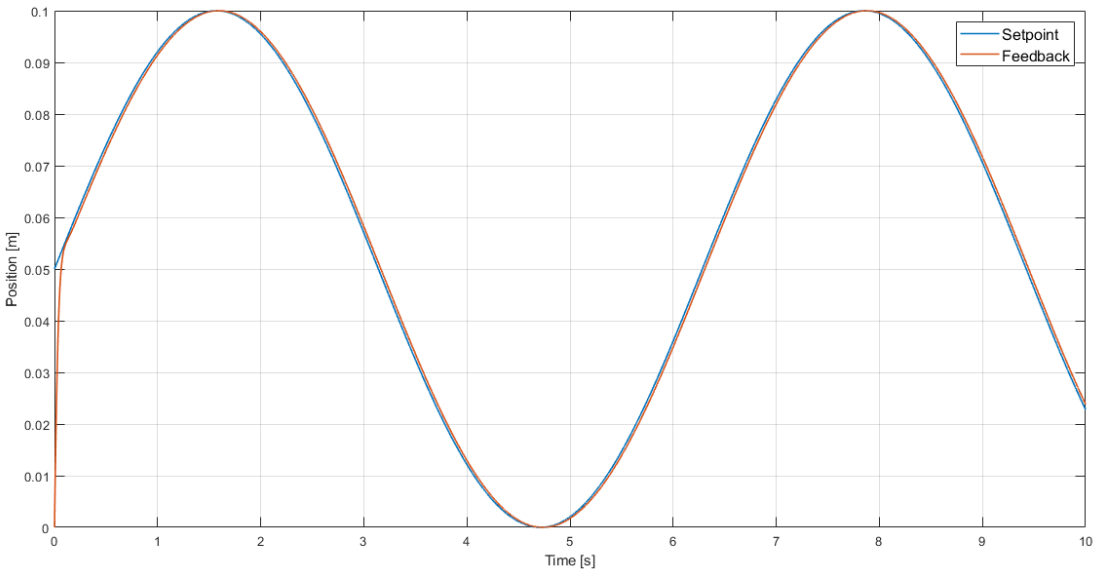


Figure 5.40: Simulink - LTI block

At this point, the response of the system is used as feedback. So, the PID tuning is performed on the new Simulink file.

Once the parameters are update, the system response obtained are:





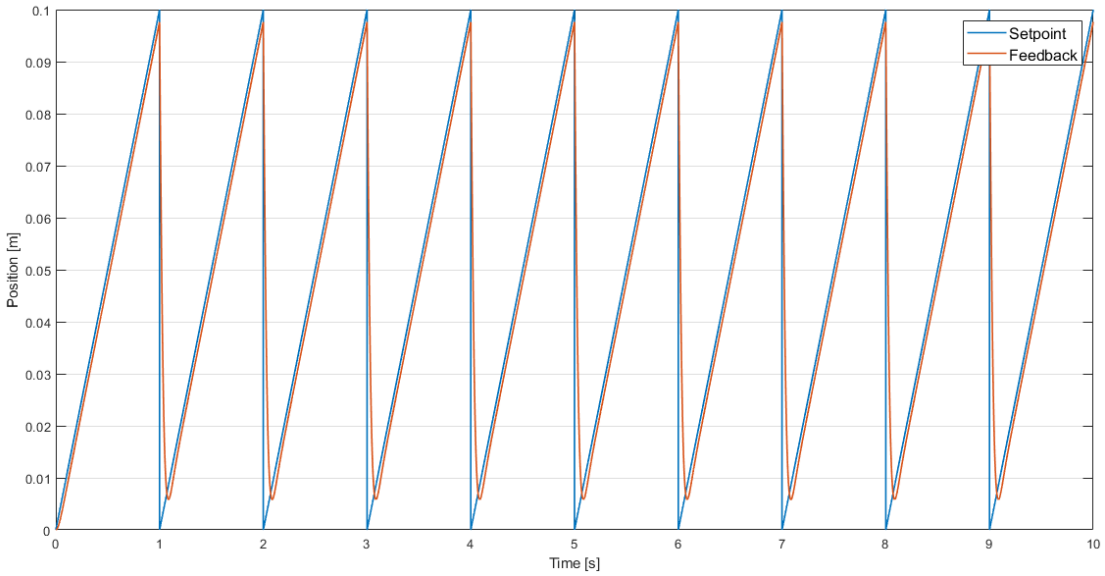


Figure 5.41: Dynamic response - LTI block

All these different responses are obtained with the following parameters:

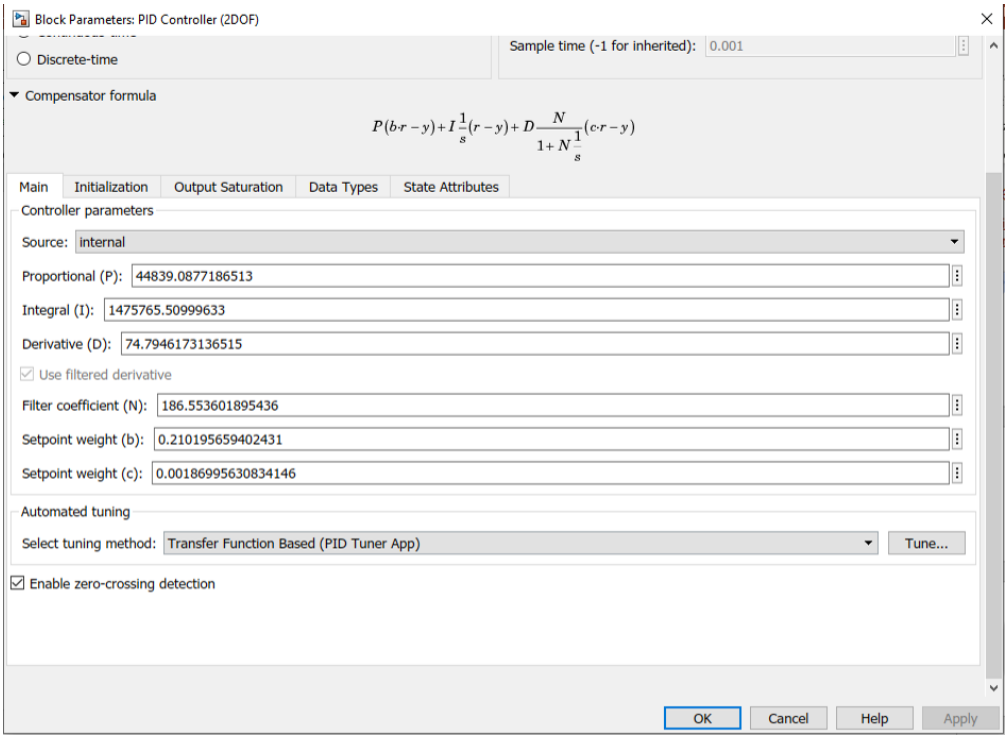


Figure 5.42: Updated parameters - LTI block

In conclusion, in both cases, using the autotunig Toolbox, all the obtained responses are better than the Pure Proportional PID behaviours.

Chapter 6

Panel design and protection circuit

6.1 Panel design

The bank is designed in order to have a re-useful configuration for all the experiments which are performed.
The HMI panel and the PLC S7-1200 are positioned in the left side of the bank. While, pistons are located in the right side and the valves are in the center of the panel. Also, the electrical connections are in the middle low side of the bank, in order to have a easier system reconfiguration.

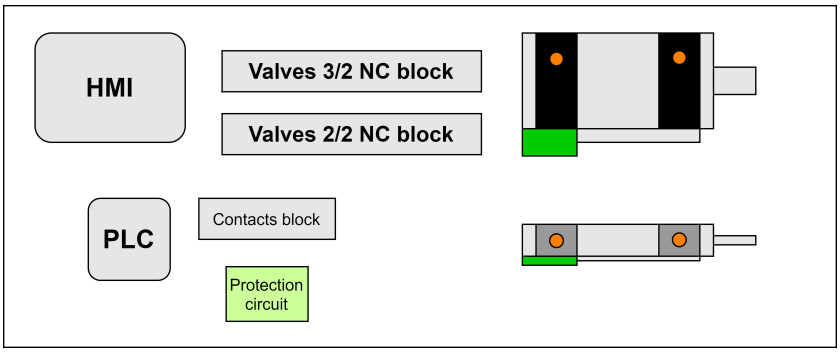


Figure 6.1: Designed bank

The real designed panel is:

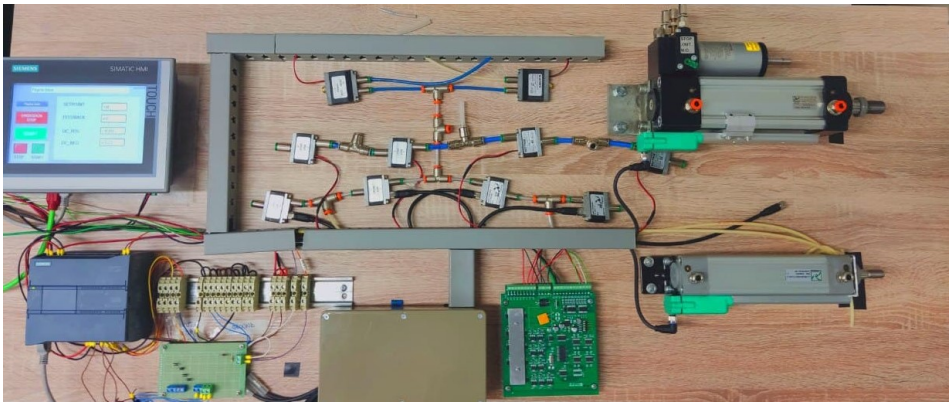


Figure 6.2: Designed real bank

In particular, in the following section is shown an electrical and pneumatic configuration, with the use of the valves 2/2:

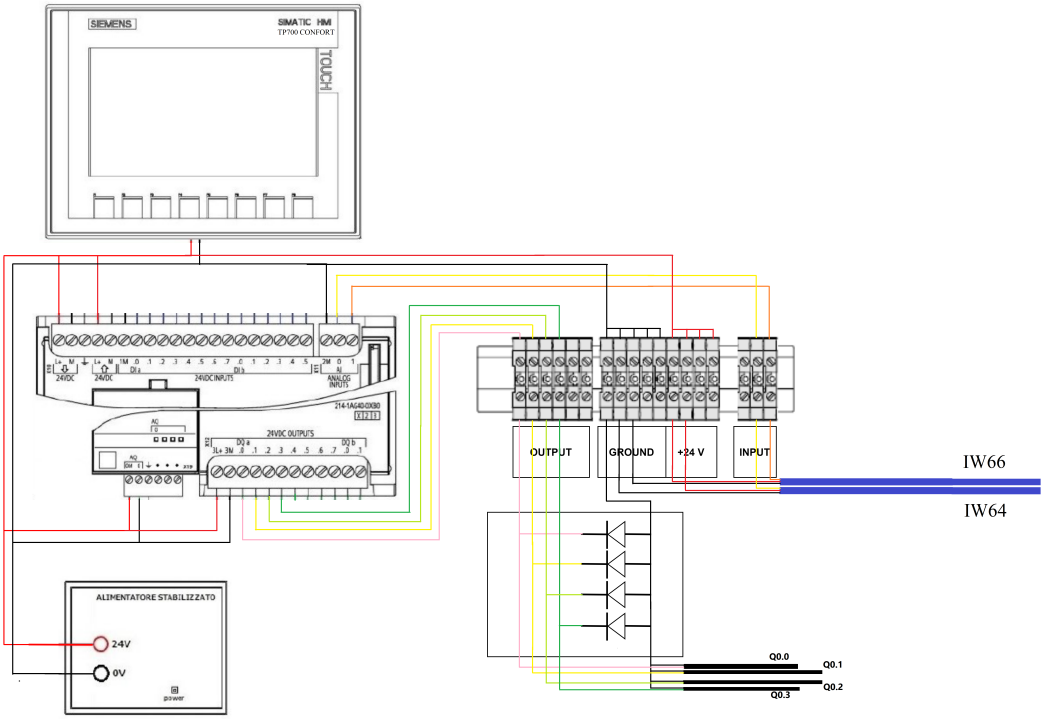


Figure 6.3: Electrical configuration

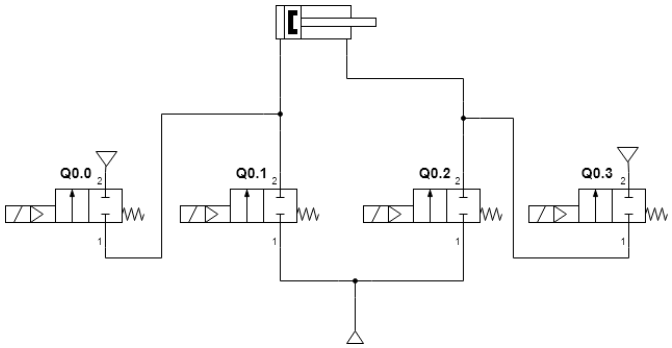


Figure 6.4: Pneumatic configuration

As shown in figure, the valves are called in such a way it is possible to recognize them. In particular, in the following table all the used input and output ports on the TIA portal are listed:

ADDRESS	COMPONETS	IN/OUT
IW64	**Trasducer [V]**	**INPUT**
IW66	**Trasducer [A]**	**INPUT**
Q0.0	**Exhaust outstroke**	**OUTPUT**
Q0.1	**Outstroke**	**OUTPUT**
Q0.2	**Instroke**	**OUTPUT**
Q0.3	**Exhaust instroke**	**OUTPUT**

Table 6.1: Address IN/OUT of the ladder diagram of the PLC

The HMI panel is used because it is useful to make the system understandable and more friendly.
For all proportional experiments, in the HMI panel, the same page is used, as shown in the fig.6.5:

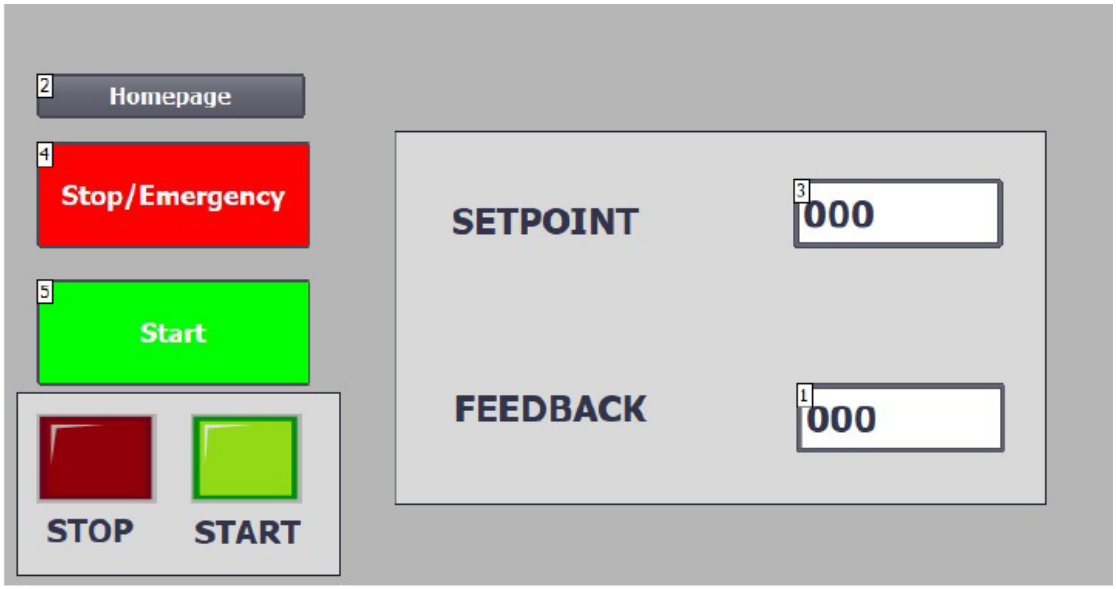


Figure 6.5: HMI page

As shown in the fig. 6.5, the buttons used are:

- Start [5]: in order to start the system
- Stop/Emergency [4]: in order to block the system in case of emergency
- Home [2]: in order to select the homepage

Also, there are two In/Out displays, [1] and [3], used to see the response of the system and to configure it. Moreover, there are two LEDs that corresponds to the state of the system.

As regards the used valves, they are Pneumax N/C solenoid valves with high working frequency. These valves work at 24 VDC and offers an absence of internal friction. In terms of response time they work in a milliseconds range, while their operating life is over 500 million cycles. These valves have many advantages:

- Compact dimension
- Short response times
- Precision and flexibility
- High flow rate

In addition, there is the presence of mounting a control with speed up. Thanks to this, it is possible to have excellent performance. In particular, with a valve opening less then 5ms and in closing of 2ms. Another important characteristic is the piloting through the PWM, PCM and PFM techniques in order to obtain a proportional flow management. Also, their main applicative sectors of these valves are:

- Positioning systems
- Pilot systems
- Pressure and flow rate control device
- Industrial automation

Moreover, the used Pneumax solenoid valves N/C 3/2 have the same behaviour of the 2/2 valves. By the table below (6.2), it is possible to observe the main dynamics response characteristic:

Series	**Rate of flow [Nl/min]**	**Function**	**Max Frequency**	**Response time [msec]**	**Connection**	**Use**
MA340	100-170	2/2 NC	500Hz	<5 KK<1	M8 3pin	Fitting d.4mm Fitting d.6mm
MA360	50-100	3/2 NC - NO	300Hz	<7 JJ<5 KK<2	M8 3pin	Fitting d.4mm Fitting d.6mm ON BASE

Table 6.2: Datasheet of HF Pneumax valves

For what concern the used cylinders, they are two: a pneumatic cylinder and a oil-pneumatic cylinder. The first one is a Pneumax cylinder, with a 100mm stroke and a 32mm diameter.
The second used Pneumax cylinder with a 100mm stroke and a 50mm diameter, is coupled with a hydraulic speed regulator. This regulator allows to uniform and control the piston's velocity in a precision way. On one hand, the pneumatic cylinder have the problem of speed control due to the air filling the chamber. Thanks to the use of oil, in this cylinder, holds a constant speed, forcing the oil to pass from the front to the rear chamber (and vice versa) through an obstruction.

The main characteristics of the used transducer are:

- Strokes from 50 to 900mm
- Orientation detection of the magnet inside the cylinder
- Direct analog output for displacement

- Working temperature: 0...+50°C
- IP65 protection
- Power supply 24 VDC ±0%

The difference between the two used transducers are that one drive in voltage (pneumatic cylinder), the other in current (oil-pneumatic cylinder). These transducers allows to obtain an essential modular structure with compact size for simple installation. Below are shown the datasheets of transducers:

Output signal	**0,5...9,5 V (N)**	**4,8...19,2 mA (E)**
Electrical zero	0,5...0,8 V	4,8...5,3 mA
Span	9 Vdc ± 100 mV max	14,4 ± 0,2 mA
Nominal power supply	24 Vdc ±20%	24 Vdc ±20%
Max. power ripple	1 Vpp	1 Vpp
Output current consumption	35 mA	60 mA
Output load	≥10 KΩ	50...500 Ω
Max. output value	12 V	35 mA
Alarm output value	10.5 V	21 mA
Electrical isolation	50 V	50 V
Prot. against polarity inversion	Yes	Yes
Prot. against overvoltage	Yes	Yes
Prot. against power supply in output	Yes	Yes

Figure 6.6: Transducer's datasheets

The first column is referred to the transducer drive in voltage and the second one to the transducer drive in current. In order to use the second cylinder, since the analog input of the S7-1200 PLC accepts only positive voltage, the current circuit designed as follows:

$$R = \frac{V}{I} = \frac{10V}{19.2mA} \simeq 520\Omega \tag{6.1}$$

This voltage is imposed in order to create the same behaviour for the two transducers.

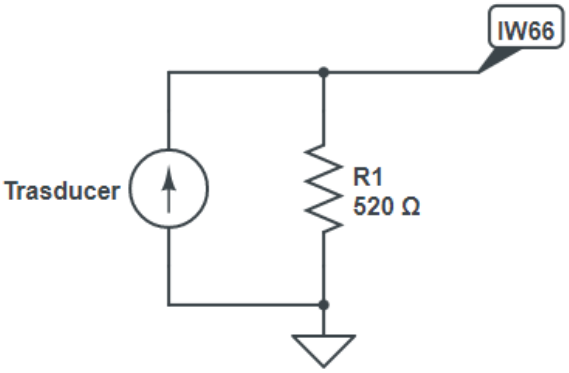


Figure 6.7: Current divider ideal circuit

Using the E12 series, this value is not present, so the sizing of the resistance is:

$$R \simeq 520\Omega = 47\Omega + 470\Omega \tag{6.2}$$

For this reason the final obtained circuit is:

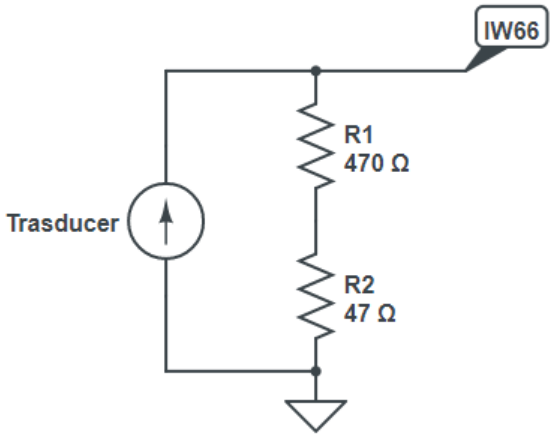


Figure 6.8: Current divider real circuit

After the design of the circuit, in order to have a more compact and a reusable configuration, a single board with this circuit and the diode protection circuit is created.

6.2 Protection circuit

In order to protect the PLC's output ports, when the output are deactivated (from +24V to 0V), the valves's coils returns a stored energy, with the opposite direction. This leads in a negative voltage on the PLC's ports. To prevent this negative voltage from damaging the output of the PLC's ports, a protection circuit is designed using four diodes, for each used port. These diodes are positioned in shunt with the valves's inductors. In this way, the diodes discharge toward ground the negative voltage and the negative pick of the voltage is avoid.

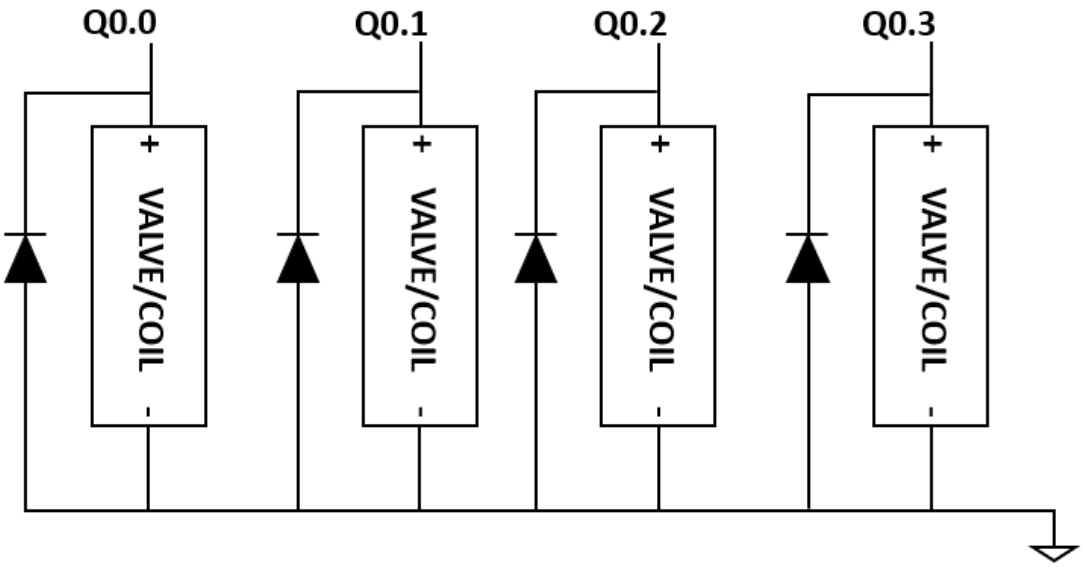


Figure 6.9: Protection circuit design

In order to make the experiment, the 2/2 valves are used, and the comparison with the made experiments in chapter 2 are done.

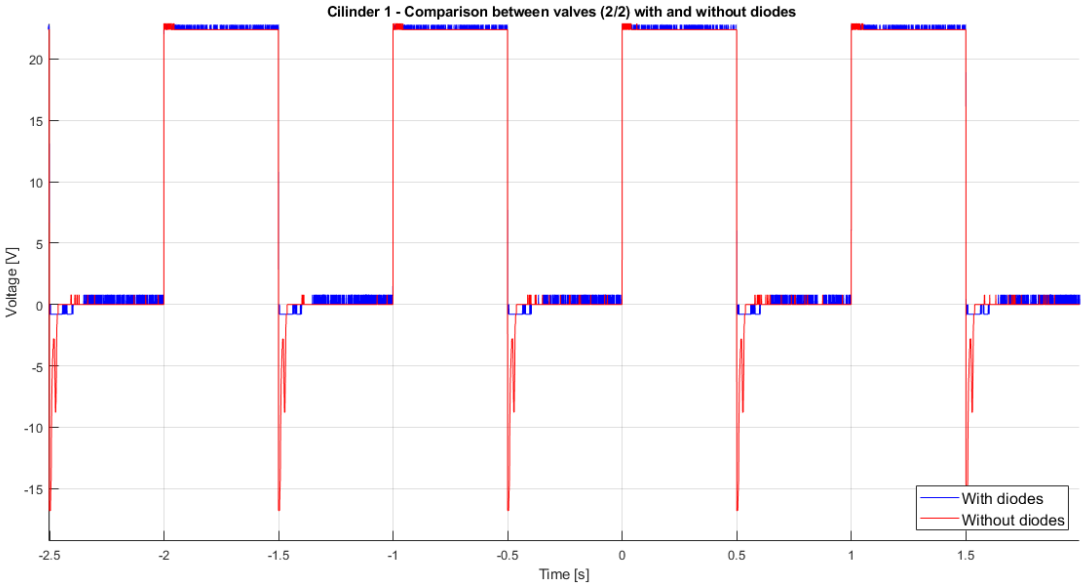


Figure 6.10: Comparison with and without diodes

By the fig.6.10 it is possible to note that the blue line is the behaviour of the circuit with diodes, while the red line is without diodes. This measure is done by the oscilloscope (fig.1.5), in particular its probe is positioned on one diode's cathode.

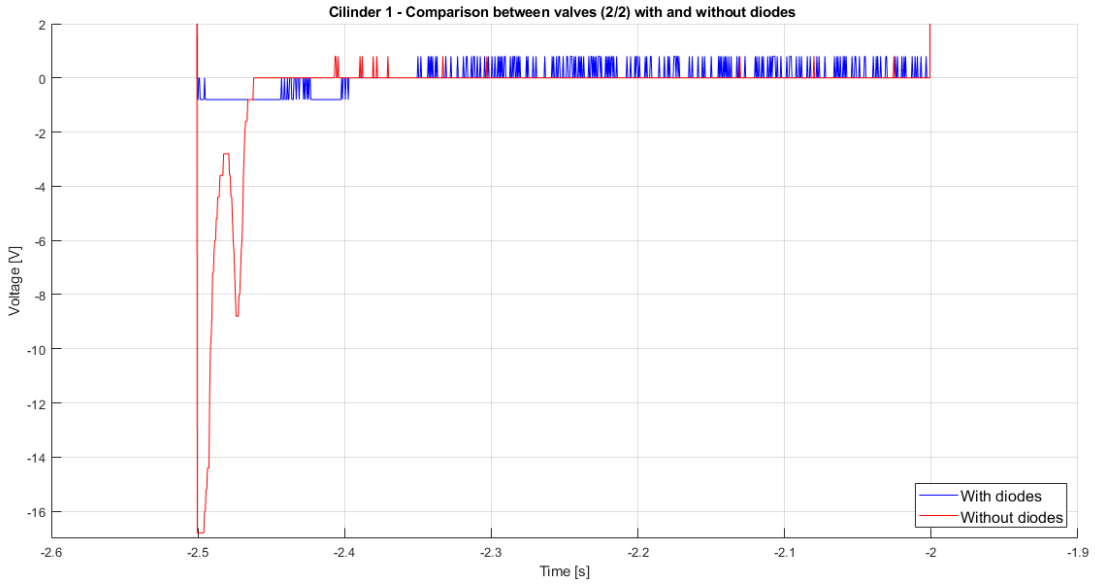


Figure 6.11: Zoom of the comparison with and without diodes

Moreover, is possible to observe that diodes avoid a negative voltage equal to 18V, returning the voltage to a maximum negative value of -1V. Also, for a correct behaviour of the algorithm the cycle time is modified from 100ms to 20ms. This because the designed project with the introduction of the protection circuit point out instability problems. So, on the TIA portal the ports are changed:

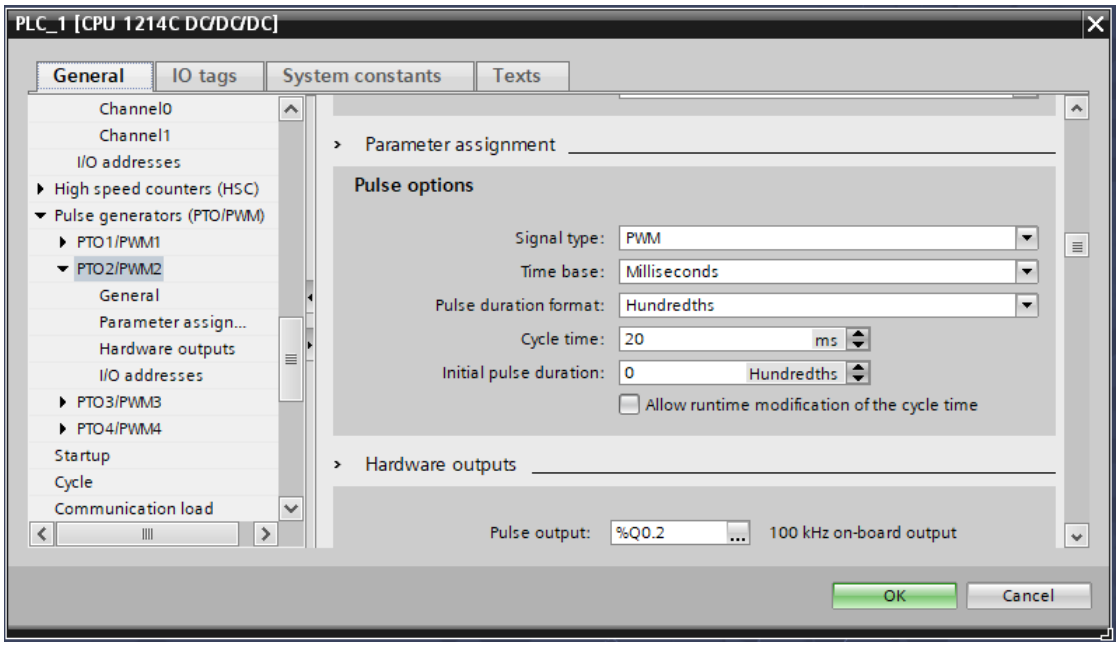


Figure 6.12: PTO2/PWM2 Cycle time: 20ms

To prove that, in the following fig.6.13, the attitude of the algorithms with and without diodes are measured.

In particular, it is possible to see, that the settling time and the overshoot are better on the without diodes case. This phenomena could be assigned to the absence of negative voltage, which led to better valves closing performance. So, the rise time not change between these experiments.

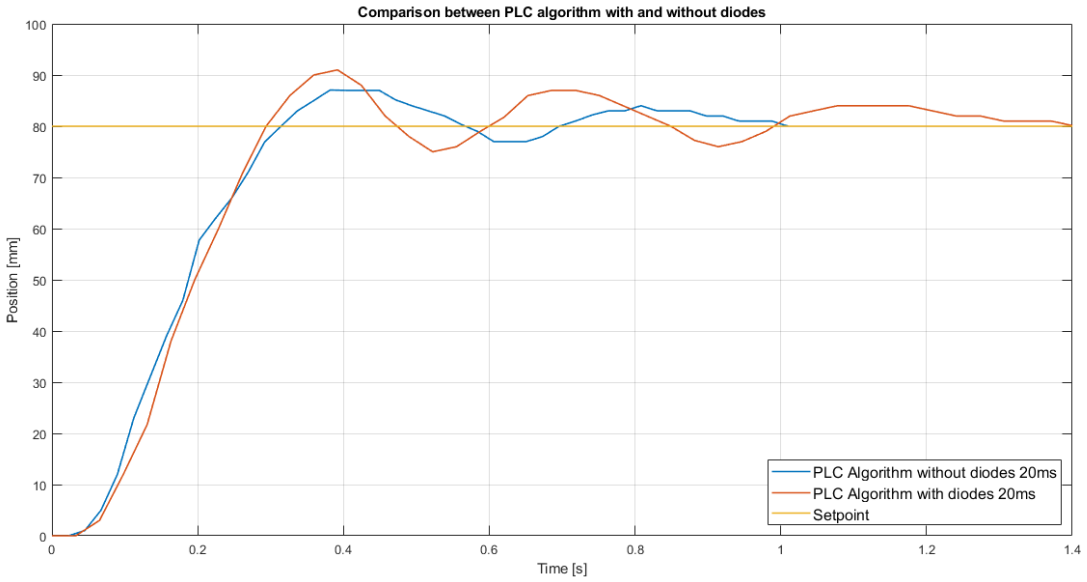


Figure 6.13: Comparison at 20 ms with and without diodes

Chapter 7

Dynamic Response in load condition

In this section are studied system responses with applied loads, in order to know the response of system and understand the influence of the loads on the cylinders. All the chapters above treated experiments in no load conditions, so in order to have a more general conditions for experiments, six different floating loads are designed with cylindrical shapes.
The designed loads are obtained by the calculation of the dimension with three constant parameters: mass, iron density and cylinder's diameter.
In particular are chosen masses equal to 1kg , 2kg and 5kg and iron density equal to 7860 kg/m^3. As regards the diameters of the cylinder, based on the used piston, the dimension of diameters are shown in the table below:

Cylinder	Mass [Kg]	Diameter [mm]	Length [mm]
Pneumatic Cylinder	1	34	140
	2	34	280
	5	80	126
Oil-pneumatic Cylinder	1	52	59
	2	52	119
	5	80	126,8

Table 7.1: Floating Loads Dimensions

The experiments are made with two kinds of controllers, applied on one hand to the pneumatic cylinder and on the other hand to the oil-pneumatic cylinder. In particular, the used pneumatic configuration are made with 2/2 valves and the same algorithms used in the chapters 2 and 3.

7.1 Pneumatic cylinder - Base control

For what concerned the pneumatic cylinder with a base control, it is possible to see that the rise times are very similar for the three experiments.
Immediately, is possible to note that the settling time with a 5 kg load is bigger that the others. This proved that a greater inertia causes a slowdown of the system. Also, the inversion of the stroke is faster as inertia decreases.

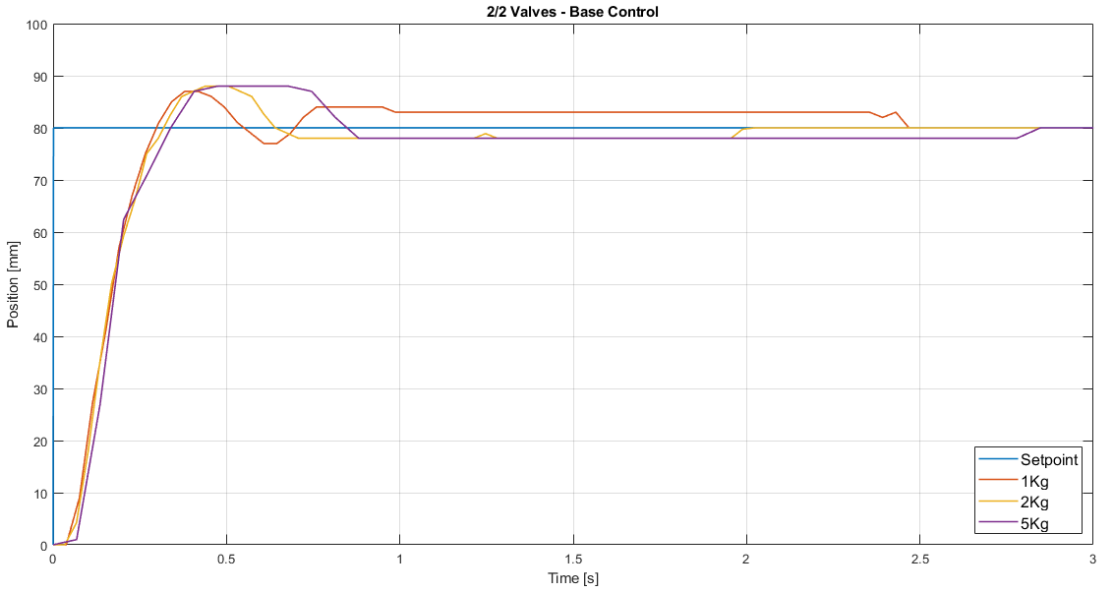


Figure 7.1: Pneumatic cylinder - Base control - Applied Loads

7.2 Pneumatic cylinder - Forced DC control

On the other hand using a forced DC, the obtained responses are:

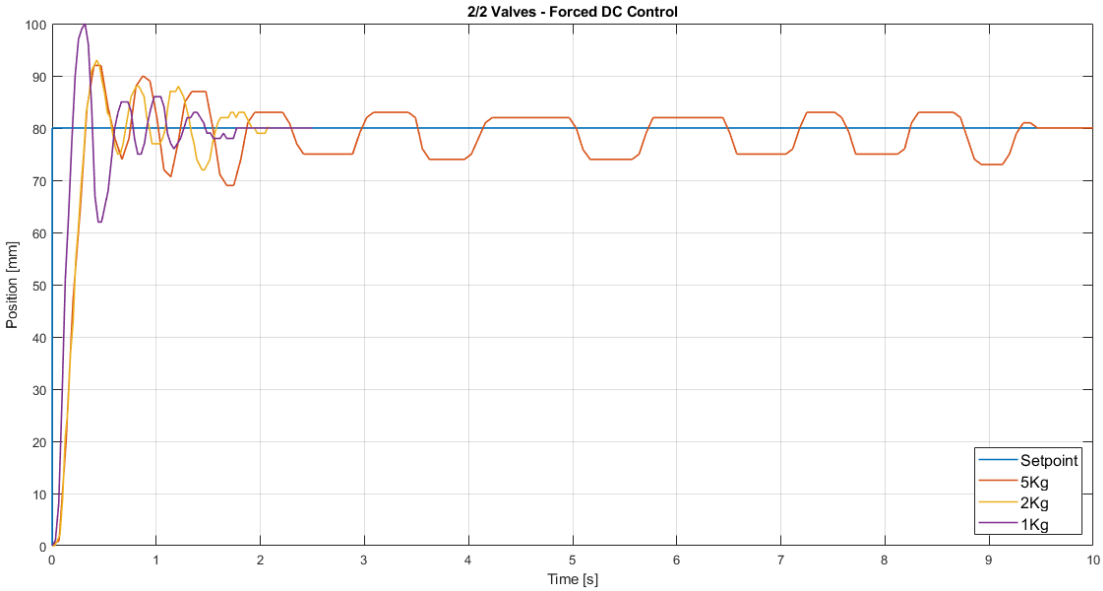


Figure 7.2: Pneumatic cylinder - Forced DC control - Applied Loads

By the figure is possible to observe that with 1 kg load the overshoot obtained is bigger than the 2kg and 5 kg loads experiments. This is due to the greater achieved velocity which the cylinder have with only 1 kg load.
Instead, as regards the settling times the experiment made with a 5 kg load carry out some oscillations around the *Setpoint*, they last more or less 10 seconds. At the end, they achieved definitely the *Setpoint*. The oscillations increase with increasing load.

7.3 Oil-pneumatic cylinder - Base control

For the oil-pneumatic cylinder, the same experiments are made. In this section is analyzed the base control algorithm and its responses.
All the loads, applied on the piston, returns a similar dynamic responses.
The overshoots and the rise times are comparable with each other, but regards the settling time, it is possible to note that the best response is obtained with the minimum load, as expected. Also, the strokes and the slopes are similar since the damper presents.

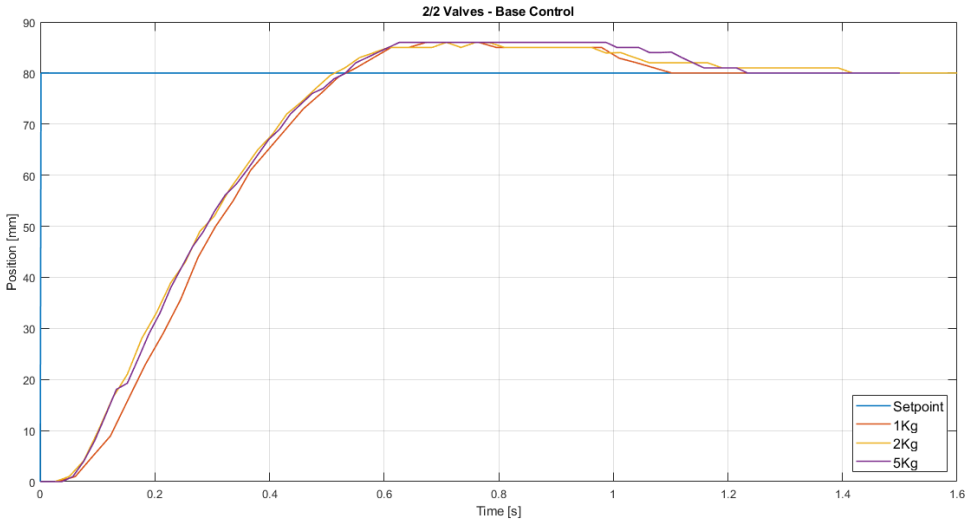


Figure 7.3: Oil-pneumatic cylinder - Base control - Applied Loads

7.4 Oil-pneumatic cylinder - Forced DC control

The last experiments are made with the oil-pneumatic cylinder with a forced DC control. By the figure below, it is possible to see that the rise times changed based on the applied load. By the increasing of the load , the rise time is lower. As the previous experiments the overshoots are similar , and settling times too.

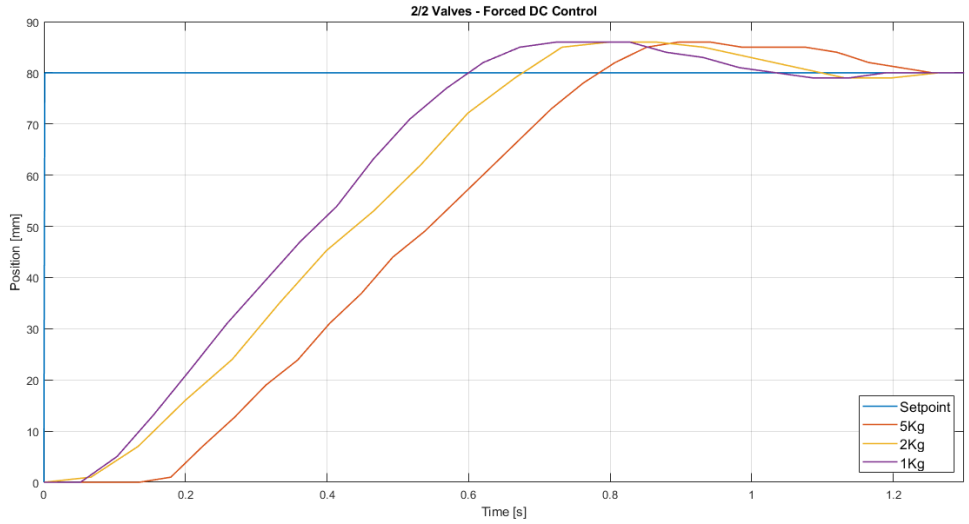


Figure 7.4: Oil-pneumatic cylinder - Forced DC control - Applied Loads

7.5 Simscape - In load condition

Thanks the use of Simscape, it is possible to compare the data obtained from the experimental test using base control algorithm on the real and the simulated system. The used configuration is the same used on the real bank. By using the 2/2 valves configuration and the parameters of pneumatic cylinder, the comparison is performed between the following figure and the figure 7.1.

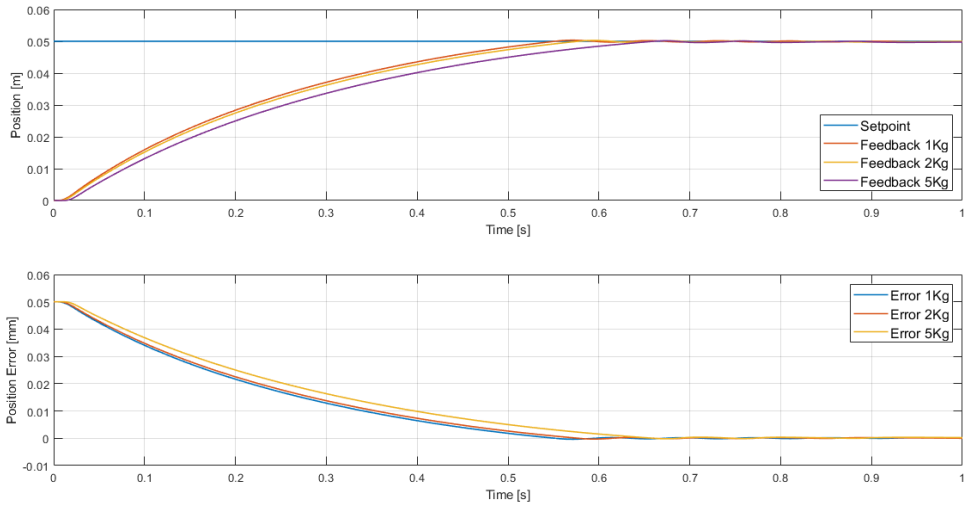


Figure 7.5: pneumatic cylinder - Base control - Applied Loads on Simscape

This figure shows as the real experiments, that the rise time increase, with the increasing of the load weight and inertia. More or less the order of magnitude of the rise time is the same between real and simulated systems. The three responses, shown many undershoot and overshoot, but in simulation have lower amplitude with respect to the real experiments.

Moreover, another difference obtained with respect to the simulation in no-load condition is that the responses do not achieved perfectly the *Setpoint*, but there is a small error in steady state condition around 0.4%.

So, in conclusion it is possible to highlights the main behaviours which are, in particular, the rise time and the settling time, which increased as load weight. This is due to the bigger inertia that slow down the system.

Chapter 8

Driver speed up

8.1 Use and main characteristics

In this section, is analyzed the behaviour using a Driver Speed up. The choose of this driver is made in order to improve the dynamic response of all controllers. This driver use a particular strategy, in order to speed up the reaction of valves with respect to the command. In particular, the used driver speed up is the Matrix P/N HSDB 990.012. That driver is a multi-channel device with 9 different I/O ports.

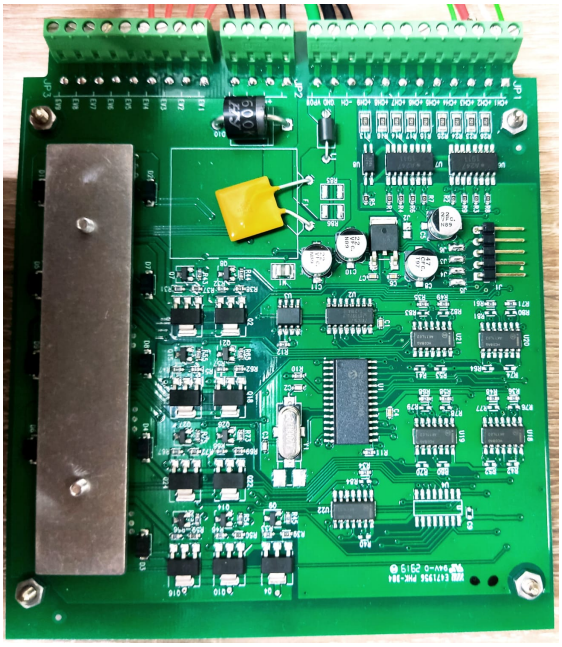


Figure 8.1: Driver speed up

Using the oscilloscope the behaviour characteristic of driver's input/output is shown in the fig. 8.2. When the driver's input is ON the driver's output continuously change between 27V and -2V, so the delta is around to 30V. With this characteristic, the driver provides to maintenance the efficiency of the transient point of view.

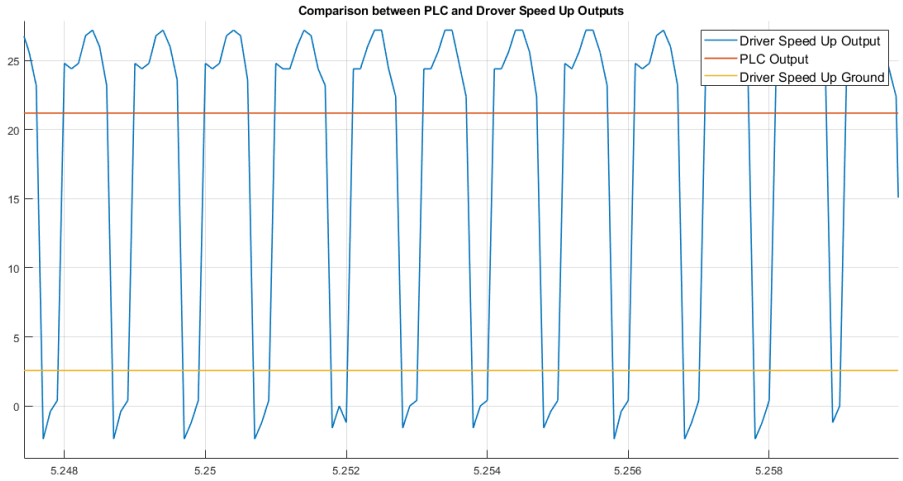


Figure 8.2: Transient dynamic response

Moreover, when the driver's input is deactivated, the output port has a delta of 60V. This behaviour proved the right work condition of driver, because provides a high voltage to increase the valves opening and closing speed.

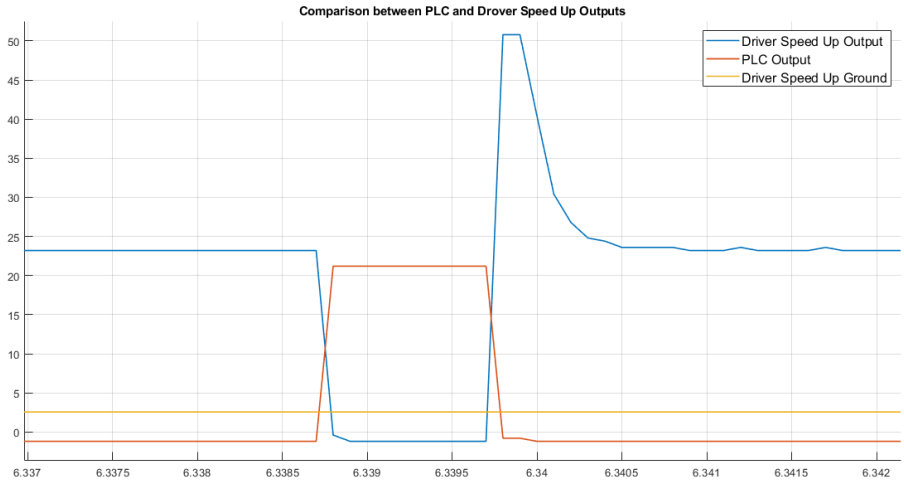


Figure 8.3: Driver's dynamic response

In the section below, the comparison between 24V 2/2 valves and KK 2/2 valves are made in four different conditions using same algorithms used on chapters 2 and 3.

8.2 Comparison between 24V and KK valves

In particular, all experiments are made with two different models of valves: MA3411F0C22400 (24V) and OX821103C2KK (KK). The used reference conditions are the same for each experiment, 80mm for the *Setpoint* and 5 bar for the air supply.

8.2.1 Pneumatic cylinder using base control

Using 2/2 valves with base control the obtained dynamic responses are:

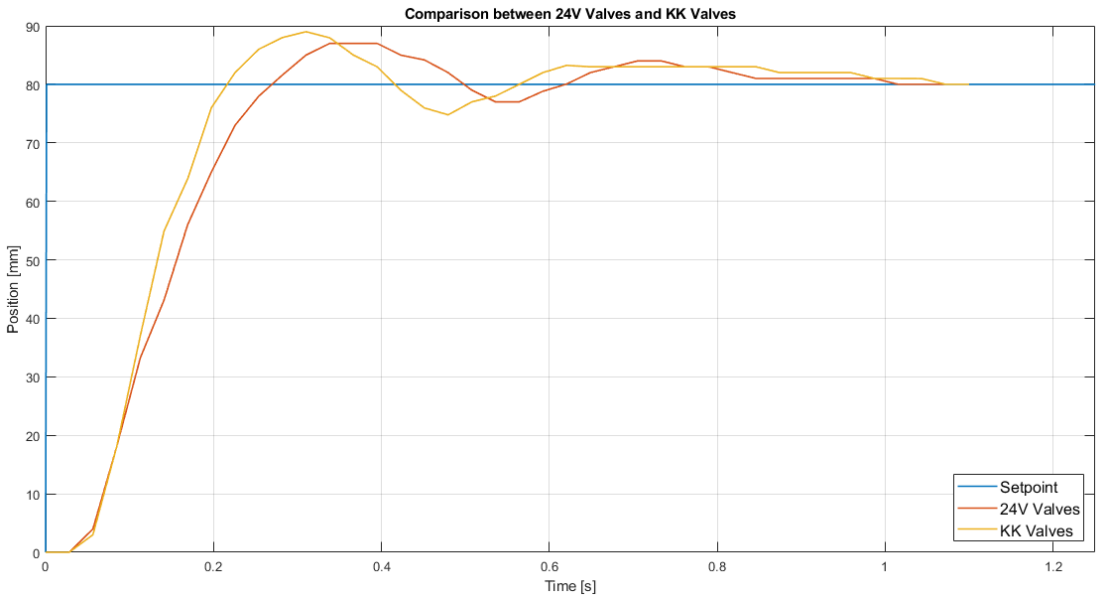


Figure 8.4: Comparison between 24V and KK valves with pneumatic cylinder - Base control

It is possible to analyze this behaviour and note that between 0mm and 30mm is obtained the same response, but between 30mm and 80mm the KK slope response is greater than the slope using the 24V valves. The overshoot of the KK valves configuration is slightly higher with respect to the 24V configuration. Meanwhile, the settling times are more or less the same, in both cases.

Algorithm	Overshoot %	Rise time [s]	Settling Time [s]
24V Valves	8,75	0,269	1,015
KK Valves	11,25	0,216	1,071

Table 8.1: Comparison between 24V and KK valves - Base control - pneumatic cylinder

8.2.2 Pneumatic cylinder using forced DC control

Using 2/2 valves with forced DC control the obtained dynamic responses are:

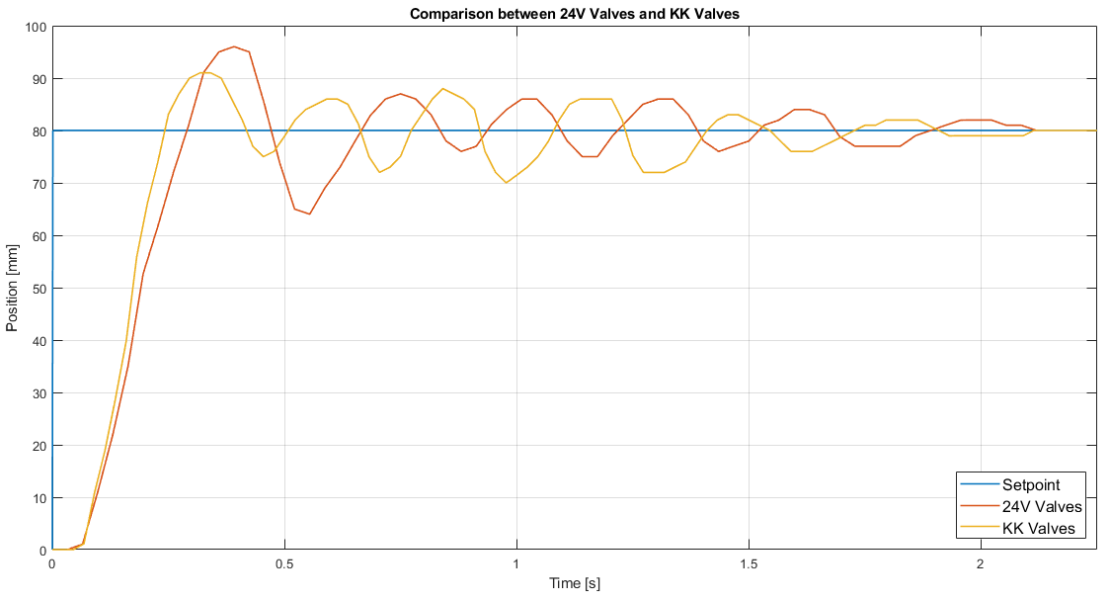


Figure 8.5: Comparison between 24V and KK valves with pneumatic cylinder - Forced DC control

The overshoot of the KK valves configuration is slightly lower with respect to the 24V configuration. Instead, the settling times are more or less the same and the rise time is better in the KK valves configuration. By the table below, it is possible to note the differences between these different experiments.

Algorithm	Overshoot %	Rise time [s]	Settling Time [s]
24V Valves	20	0,290	2,119
KK Valves	13,75	0,242	2,113

Table 8.2: Comparison between 24V and KK valves - Forced DC control - pneumatic cylinder

8.2.3 Oil-pneumatic cylinder using base control

The same experiments are made with the oil-pneumatic cylinder with the same configurations. A key point in this launch is the constant slope of the characteristic due to the damper of the cylinder, that makes constant the velocity. In particular, in this section, using base control, the experiments are made.
So, using 2/2 valves the dynamic responses are obtained. Analyzing these behaviours, it is possible to note that the rise time is better using KK configuration, due to the slope that is greater with respect to the 24V configuration. Instead, the overshoots and the settling times are more efficient with the configuration without driver speed up. The responses are shown below, both with the numerical results.

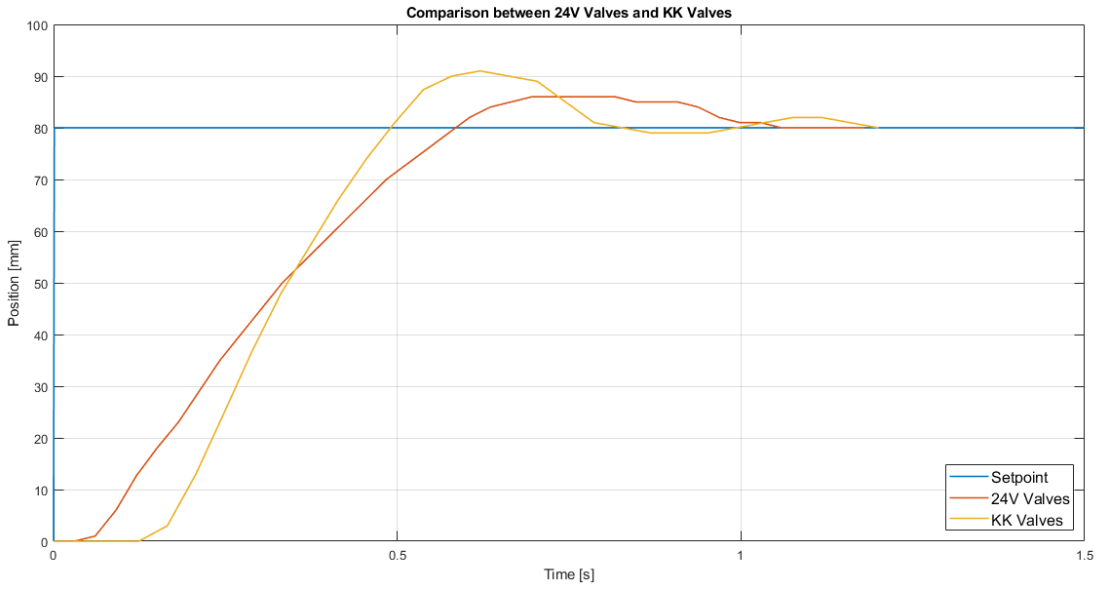


Figure 8.6: Comparison between 24V and KK valves with oil-pneumatic cylinder - Base control

Algorithm	Overshoot %	Rise time [s]	Settling Time [s]
24V Valves	7,5	0,585	1,06
KK Valves	13,75	0,490	1,2

Table 8.3: Comparison between 24V and KK valves - Base control - oil-pneumatic cylinder

8.2.4 Oil-pneumatic cylinder using forced DC control

Now, using the forced DC control the obtained results are:

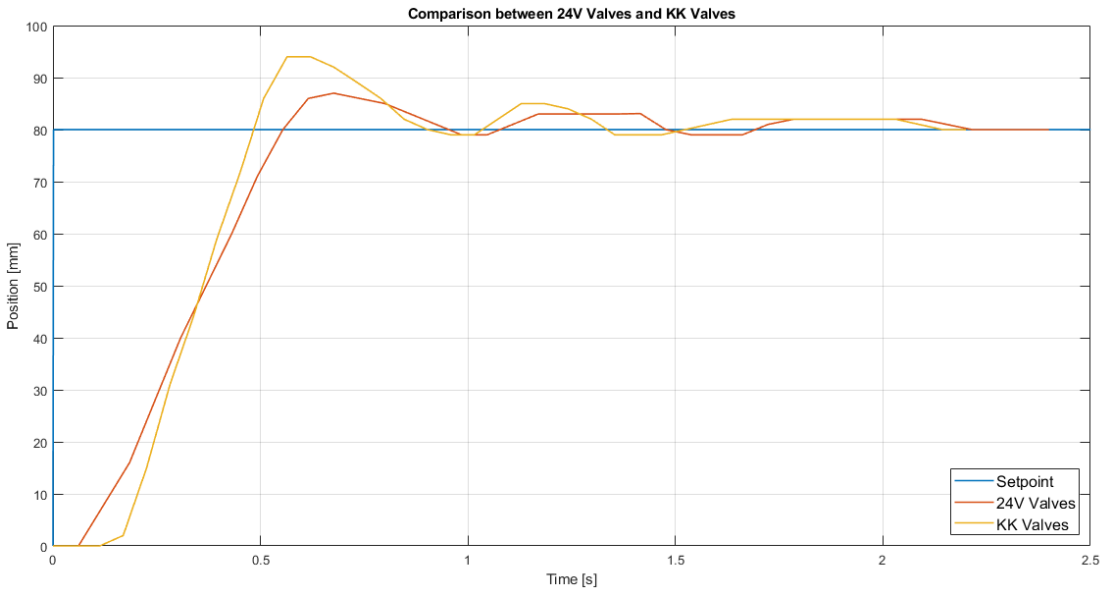


Figure 8.7: Comparison between 24V and KK valves with oil-pneumatic cylinder - Forced DC control

By the figure it is possible to see that with this algorithm the slope remains greater using KK configuration, but this provides a worst overshoot with respect to 24V configuration. Instead, the rise time are comparable.

Algorithm	Overshoot %	Rise time [s]	Settling Time [s]
24V Valves	8,75	0,553	2,2
KK Valves	17,5	0,472	2,143

Table 8.4: Comparison between 24V and KK valves - Forced DC control - oil-pneumatic cylinder

In conclusion, based on the design choices it is possible to obtained a better rise time response using the driver speed up with KK valves, instead if it is needed to have a threshold between overshoot and rise time the better configuration is the 24V valves configuration.

Chapter 9

Conclusion and Future applications

In this section, the developments managed and future applications are analyzed.
In the panel used for the different experiments, there are two possible pneumatic configurations and two different actuators.

It's possible to reconfigure the system to have different experiences. This is thanks to the use of a terminal board and also to the use of the HMI panel, which makes the system user-friendly and independent from the PC. Since once the project has been loaded, the system is fully controllable via HMI.

As regards for the behaviors studied, good results were reported for each type of control.
In particular, as mentioned in the previous chapters, proportional controls, PID controls and controls in a simulated environment were analyzed.

Through the fig. 9.1 it is possible to see all the experiments performed.
In particular, in some experiments they show better performance with respect to overshoot, rise time and settling time.

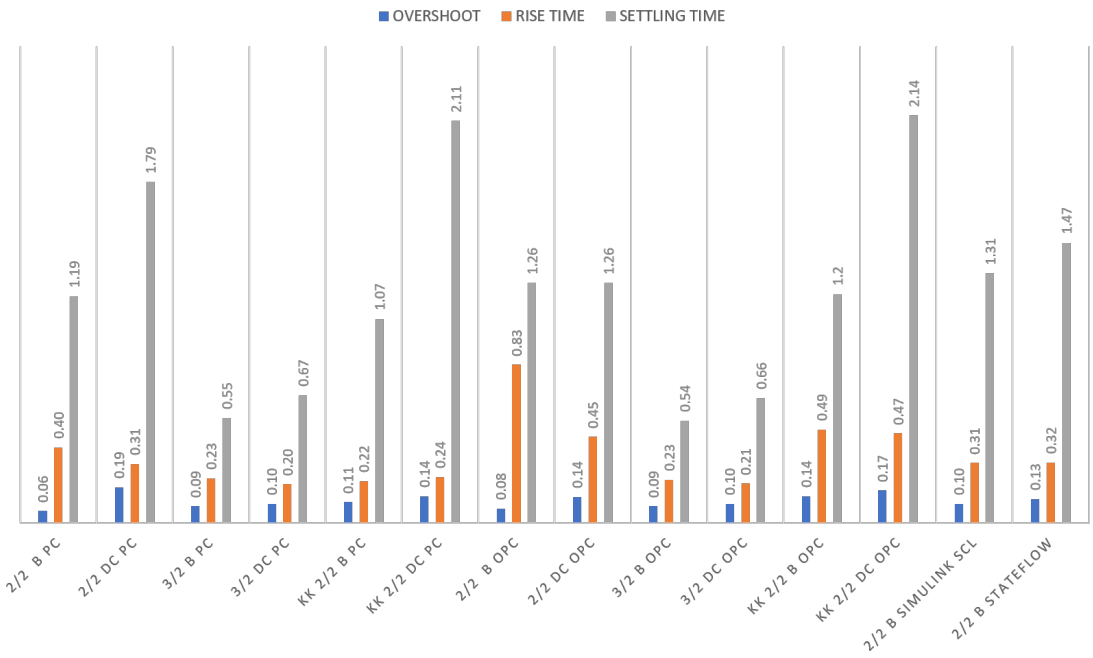


Figure 9.1: Final results

Based on the figure it is possible to note that the histogram presents all the configurations using 24V 2/2 valve, 3/2 valves and KK too. In the left side all the experiments made with the pneumatic cylinder (PC), using base (B) and forced DC (DC) controls. On the center, are appreciable the experiments made with the oil-pneumatic cylinder (OPC) and at the end, in the right side the last two experiments are made using Simulink environment.

For each experience there are many points of view, based on the final request of work. As for the pneumatic cylinder, if the priority is to have a faster response time, then the best choice could be the use of 3/2 valves. On the other hand, if it is not possible to have valves of this type, the best choice would be to use 2/2 valves with base algorithm. However, both cases have brought better results with the use of 2/2 KK valves and 3/2 valves using forced DC control.

For what concerned the oil-pneumatic cylinder, if you want to have a faster response time, the best choice is to use 3/2 valves with forced DC. Instead, as for the use of 2/2 valves, these are not recommended, as their performance are worse both in terms of rise time and overshoot.

Possible future developments are:

- The use of valves controlled by speed up drivers, using KK 3/2 valves, obtaining a comparison with respect the use of a simply configurations 24V valves.
- The increase of pneumatic lines's diameters.
- The use of precision sensors that allows more accurate measurements.
- The use of air supply pressure equal or not greater than the maximum operating pressure of valves.

Appendix A

How to create a HMI panel

In order to create a new page on the HMI panel the following section is created.

In this project, the used HMI is the TP-700 model. In particular this model needed the add-on package WinCC_Professional. The procedure in the starting phase is standard, while the structure of the page , in the second phase, the objects used can change with respect to the used. The example of an HMI page is shown below.
In order to add an HMI device on the TIA portal, the following step are required:

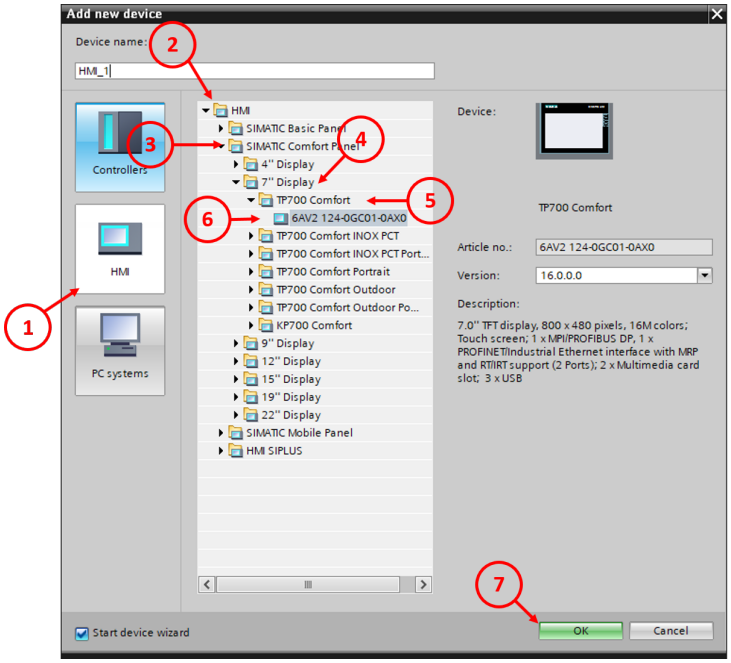


Figure A.1: Instruction in order to add HMI panel

As shown in the figure A.1, the command used are, respectively: left click to HMI, choose the model, and select the own firmware release.
So, a window appears (fig.A.2)and the tool offers the possibility to connect the HMI and create pages in a systematic way. Otherwise, if you want to perform a manual procedure, just left click on Finish icon. The following instruction are required.

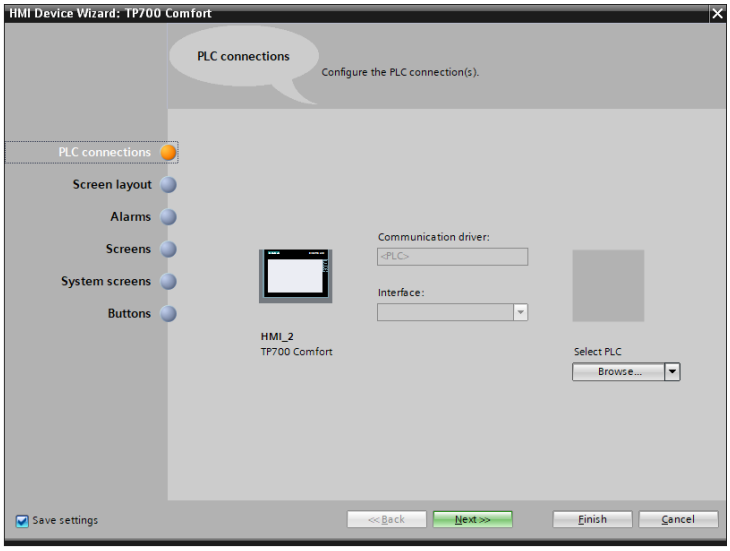


Figure A.2: Automatic configuration

After these simply configurations, the device is presented on the TIA portal. To connect PLC and HMI, by drag and drop, it is possible to obtain the desired connection, as shown in the figure A.3

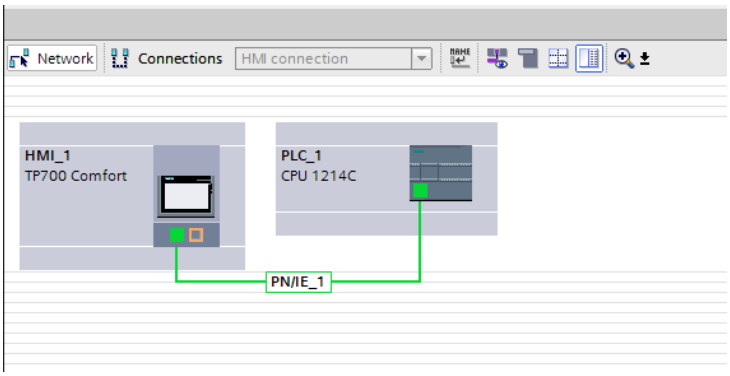


Figure A.3: Connection between PLC and HMI panel

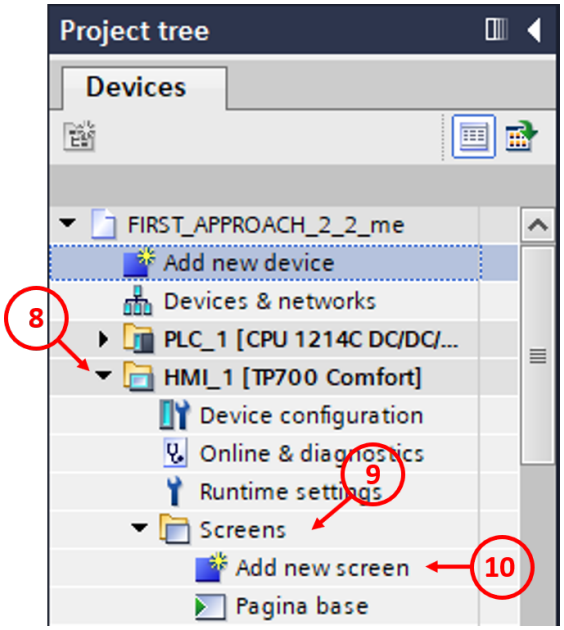


Figure A.4: Add new page in the HMI panel

So, the following settings are made (A.4), in particular in order to have a linear structure, by adding a new screen, is possible to have an entire new page dedicated to own project fig.A.5

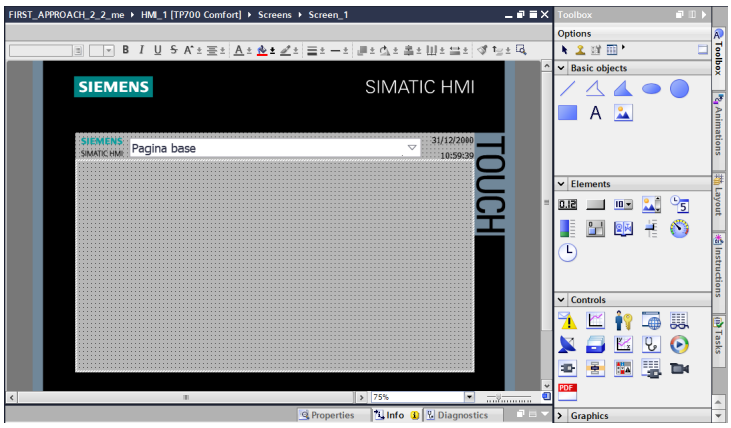


Figure A.5: HMI page

Therefore, after the page creation, in the screen is possible to add some digital objects. In the right side of the page, are present the main used elements. In particular, are presented the basic objects, like rectangular or text, elements, like push buttons, sliders and displays and final controls. While, if these libraries are not

enough, it is possible to have more elements (fig.A.6), by the following procedures:

- 11. Left click on the bottom arrow , until the *Library* section appears.
- 12. Left click on the *Library* section
- 13. Left click on the *Buttons-and-Switches* section
- 14. Left click on Master copies folder, and add the desired block.

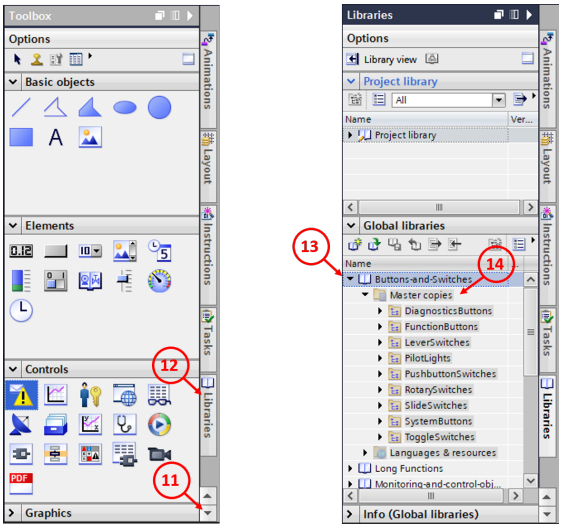


Figure A.6: HMI's Library

Lastly, is shown the procedure to connect a monostable push button with a variable.
By clicking the button it is possible to have two behaviors: the first is *SET BIT* (bit at 1) and *RESET BIT* (bit at zero), the second one is *SET BIT WHILE KEY PRESSED* (bit at 1).

BUTTON	VARIABLE
Hold	TRUE
Release	FALSE

Table A.1: Button's characteristic

In the following figure the passages in order to set bit are shown:

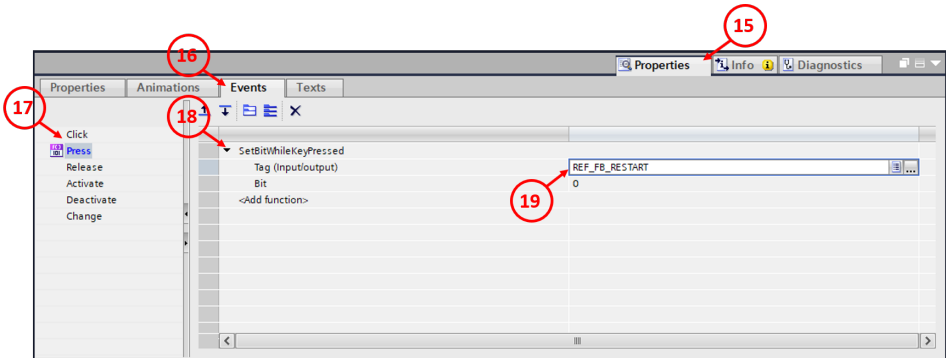


Figure A.7: Set bit

In conclusion, the variable used in the push buttons is set with a acquisition cycle equal to 100ms in order to have a quick refresh. This property is found in the HMI's tags.

Appendix B

Blocks on TIA Portal

Each controllers have an operating system, which responds to each process of the CPU. On TIA portal the main blocks which allows to have a more readable and useful code are:

- OB
- FC
- DB
- FB

The operating system is an integral part of the CPU and is included in the delivery version on the program.
The user program contains all the functions necessary for processing a task specific automation. The tasks of the user program include, for example:

- Verification of the prerequisites for a start with the help of OB startup
- Process data processing, i.e. state-dependent control of output signals
- Reaction to alarms and alarm inputs

Organization blocks (OB) form the interface between the controller's operating system (CPU) and the user program. They are invoked by the operating system and command the following operations:

- Cyclic program processing (i.e. OB1)
- Behavior of the controller at startup
- Handling of errors

A project must contain at least one organization block for cyclical processing of the program. The cyclical processing of the program follows this order:
1. At the beginning of the cyclic program, it is checked whether the individual inputs carry voltage or less. This state of the inputs is saved in the process image of the inputs.
2. The processor then processes the program saved in the organization block. To obtain the necessary input information is accessed to the process image of the inputs read previously and the combinatorial logic results are written in a so-called process image of the outputs.
3. At the end , the process image of the outputs is transferred as a status signal to the output modules and the latter are activated/deactivated. Later processing resumes from point 1.

Functions (FC) are blocks of code without memory. The functions are not provided with data memory in which to save the parameter values of the blocks. For this reason when a function is called all interface parameters must be connected. To have permanent data, global data blocks must be created first. A function contains a program that always runs when the function comes called from another block of code. The functions can be used for the following purposes:

- Mathematical functions
- Technological functions - such as single controls with binary operations

Function blocks (FB) are executed every time a function block is called from a another block of code. A function block can also be called several times at different points into a program. Programming of complex, frequently occurring functions is greatly simplified.
Function blocks are blocks of code that permanently store their own input, output, transit variables and also static variables in instance data blocks.

Data blocks (DB) contain variable data which are used by the user program and allow for user data memory. The structure of the global data blocks can be freely defined.
Global data blocks contain data that can be used by all other blocks. The max. data blocks vary depending on the CPU.

Bibliography

[1] S. Chillari, S. Guccione, and G. Muscato. «An experimental comparison between several pneumatic position control methods». In: *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228)*. Vol. 2. 2001, 1168–1173 vol.2. DOI: 10.1109/CDC.2001.981043.

[2] Osama Olaby, Xavier Brun, Sylvie Sesmat, Tanneguy Redarce, and Eric Bideaux. «Characterization and Modeling of a proportional valve for control synthesis». In: vol. 2005. Dec. 2005. DOI: 10.5739/isfp.2005.771.

[3] Sheng-Chih Hsu and Chi-Ying Lin. «Periodic motion control of a heavy duty pneumatic actuating table using low-cost position sensors and hybrid repetitive control». In: *2013 IEEE International Symposium on Industrial Electronics*. 2013, pp. 1–6. DOI: 10.1109/ISIE.2013.6563590.

[4] Boris Andrievsky, Dmitry V. Kazunin, Daria M. Kostygova, Nikolay V. Kuznetsov, Gennady A. Leonov, Pavel Lobanov, and Andrey A. Volkov. «Control of pneumatically actuated 6-DOF Stewart platform for driving simulator». In: *2014 19th International Conference on Methods and Models in Automation and Robotics (MMAR)*. 2014, pp. 663–668. DOI: 10.1109/MMAR.2014.6957433.

[5] Jun Wu, Jian Huang, Yongji Wang, and Kexin Xing. «Nonlinear Disturbance Observer-Based Dynamic Surface Control for Trajectory Tracking of Pneumatic Muscle System». In: *IEEE Transactions on Control Systems Technology* 22.2 (2014), pp. 440–455. DOI: 10.1109/TCST.2013.2262074.

[6] M. Heidari and H. Homaei. «Improving the pneumatic control valve performance using a PID controller». In: *Turkish Journal of Engineering and Environmental Sciences* 38 (2014), pp. 240–247.

[7] *Documentazione per corsisti/formatori*. URL: https://www.automation.siemens.com/sce-static/learning-training-documents/tia-portal/basics-programming-s7-1200/sce-031-100-fc-programming-s7-1200-r1709-it.pdf.

[8] K.U. Fil, V.N. Iliukhin, and P.I. Greshniakov. «Digital Control System of Gas Pressure Regulator». In: *2020 International Conference on Dynamics and Vibroacoustics of Machines (DVM)*. 2020, pp. 1–5. DOI: 10.1109/DVM49764.2020.9243895.

[9] Michele Calo'. *Automazione di un banco elettropneumatico con controllo mediante PLC e HMI*. 2020/2021, pp. 33–86.