

## Table of contents

Table of contents.....	1
Preface .....	3
Abstract .....	4
List of abbreviations.....	5
Company Description .....	6
Objective .....	7
1 Introduction .....	8
2 Technologies .....	10
2.1 EtherCAT.....	10
3 Hardware.....	14
3.1 The pump installation .....	14
3.1.1 Temperature sensor .....	15
3.1.2 Pressure sensor .....	16
3.1.3 Optical speed sensor .....	18
3.1.4 Motor controller.....	20
3.1.5 Demultiplexer.....	21
3.2 Beckhoff devices.....	23
3.2.1 Control Panel CP66XX: CP6607-0001-0000.....	23
3.2.2 EK1100: EtherCAT Coupler for E-bus terminals .....	24
3.2.3 EL1008: E-BUS 8-channel digital input terminal 24V DC.....	25
3.2.4 EL2008: E-BUS 8-channel digital output terminals 24 V DC, 0.5 A.....	26
3.2.5 EL3002: E-BUS 2- channel analog input terminals -10...+10 V .....	27
3.2.6 EL4002: E-BUS 2-channel analog output 0...10V, 12bit .....	28
3.2.7 BK1250 EtherCAT "Compact" Coupler between E-bus and K-bus terminals .....	29
3.2.8 KL2531 Stepper motor terminal 24V DC,1.5A, 2phases, 2 digital inputs.....	31
3.2.9 KL9010: K-bus end terminal .....	32

4	Software .....	33
4.1	Structured Text.....	33
4.2	TwinCAT.....	38
4.2.1	TwinCAT PLC control .....	38
4.2.2	TwinCAT system manager .....	40
4.3	The Code.....	46
5	Conclusions .....	49
6	Appendices .....	52
	Appendix A: The Code .....	52
7	Literature and references.....	57

## Preface

There are a lot of people that have participated in this project and we would like to thank them all for their help.

Beginning with our promoters, Vincent Claes, who has given us the opportunity to work at this project and who has contributed with his time and knowledge, and Patrick Hilven, as program coordinator of the electronics-ICT department. Also Gwen Vanheusden, for helping us to start using EtherCAT and Beckhoff technology.

We are also grateful to the office of Beckhoff in Hasselt, for the technical support and their advice about how to use their technology.

We also want to thank our families who gave us the opportunity to follow these studies and the support during our studies.

And last but not least, we would like to say thank you to all our friends who supported us in every possible way they could.

As an Erasmus Student, I, David Pérez, would like to express my gratitude to those who made it possible for me to undertake this Erasmus experience. The staff from XIOS Hogeschool Limburg: Els Wieërs, as International Coordinator in Industrial Sciences and Technology department. Kristien Bauwens, as the International Coordinator in XIOS Hogeschool Limburg, and Moniek Gielen, who is responsible for StuvoX, for their help. On my origin institution, Universidad del País Vasco – Basque Country University, I appreciate their help from International Relationship department, with Rosa M<sup>a</sup> del Rio as the head on the department at my school. As well, I would like to mention all the Erasmus students I have met here and finally.

## Abstract

EtherNET like we know it, is a network protocol which computers use to communicate with each other through a Local Area Network i.e. LAN. This is a widely used protocol in daily life, where transport of big amounts of data is common. However for industrial applications, where transport of smaller amounts of data is more common, this protocol is not ideal.

EtherCAT provides an innovative solution for this problem. It is a protocol that is becoming more and more used at the moment. Using the standard connections of an EtherNET communication, we get an efficient industrial network with a low cost wiring compared with other industrial networks.

The goal of our project is about to implement an EtherCAT network to control a pump installation, using Beckhoff equipment.

To get this goal, we will implement a Beckhoff system which will be able to control the installation automatically. This gives us the possibility to work in different areas of electronics. To start, we have to make the sensors of the pump installation compatible with the Beckhoff slave devices. Next step will be to use the data from the sensors to control the installation. We will use a Beckhoff software tool called TwinCAT to program a PLC in Structured Text language. This PLC will work as a master device in the automated pump installation. Finally, we will create an interface which will be shown on a touch panel to control and configure the variables of the pump installation.

## List of abbreviations

ADS:	Automation Device Specification
CRC (Checksum):	Cyclic Redundancy Check
EtherCAT:	Ethernet for Control Automation Technology
HMI:	Human Machine Interface
I/O:	Inputs/Outputs
IP:	Internet Protocol
LAN:	Local Area Network
Op. Amp.:	Operational Amplifier
PC:	Personal Computer
PCB:	Printed Circuit Board
PCI:	Peripheral Component Interconnect
PLC:	Programmable Logic Controller
RT:	Real Time (computing)
TFT:	Thin-Film Transistor
USB:	Universal Serial Bus

## Company Description

We are working for FPGA-2. This is a research department of the XIOS Hogeschool Diepenbeek. FPDA-2 stands for **FPGA Platform with Data-I/O capabilities for industrial and education Applications-2**. This department does researches in a domain that is important for the school. It is focused on practical researches of modular embedded systems. Hereby they can combine more specifications without giving in on quality. Because there is still no clear line between data- and telecommunication, there is need for a user-friendly development environment that can produce a satisfactory program in a short time. Here we can do the research we need to understand the EtherCAT protocol.

## **Objective**

The final aim of this project is to be able to control an automation system composed by a pump installation (with some sensors and actuators on it) and a PLC placed inside a touch panel and some devices with I/O to communicate between the pump installation and the PLC.

To make this possible, it will be necessary to use and learn about the EtherCAT fieldbus as well as the Beckhoff devices, including the software TwinCAT and Structured Text language to program the PLC.

## 1 Introduction

This project is mainly thought as an introduction in Beckhoff technology and how to implement it on an automated system. To realize the project we will use several Beckhoff devices, from a PLC working as a master device to some other slave devices. The way all these devices communicate with each other is also an innovative solution: EtherCAT (Ethernet for Control Automation Technology). It is an Ethernet based technology which has been adapted to the industrial requirements, using the same cables and connectors as Ethernet. To program the PLC we use TwinCAT, it is a software program developed by Beckhoff. We also use it to create the interface for a touch panel which gives us the possibility to monitor and control the data of the slave devices. Finally, to test some of the possibilities of this technology, an Armfield pump installation will be used as a system to be controlled by our automation system.

The Beckhoff Group is a company which has been working in automation for 30 years now. They are specialized in automation systems based on PC control technology. To get the goal of our project, we will use a Touch Panel CP6607 which works under Windows CE 6.0, running a PLC. This will be the master device of the system. Then, to communicate with sensors and motors, it will be necessary to use several slave devices, also from Beckhoff.

For the communication between all these devices, Beckhoff offers several buses and field buses to connect them. We have chosen to use the option of the EtherCAT field bus for our application. EtherCAT is based on the Ethernet protocol, but it is improved for automated systems. In fact, the main difference with Ethernet is the option of sending and receiving short data messages in short periods of time, what is essentially the most important feature in an automated field bus. With EtherCAT we get an easy wiring and low cost connection between all the devices because it uses exactly the same wiring as Ethernet.

The purpose of the slave devices is to measure or control the sensors and motors from the pump installation. In the pump installation we can find two water pumps, three pressure sensors, two optical speed sensors and one temperature sensor. We have analog and digital inputs and outputs, a stepper motor terminal, etc. Normally, it would be enough just connecting the terminals in the devices with the outputs of the sensors or the connectors of the step motor, but first it is important to be sure that the electrical characteristics of both, slave devices and sensor/motors, are compatible. As it wasn't our case, we also designed new circuits to create a proper output signal in the sensors so then we can read/write them from the slaves' terminals.

Once we have finished introducing all the hardware involve in this project, we still should introduce the software we have used. It is also developed by Beckhoff and its name is TwinCAT. TwinCAT is a group of programs with different purposes, but all together we get our objective, to program the PLC inside the touch panel to get all the information we want from the slave devices and control the pump installation. Inside the TwinCAT software we can find the two programs we need to use for our project. First, we will use PLC Control to create the program in Structured Text language. Also we can create a visualization that will work as an HMI (Human-Machine Interface) in the touch panel. When the program is finished, there is another tool inside TwinCAT called System Manager; its function is to communicate the PC with the master device of the automated system.

Finally we get to our objective. To get the control of our pump installation, from a touch panel screen where we can see the state and measurements from the installation and also adjust any settings instantly.

## 2 Technologies

### 2.1 EtherCAT

EtherCAT is a fieldbus system designed for industrial networks and automation applications.

The way it works, is very similar to Ethernet, but EtherCAT improves some characteristics from it. Mainly, it improves performance, deterministic (in RT applications) and it is cheaper.

Functional principle of EtherCAT is one of the differences with Ethernet. While in Ethernet, every Ethernet packet is opened in every node, with EtherCAT, we send the data also as an Ethernet packet, but it will be opened only in those nodes we want to. So, what we have is a master device with n slave devices. The master will send the packet, and slaves will know when they are needed. In case one slave is called, it will attend the instructions taking out the information it needs and, once it has finished the instructions, giving back data into the Ethernet packet so the master will receive it. Anyway, probably not all the slaves would be used in each packet, but the packet will pass through every node and in every node slaves will decide if they have to open it or give back any data. For this reason, the working of EtherCAT is called “on the fly”.

For transmission with EtherCAT, we will use the standard IEEE 802.3 Ethernet frame using the EtherType 0x88a4. With EtherCAT use over Ethernet network, process data will be improved.

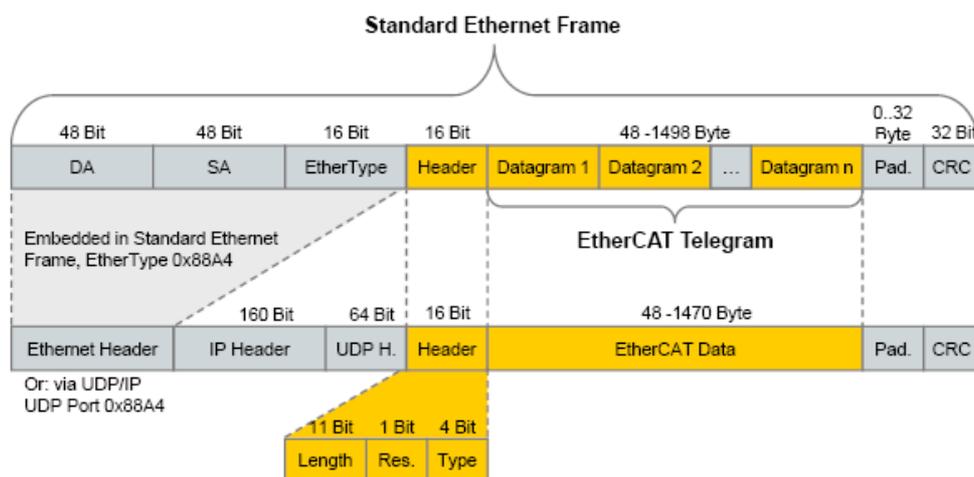
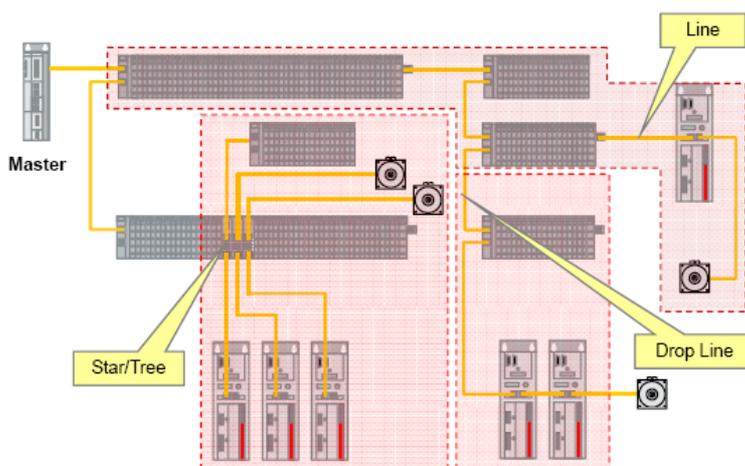


Figure 1: EtherCAT data inside Ethernet frame

Data sequence is independent from the order of physical in the network, that means addressing can follow any order. Broadcast, multicast and communication between slaves is possible if there is a master slave to control it.

One of the most important advantages of using EtherCAT instead of using Ethernet is that it is possible to use almost any typology you want (remind, that in Ethernet star typology is used most of the time). This means, in EtherCAT systems, we can use tree, star, line, drop line ... and all of them in the same system. This allows making our system so much flexible.

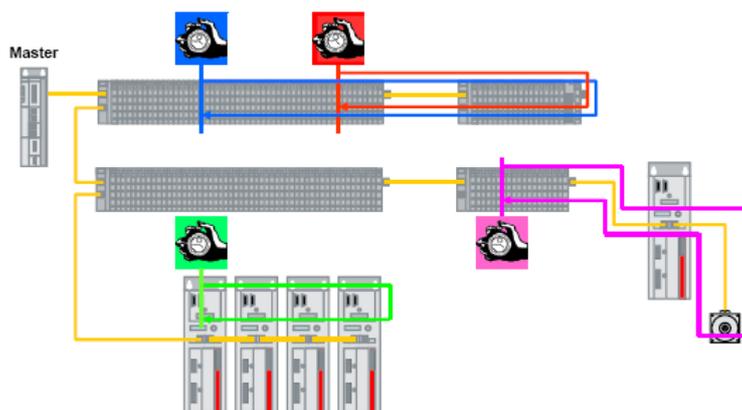


**Figure 2: Different typologies**

Also, costs will be lower because of the difference of wire is saved with this typology instead of using only star typology.

Finally, what we have is an open ring typology going through slave devices. When we send a packet, the first slave will receive it, and once the packet has been in every node, data returns to master device. In this way, we can place more importance on critical devices in the network, just connecting them in drop lines alone, and then use lines to connect less important devices. This could be useful to make failures in the network easier to find.

EtherCAT is a perfect fieldbus system which we can use in I/O level. It is very efficient and we can get good data transfer speed. Anyway, to make the best use of our EtherCAT network, it is important to synchronize it. To synchronize EtherCAT, the master sends a message to every slave and takes time measurements for two times in each slave: the time of the message to arrive into the slave and the time the message leaves the slave. This information then is used as reference system clock. The master calculates the delay of each slave and after that it will configure an offset time for every slave. The time of the first slave will be the time of the system clock.



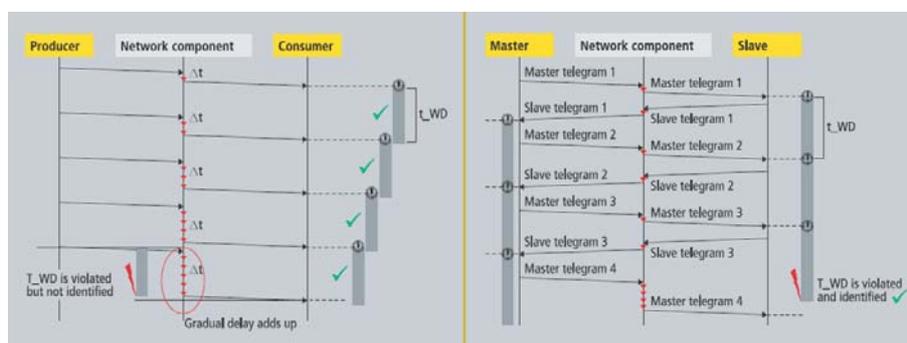
**Figure 3: Slave delays**

To keep slaves synchronized after the first synchronization, it is good to repeat the process once in a while because of some effects that could appear, like differences between internal clocks of each slave.

EtherCAT is also prepared for safety communications and instead of looking for solutions after the technology is developed, EtherCAT integrates safety technology into the machine design.

We can detect several error assumptions in a network: corruption, repetition, interchanging, loss, delay, insertion, masquerading and invalid addressing of messages.

The main protection, due to control RT critical situations, is against message delay. If we only use a watchdog, we will not detect any delays because of  $\Delta t$  in each cycle. One consequence of this could be that we may lose some messages without knowledge of it. To solve this problem, EtherCAT uses a master/slave relationship. The safety over EtherCAT connection can ensure that each device only returns its own message once a new message has been received. If a new message is sent before the last one has been received (because of the accumulation of  $\Delta t$ ), we will detect the failure.



**Figure 4: Master/slave relationship**

With safety over the EtherCAT protocol we can also detect some other failures. I.e.: Startup sequence, which will be controlled by a "session-number" that will detect buffering of a complete startup sequence. Misrouted messages, we will give a unique connection ID and a unique slave address to detect them. Message corruption from source to sink, with a CRC checksum (CRC16) and finally, we can also detect interchange, repetition, insertion or loss of whole messages just giving a sequence number to each message.

## 3 Hardware

### 3.1 The pump installation

Because we cannot test our equipment without an application, the school gave us a demonstration unit with a pump installation:



**Figure 5: Armfield pump installation**

The unit contains a water reservoir and two motor driven centrifugal pumps, which can be configured for single pump operation, two pumps in parallel or two pumps in series. You can choose this by manually open or close some ball valves. The package also includes some sensors:

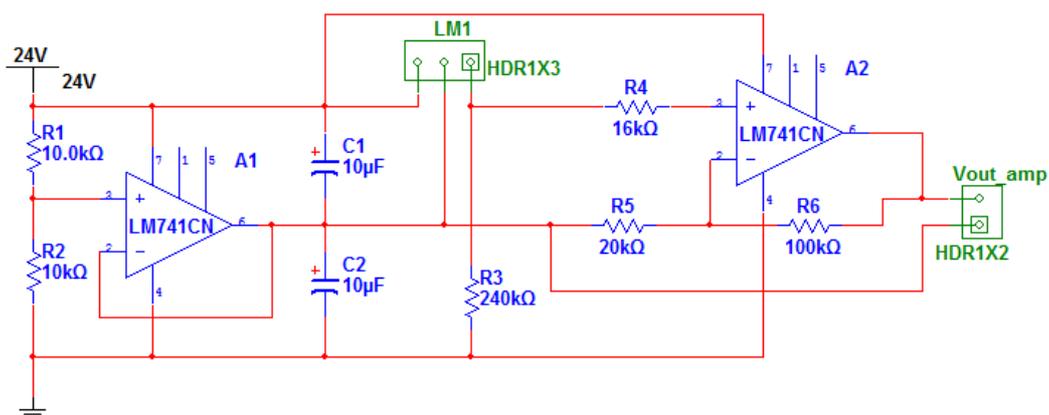
- 2 pressure sensors to measure the pressure produced by each pump
- 1 pressure sensor to measure the flow rate
- 1 temperature sensor to measure the temperature of the water
- 2 optical speed sensors to measure the speed of the motors

The use of these sensors is to monitor the characteristics of the different kinds of pump configurations. After that you can compare the effects of the different configurations to see what happens and which configuration is the best to use.

Now our job is to adapt all those sensors so they all work with 24 volts and give an output voltage that we can connect to the Beckhoff system.



When we put everything together, we get this:

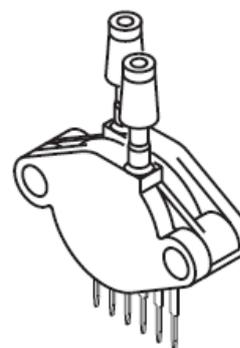


**Figure 9: Schematic of the temperature sensor**

We use the Op-Amp to amplify the output signal of the sensor with a factor 6.

### 3.1.2 Pressure sensor

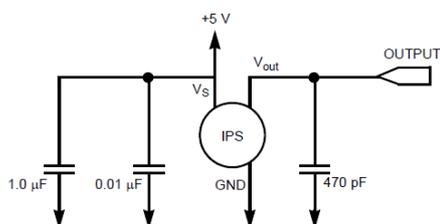
To measure the pressure we use an MPX5100DP. It works with a piezoresistive element and it is designed for a wide range of applications. This sensor is developed using high technologic techniques such as advanced micromachining, thin-film metallization and bipolar processing. Because of that it provides an accurate, high-level analog output signal that is proportional to the applied pressure. The output signal is a value between 0,2 and 4,7 Volts. The pressure can be calculated this way:



**Figure 10: MPX5100P**

$$V_{out} = V_s \cdot (P \times 0,009 + 0,04)$$

Here you can see how we have connected it:



**Figure 11**

- We use a supply voltage of 5V
- The capacitors are necessary for decoupling and output filtering.

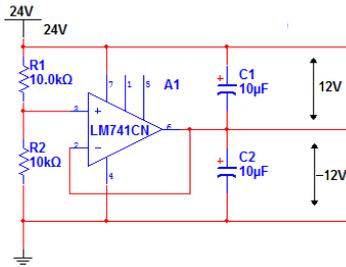


Figure 12

- We make -12v and +12V again to feed the Op-amp.

This is the scheme of how we putted it al together:

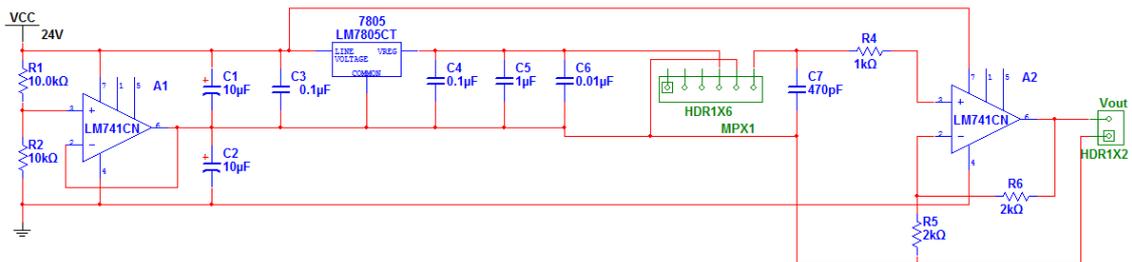


Figure 13: Schematic of the pressure sensors

We use that op-amp to amplify the output of the sensor with a factor 2.

### 3.1.3 Optical speed sensor

To measure the speed of the motors that drive the pumps, we placed a reflective sticker on one half of the black motor shaft like shown on the image:



Figure 14:  
Motor shaft

Then we use an optical reflecting objective sensor, the OPB704. This sensor consists of an infrared emitting diode and a NPN silicon phototransistor mounted side-by-side in one plastic housing. When a reflective object passes within the field of view of the sensor, it reflects the radiation from the emitter. This reflected radiation is detected by the phototransistor. We use a supply voltage of 5V, so we get an output voltage somewhere between 0 and 5 volts if a reflected signal is detected, depending of the distance between the sensor and the object.



Figure 15:  
OPB704

When we now place the sensor right in front of the motor shaft, we get a pulsed output signal somewhere between 0 and 5 volts. We can use this signal as a digital signal. But one problem we have is that the digital input module of the Beckhoff PLC needs a signal between 0 and 24 volts. Therefore we made a smith trigger:

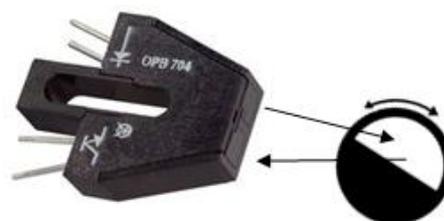


Figure 16: Reflection of motor shaft

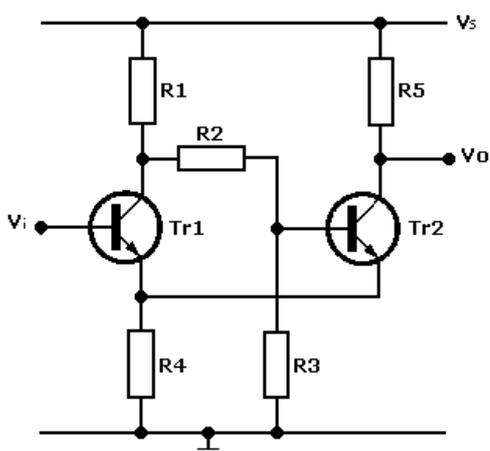


Figure 17

When there is no voltage  $V_i$  supplied to the transistor  $Tr_1$ , the transistor  $Tr_2$  will conduct. A voltage drop appears across resistor  $R_5$ , and  $V_o$  will be zero. Across  $R_4$  there is a voltage that depends on its own value and the emitter current of  $Tr_2$ . Now when  $V_i$  gets higher than the voltage across  $R_4$ , there will be a change.

$Tr_1$  starts to conduct now and its collector voltage will drop thus also the voltage on the base of  $Tr_2$ . Real quick now  $Tr_1$  will conduct fully while  $Tr_2$  will close. By this reason the output voltage  $V_o$  becomes the same as the supply voltage  $V_s$ . If now  $V_i$  still rises, it will not influence  $V_o$ .

When  $V_i$  drops again, the collector current of  $Tr_1$  will decrease and the collector voltage will increase. The base voltage of  $Tr_2$  will increase too until it gets high enough to let  $Tr_2$  conduct again. Again this change will occur very fast. We now have the same state as when we started. The difference between these two critical voltages is called hysteresis.

If we now use the output of the sensor as input for  $Tr_1$  and 24 volts as supply voltage, we get a nice digital pulsed signal between 0 and 24 volts:

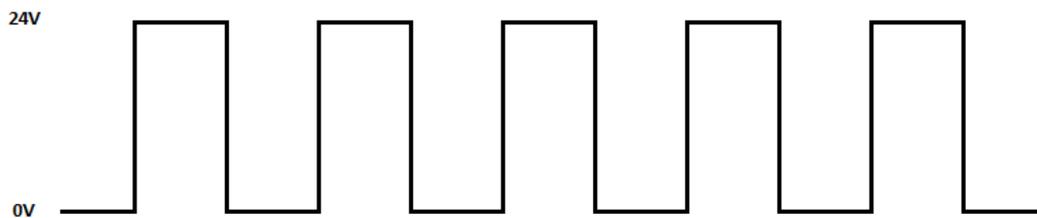


Figure 18

This signal we can easily connect to the PLC to measure the frequency of the pulses. This frequency is the same as the speed of the motor.

This is how we have made it:

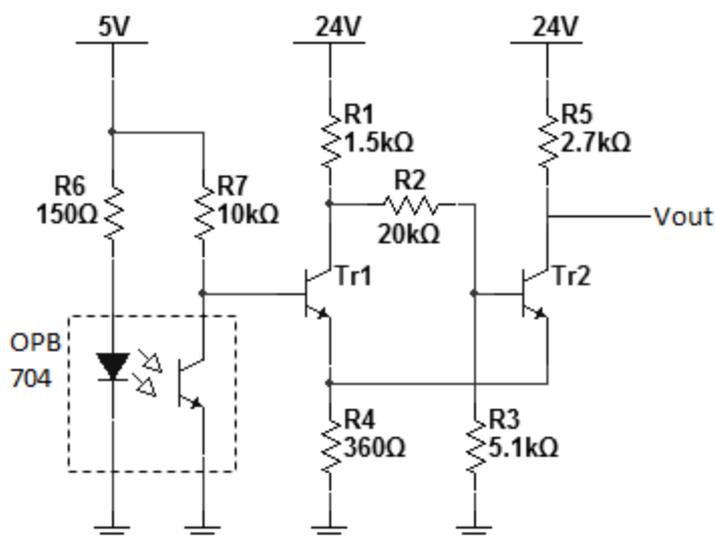


Figure 19: Schematic of the optical speed sensor

### 3.1.4 Motor controller

At the beginning, the adjustment of the speed of the pumps had to be done manually. But because we wanted to control everything from the touch panel, we placed two stepping motors that take over the manual control. The stepping motor that we use is one from RS-components. It works on 12V and uses 0,16mA. It is a uni-polar stepper motor; this means that it has two windings per phase.



Figure 20: Stepping motor

To control this motor, we need two H-bridges per motor. For this reason we use a L298N dual full-bridge driver because it contains 2 bridges.

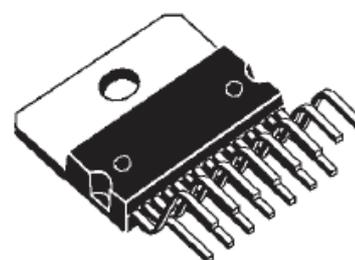


Figure 21: L298N

To connect it with the PLC, we need four digital outputs. But because we only got eight and we are already using one for the demultiplexer (see next paragraph), we use another slave module: EtherCAT SPI Slave. This module was designed by two other students, Olivier Ottenburgs and Dieter Vanrykel. They are also working on a project about EtherCAT. To protect this module for the higher voltages we are using for the motor, we use an optocoupler for each input signal.

Here you can see how we made it:

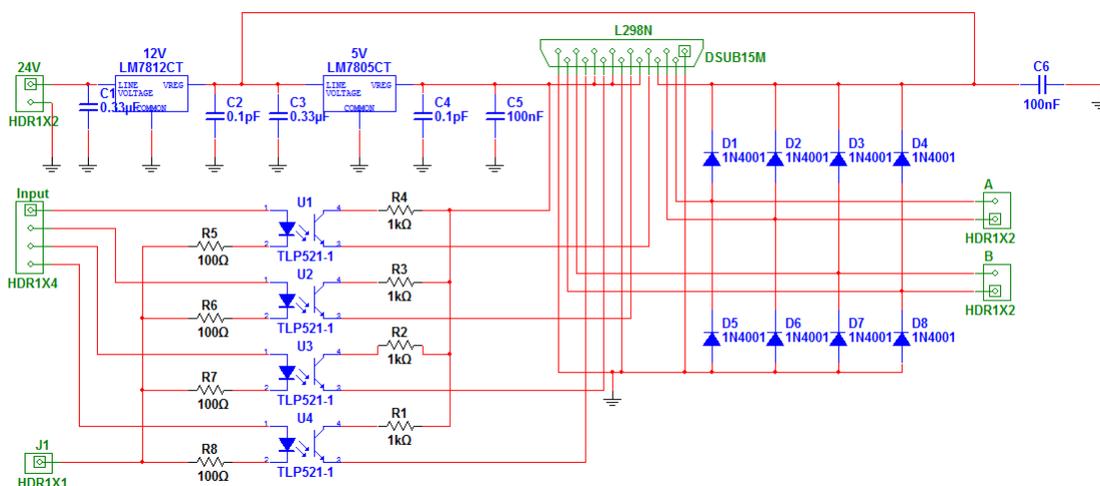


Figure 22: Schematic of the motor controller

### 3.1.5 Demultiplexer

Because we need to process four analog signals from the sensors, but we only have two analog inputs on the EL3002 device from Beckhoff, we have decided to demultiplex the signals into these two inputs.

The kind of demultiplexer we need is an analog mux/demux. The best option would be to use a dual analog mux/demux into one chip, but the one available for us was the HEF4051BP of Philips. This component is a 8-channel analog multiplexer/demultiplexer. The

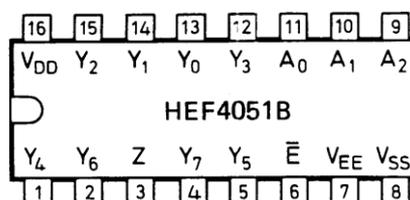


Figure 23: HEF4051B

The disadvantage of using this chip is that, if we want to use only one to simplify the PCB, the four analog signals will be on the same output. That means that we would only use one of the two available analog inputs. Also, we will need longer time cycles on the program to read the value from all the sensors.

This last trouble was the one which made us to decide to use two HEF4051BP instead of one. We are using one chip more than necessary, but our system will be easier to control and the measuring will be faster.

From the hardware point of view, the use of a multiplexer/demultiplexer is not complicated. There are not so many peripheral components necessary. Its function is mainly to connect directly the analog signals to its inputs and connect the outputs also directly to the analog inputs of the slave device EL3002.

The schematic we have used for the PCB is the following one. The signal from the Digital Input is a square signal between +/- 12V, so we use a diode to transform that signal into another one without negative voltage.

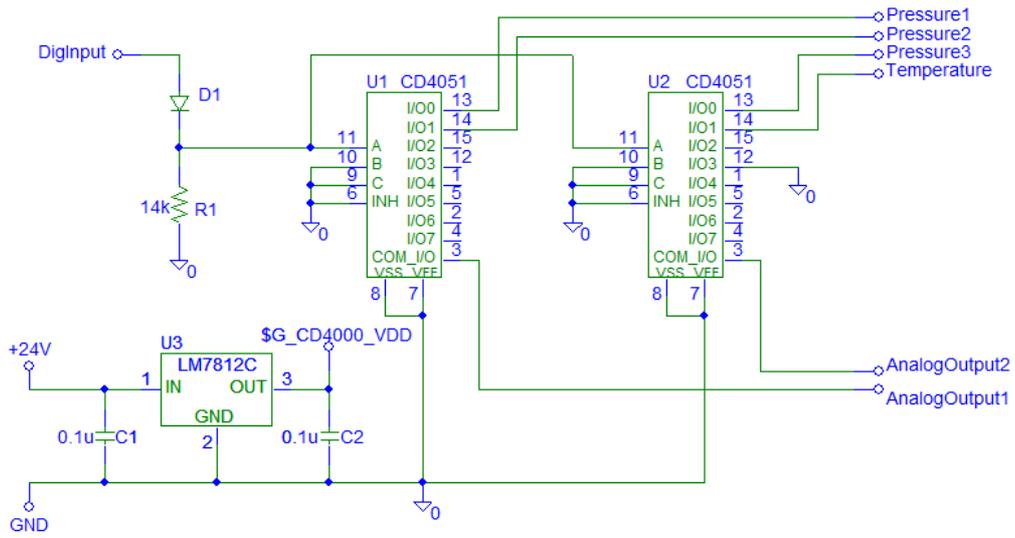


Figure 24: Schematic of the demultiplexer

## 3.2 Beckhoff devices

### 3.2.1 Control Panel CP66XX: CP6607-0001-0000



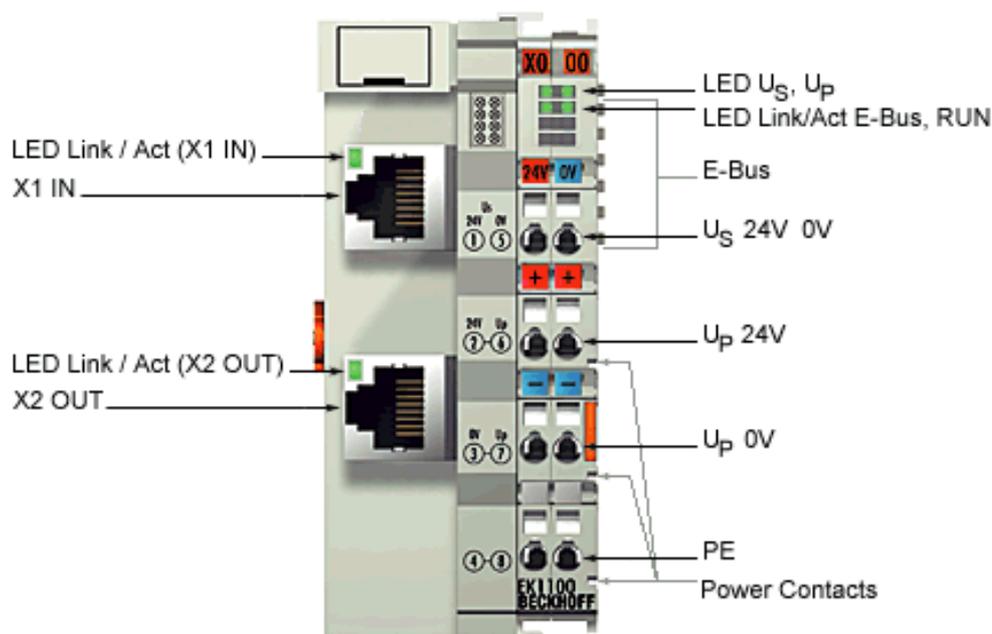
**Figure 25: CP6607 touch panel**

CP6607 is the control panel we are going to use to show all the information from our system. It is a TFT display of 5.7 inch with a resolution of 640x480. This model also works as a resistive touch panel, so it will be able to control and adjust settings from it, without the necessity of using a computer once the program is done and working properly.

The main characteristics are:

- integrated PC, 3½-inch motherboard for Intel® IXP processor with XScale® technology
- Intel® IXP420 with XScale® technology, clock frequency 533 MHz
- 128 MB on-board RAM
- 32 MB on-board flash
- lithium battery of the system clock, accessible from the rear side
- on-board graphic adapter, graphic chip SM501 (max. resolution 1024 x 768)
- 1 Mini PCI slot free, e.g. for a fieldbus interface card
- 2 USB ports
- 2 RJ 45 Ethernet connectors 10/100 Mbits
- 1 RS232 serial port, D-sub, 9-pin
- all connectors at the lower rear side
- 24 V power supply
- operating temperature 0...55 °C
- pull-out clamping levers for fast installation without loose parts
- operating system Microsoft Windows CE, English

### 3.2.2 EK1100: EtherCAT Coupler for E-bus terminals



**Figure 26: EK1100 EtherCAT coupler for E bus terminals**

The EK1100 is a bus coupler to connect EtherCAT with EtherCAT terminals (ELxxxx). This means, that to use the modules we want to use, e.g.: digital/analog inputs/outputs, etc. they belong to the group of devices ELxxxx. To connect them, first we need to use an EK1100. Its function is to convert the passing telegrams from Ethernet 100BASE-TX to the E-bus signal representation.

The proper use of EK1100 is connecting it to the network (where EtherCAT telegrams are being transferred) via the upper Ethernet interface. There is only one restriction about connecting the EK1100 bus coupler, and that is that it is not allowed to connect it directly to a switch. If we want or we need to connect the coupler to a switch, then we need the EK1000 that could work in that situation.

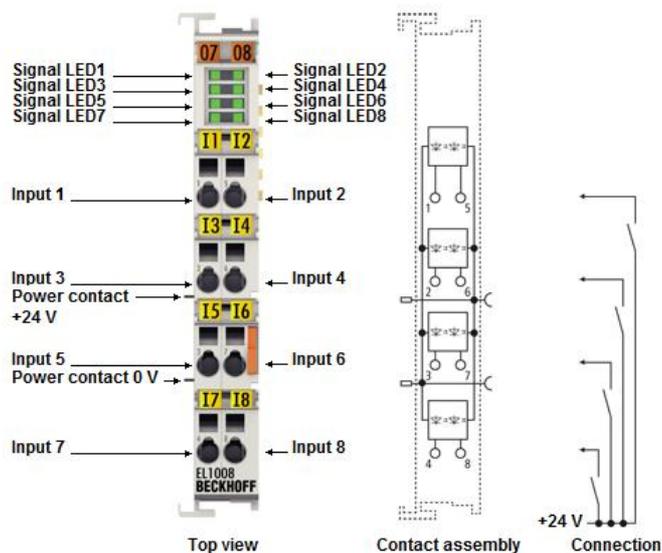
The lower Ethernet connection can be used to extend the EtherCAT network to more compatible devices.

The diagnoses we can do due to the leds are:

LED	Display	Status	Description	
U <sub>S</sub> 24 V	green	on	on	power supply for Bus Coupler electronic and E-Bus is active
U <sub>P</sub> 24 V	green	on	on	power supply for Power Contacts is active
Link/Act E-Bus	green	off	-	no connection / no Communication with internal E-Bus
		on	linked	E-Bus device (bus terminal) connected
		flashes	active	Communication with E-Bus device (bus terminal)
RUN	green	off	Init	the bus coupler has the status Initialization
		flashes	pre operational	the bus coupler has the status Pre Operational
		single flash	safe operational	the bus coupler has the status Safe Operational
		on	operational	the bus coupler has the status Operational
		flickering	bootstrap	A firmware is being loaded
Link/Act (X1 IN)	green	off	-	no connection / no Communication at the incoming EtherCAT cable
		on	linked	former EtherCAT device is connected
		flashes	active	Communication with former EtherCAT device
Link/Act (X2 OUT)	green	off	-	no connection / no Communication at the outgoing EtherCAT cable
		on	linked	following EtherCAT-device is connected
		flashes	active	Communication with following EtherCAT device

**Table 1: Specifications of the EK1100**

### 3.2.3 EL1008: E-BUS 8-channel digital input terminal 24V DC



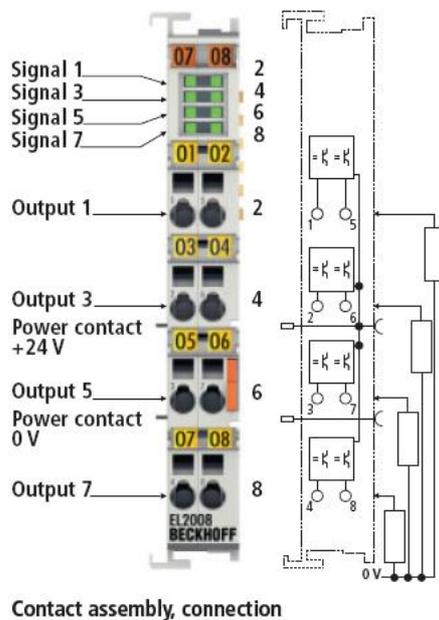
**Figure 27: EL1008 digital input terminal**

The EL1008 is an EtherCAT terminal with 8 digital inputs. With this device we can acquire 8 binary control signals from the pump installation and transmit them, in an electrically isolated form, to the PLC. Digital input terminals from the EL100x series have a 3 ms input filter. Also include in the upper side, as the rest of EtherCAT terminals, leds to indicate the state every connection.

Technical data	EL1002   ES1002	EL1004   ES1004	EL1008   ES1008
Number of inputs	2	4	8
Nominal voltage	24 V DC (-15 %/+20 %)		
"0" signal voltage	-3...+5 V (EN 61131-2, type 3)		
"1" signal voltage	15...30 V (EN 61131-2, type 3)		
Input current	typ. 3 mA (EN 61131-2, type 3)		
Input filter	3.0 ms		
Current consumption E-bus	typ. 90 mA		
Electrical isolation	500 V <sub>max</sub> (E-bus/field potential)		
Bit width in the process image	2 inputs	4 inputs	8 inputs
Configuration	no address or configuration setting		
Weight	approx. 50 g	approx. 50 g	approx. 55 g
Operating/storage temperature	0...+55 °C/-25...+85 °C		
Relative humidity	95 %, no condensation		
Vibration/shock resistance	conforms to EN 60068-2-6/EN 60068-2-27/29		
EMC immunity/emission	conforms to EN 61000-6-2/EN 61000-6-4		
Protect. class/installation pos.	IP 20/variable	IP 20/variable	IP 20/see documentation
Pluggable wiring	for all ESxxxx terminals		
Further information	www.beckhoff.com/EL1002		

**Tabel 2: Specifications of the EL1008**

### 3.2.4 EL2008: E-BUS 8-channel digital output terminals 24 V DC, 0.5 A



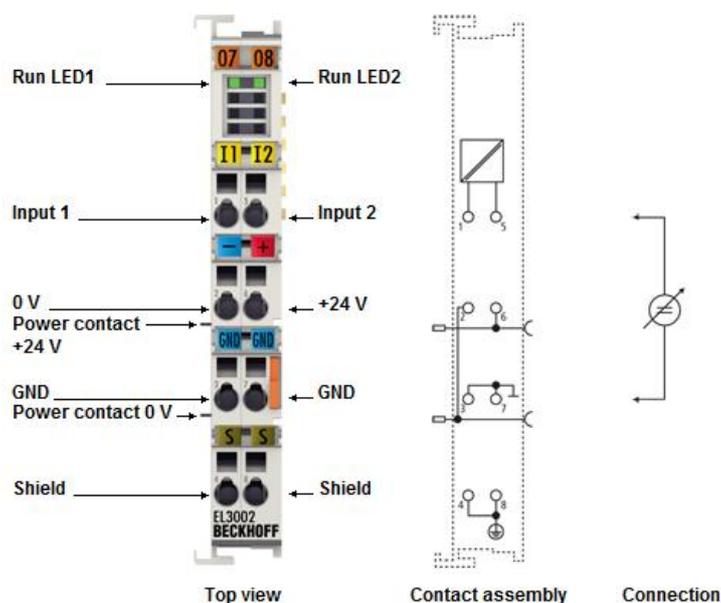
**Figure 28: EL2008 digital output terminal**

The EL2008 is an EtherCAT terminal with 8 digital outputs. This device will be useful to send digital signals which have been created before in our program. In this way, we can control external digital devices easily.

Technical data	EL2002   ES2002	EL2004   ES2004	EL2008   ES2008
Number of outputs	2	4	8
Rated load voltage	24 V DC (-15 %/+20 %)		
Load type	ohmic, inductive, lamp load		
Max. output current	0.5 A (short-circuit-proof) per channel		
Reverse voltage protection	yes		
Breaking energy	< 150 mJ/channel		
Switching times	typ. T <sub>ON</sub> : 60 µs, typ. T <sub>OFF</sub> : 300 µs		
Current consumption E-bus	typ. 100 mA	typ. 100 mA	typ. 110 mA
Electrical isolation	500 V <sub>ins</sub> (E-bus/field potential)		
Current consumption power contacts	typ. 15 mA + load		
Bit width in the process image	2 outputs	4 outputs	8 outputs
Configuration	no address or configuration setting		
Weight	approx. 55 g		
Operating/storage temperature	0...+55 °C/-25...+85 °C		
Relative humidity	95 %, no condensation		
Vibration/shock resistance	conforms to EN 60068-2-6/EN 60068-2-27/29		
EMC immunity/emission	conforms to EN 61000-6-2/EN 61000-6-4		
Protect. class/installation pos.	IP 20/variable	IP 20/variable	IP 20/see documentation
Pluggable wiring	for all ESxxxx terminals		
Further information	www.beckhoff.com/EL2002		

**Tabel 3: Specifications of the EL2008**

### 3.2.5 EL3002: E-BUS 2- channel analog input terminals -10...+10 V



**Figure 29: EL3002 analog input terminal**

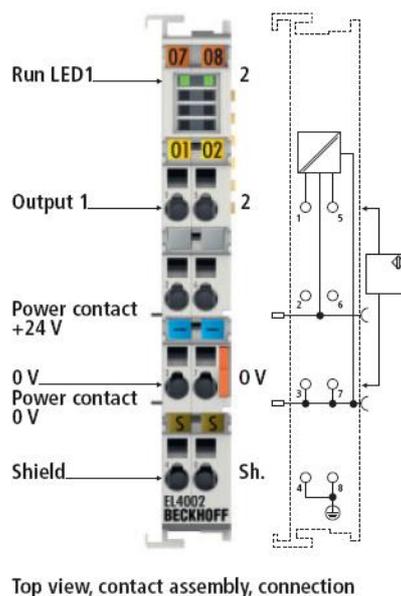
Because we are going to use several analog signals from the pump installation, this terminal will be especially useful. The EL3002 is a 2-channel analog input device. The range it is able to process the signals is between -10 and +10V. When we read the value of the inputs, we are not going to get the absolute value. What we get is a digitized value of the voltage, with a

resolution of 12bits. The inputs terminals of EL3002 are single-ended with a common, internal ground potential.

Technical data	EL3001   ES3001	EL3002   ES3002
Number of inputs	1 (single-ended)	2 (single-ended)
Power supply	via the E-bus	
Signal voltage	-10...+10 V	
Internal resistance	> 130 kΩ	
Input filter limit frequency	1 kHz	
Conversion time	typ. 0.625 ms (default setting: 50 Hz filter)	
Resolution	12 bits (16 bits presentation)	
Measuring error	< ±0.3 % (relative to full scale value)	
Surge voltage resistance	30 V max.	
Electrical isolation	500 V <sub>rms</sub> (E-bus/field potential)	
Current consumption E-bus	typ. 130 mA	
Bit width in the process image	inputs: 4 byte	inputs: 8 byte
Configuration	no address or configuration setting required	
Weight	approx. 70 g	
Operating/storage temperature	0...+55 °C/-25...+85 °C	
Relative humidity	95 %, no condensation	
Vibration/shock resistance	conforms to EN 60068-2-6/EN 60068-2-27/29	
EMC immunity/emission	conforms to EN 61000-6-2/EN 61000-6-4	
Protect. class/installation pos.	IP 20/variable	
Pluggable wiring	for all ESxxx terminals	

**Table 4: Specifications of the EL3002**

### 3.2.6 EL4002: E-BUS 2-channel analog output 0...10V, 12bit



**Figure 30: EL4002 analog output terminal**

The EL4002 is a 2-channel analog output terminal which generates signals in the range between 0 and 10 V. The voltage is supplied to the process level with a resolution of 12 bits (EL4001/EL4002) and is electrically isolated. The output channels of the EtherCAT Terminals have a common ground potential

Technical data	EL4001   ES4001	EL4002   ES4002	EL4102   ES4102
Number of outputs	1	2	2
Power supply	24 V DC via the power contacts	24 V DC via the power contacts	via the E-bus
Signal voltage	0...10 V		
Distributed clocks	yes		
Load	> 5 k $\Omega$ (short-circuit-proof)		
Measuring error	< 0.1 % (relative to full scale value)	< 0.1 % (relative to full scale value)	$\pm$ 0.1 % (relative to full scale value)
Resolution	12 bits	12 bits	16 bits (incl. sign)
Electrical isolation	500 V <sub>ins</sub> (E-bus/signal voltage)		
Conversion time	~ 100 $\mu$ s	~ 150 $\mu$ s	~ 40 $\mu$ s (fast mode ~ 25 $\mu$ s)
Current consumption E-bus	typ. 190 mA	typ. 190 mA	typ. 210 mA
Bit width in the process image	1 x 16 bit AO output	2 x 16 bit AO output	2 x 16 bit AO output
Weight	approx. 60 g		
Operating/storage temperature	0...+55 °C/-25...+85 °C		
Relative humidity	95 %, no condensation		
Vibration/shock resistance	conforms to EN 60068-2-6/EN 60068-2-27/29		
EMC immunity/emission	conforms to EN 61000-6-2/EN 61000-6-4		
Protect. class/installation pos.	IP 20/variable		
Pluggable wiring	for all ESxxxx terminals		
Further information	www.beckhoff.com/EL4001		

Table 5: Specifications of the EL4002

### 3.2.7 BK1250 EtherCAT "Compact" Coupler between E-bus and K-bus terminals

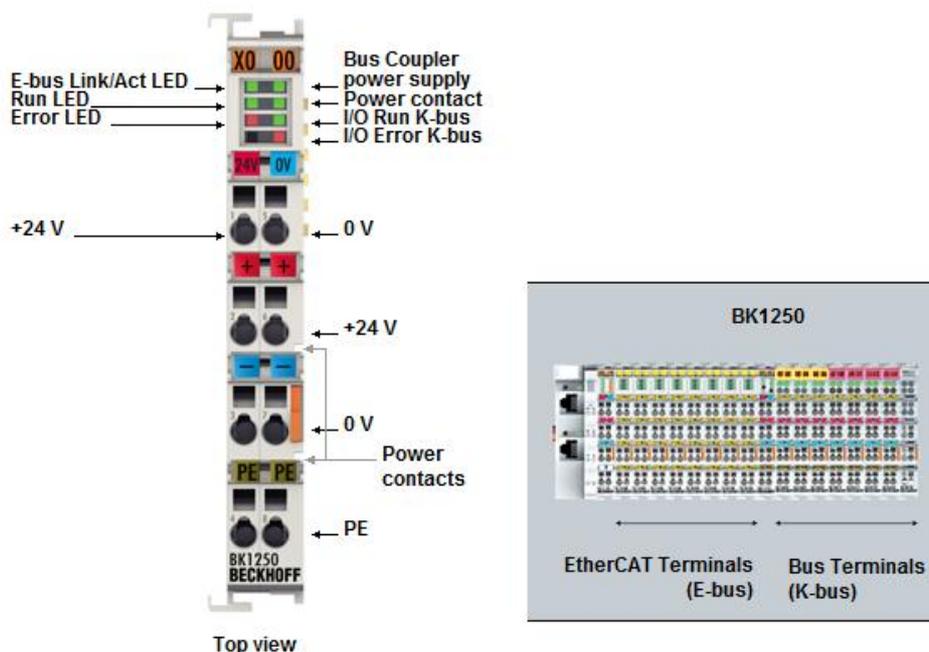


Figure 31: BK1250 coupler between E-bus and K-bus terminals

The BK1250 is the solution to use in the same block EtherCAT terminals (ELxxxx) and standard Bus Terminals (KLxxxx). In our case, we will use a standard Bus Terminal to control a stepper motor. With this device we can implement it in the same block placing it between the last EtherCAT terminal and the first standard Bus Terminal.

At least 64 Bus Terminals can be connected to a BK1250. Other function of the the Bus Coupler is to recognize the Bus Terminals which are connected and automatically allocates them into the EtherCAT process image.

Technical data	BK1250
Number of Bus Terminals	64 (255 with K-bus extension)
Max. number of bytes fieldbus	1,024 byte input and 1,024 byte output
Digital peripheral signals	8,192 inputs/outputs
Analog peripheral signals	256 inputs/outputs
Protocol	EtherCAT
Configuration possibility	via KS2000 or EtherCAT (ADS)
Data transfer rates	100 Mbaud
Bus interface	via E-bus contacts
Power supply	24 V DC (-15 %/+20 %)
Input current	70 mA + (total K-bus current)/4, 200 mA max.
Starting current	approx. 2.5 x continuous current
Recommended fuse	≤ 10 A
Current supply K-bus	max. 500 mA
Power contacts	24 V DC max./10 A max.
Electrical isolation	500 V <sub>rms</sub> (power contact/supply voltage/Ethernet)
Weight	approx. 55 g
Operating/storage temperature	0...+55 °C/-25...+85 °C
Relative humidity	95 %, no condensation
Vibration/shock resistance	conforms to EN 60068-2-6/EN 60068-2-27/29
EMC immunity/emission	conforms to EN 61000-6-2/EN 61000-6-4
Protect. class/installation pos.	IP 20/variable

**Tabel 6: Specifications of the BK1250**

### 3.2.8 KL2531 Stepper motor terminal 24V DC,1.5A, 2phases, 2 digital inputs

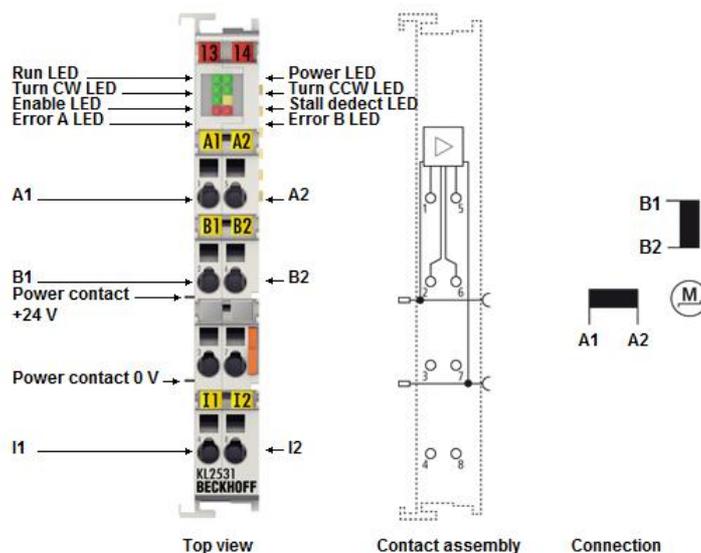


Figure 32: KL2531 Stepper motor terminal

To make it easier to control the stepper motors that will control the power of the water pumps, this device will help us. It is the KL2531 and its function is to control small stepper motors connecting them directly to its terminals. Its 64-fold micro-stepping ensures a quiet and precise motor operation. Encoder system or limit switch may be unnecessary due to the integrated zero-speed monitoring.

Technical data	KL2531   KS2531
Number of outputs	1 Stepper Motor, 2 phases
Number of inputs	2
Output current	2 x 1 A, 2 x 1.5 A peak current, overload- and short-circuit-proof
Maximum step frequency	125,000 steps/s
Step pattern	full step, half step, up to 64-fold micro stepping
Current controller frequency	approx. 25 kHz
Diagnostics LED	error phase A and B, loss of step/stagnation, power, enable
Resolution	approx. 5,000 positions in typ. applications (per revolution)
Power supply	8...24 V DC (for output stage over power contacts)
Electrical isolation	500 V <sub>rms</sub> (K-bus/signal voltage)
Current consumption K-bus	typ. 60 mA
Bit width in the process image	input: n x 2 x 16 bit data, 2 x 16 bit control/status
Weight	50 g
Operating/storage temperature	0...+55 °C/-25...+85 °C
Relative humidity	95 %, no condensation
Vibration/shock resistance	conforms to EN 60068-2-6/EN 60068-2-27/29
EMC immunity/emission	conforms to EN 61000-6-2/EN 61000-6-4
Protect. class/installation pos.	IP 20/variable
Pluggable wiring	for all KSxxxx Bus Terminals

Tabel 7: Specifications of the KL2531

### 3.2.9 KL9010: K-bus end terminal



**Figure 33: KL9010 K-bus end terminal**

If we decide to use standard Bus Terminals (KLxxxx), then we will have to use this device. The function of the KL9010 is to indicate that there isn't any other device. Without it, the communication and the data exchange between the Bus Coupler and the Bus Terminals won't work.

The KL9010 has to be placed at the end of every group of Bus Terminals.

Technical Data	KL9010
Current consumption from the K-Bus	-
Electrical isolation	500 V <sub>eff</sub> (K-Bus/signal voltage)
Bit width in the process image	-
Configuration	no address or configuration settings
Weight	ca. 50 g
Permissible ambient temperature range	-0°C ... +55°C (during operation)
	-25°C ... +70°C (during storage)
Permissible relative humidity	5% ... 95%, no condensation
EMC resistance burst / ESD	conforms to EN 61000-6-2 / EN 61000-6-4
Vibration / shock resistance	conforms to EN 60068-2-6 / EN 60068-2-27, EN 60068-2-29
Protection class	IP20
Installation pos.	Variable
Approval	CE

**Tabel 8: Specifications of the KL9010**

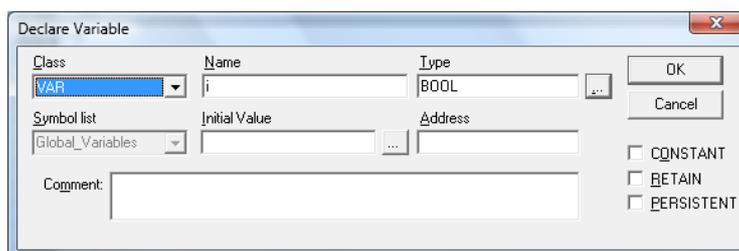
## 4 Software

In this chapter we are going to explain the software we use during the entire project and give a brief guide on how to use it. All these software is produced by Beckhoff.

### 4.1 Structured Text

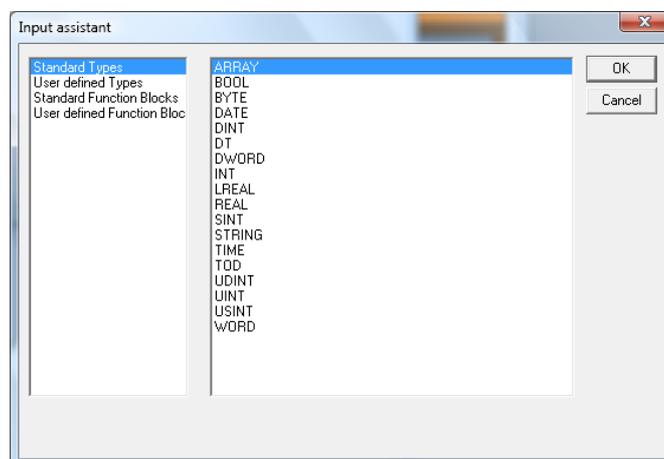
Once we know how to start a program, it would be nice to know how to program it. The appearance is almost the same as in C: the different statements are separated by semicolons. The variables can be used to define values, for internally stored variables and, of course, for inputs and outputs. Spaces can be used to separate statements and variables, although they are not often necessary. Structured text is not case sensitive, but the standard is to write variables in lowercase and statements in uppercase. It can also be helpful to include comments that explain the process of the program to make it easier to understand.

The Structured Text environment has its own “variable detector”, so don’t worry if you forget to declare a variable in your program. After declaring a new variable, a window will appear in the screen, where you can configure it:



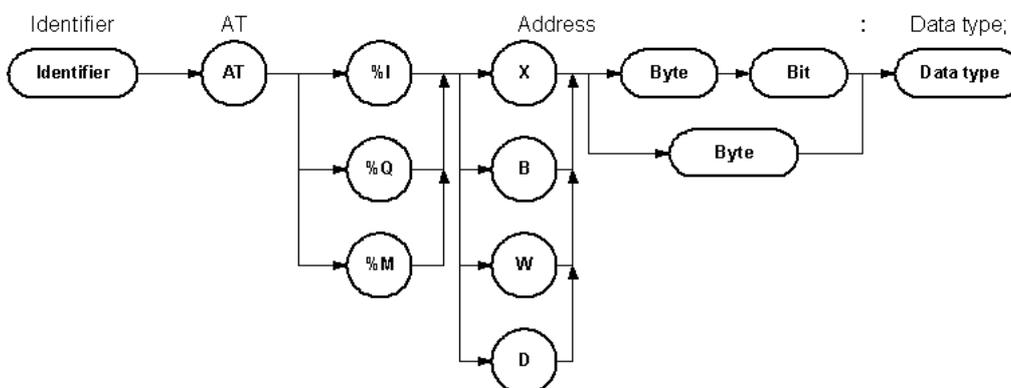
**Figure 34: Declaration of variables**

Also, a menu with all the type of variables can be opened:



**Figure 35: List of all types of variables**

In any case, variables can be declared in the classic way, before using them in the main program (or any function block, etc.). The declaration of variables follows this order:



**Figure 36: Declaration of variables in ST**

Where in the first step the name of the identifier is written (i.e. DigitalOutput), then the connection word AT and after that, a variable will be configured. With %I, %Q or %M, the use of the variable will be specified:

- “I” means that the variable will be used as an Input
- “Q” will configure to Output
- “M” will be used to declare a memory location (not external use).

With the next letter, the length of the variable will be defined. X for bits, B for bytes, W for words (2 bytes) and D for double (4 bytes). It only rest to write the address for the variable and the data type (we have seen all data types available). About the address, it is possible to write a fix address or a random one.

Continuing with the example of DigitalOutput, let’s see how the variable declaration finally looks like:

*DigitalOutput AT %QX0.0: BOOL;* It’s a bit working as an output.

*DigitalOutput AT %QB\*: BYTE;* It’s a byte working as an output, and with automatic address.

There are some names that may not be used to name variables:

START, DATA, PROJECT, SFC, SFC2, LADDER, I/O, ASCII, CAR, FORCE, PLC2, CONFIG, INC, ALL, YES, NO, STRUCTURED TEXT

Also, another difference with other high-level languages is the nature of program flow control. A Structured Text program will be run from the beginning till the end many times each second. A traditional program should not reach the end until it is completely finished.

In conclusion, it is not necessary to start the program with classic “main ()” or end the program with some “}” or “END” as is used in other languages.

Most of the statements used in other languages, are also used in Structured Text. Some examples are:

IF:

```
IF <Boolean_printout1> THEN
    <IF_instructions>
{ELSIF <Boolean_printout2> THEN
    <ELSIF_instructions1>
    .
    .
ELSIF <Boolean_printout n> THEN
    <ELSIF_instructions n-1>
ELSE
    <ELSE_instructions>}
END_IF;
```

FOR:

```
INT_Var :INT;
FOR <INT_Var> := <INIT_VALUE> TO <END_VALUE> {BY <stepsize>} DO
    <instructions>
END_FOR;
```

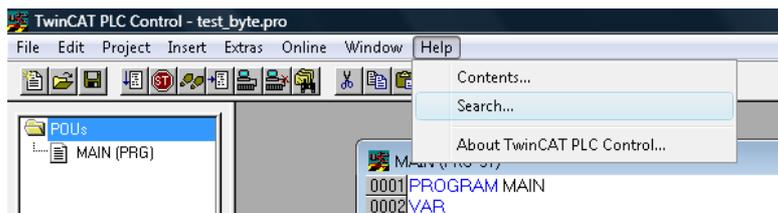
WHILE:

```
WHILE <Boolean expression> DO
    <instruction>
END_WHILE;
```

REPEAT:

```
REPEAT
    <instructions>
UNTIL <Boolean expression>
END_REPEAT;
```

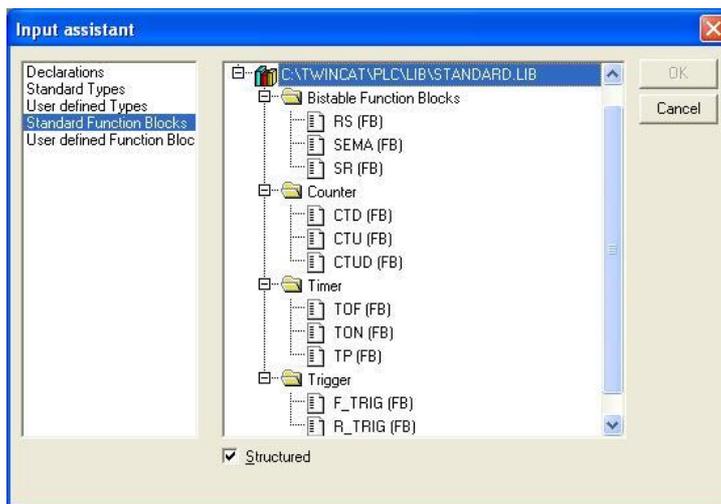
For any other statement necessary or functional examples, it is easy to search it in PLC Control Help.



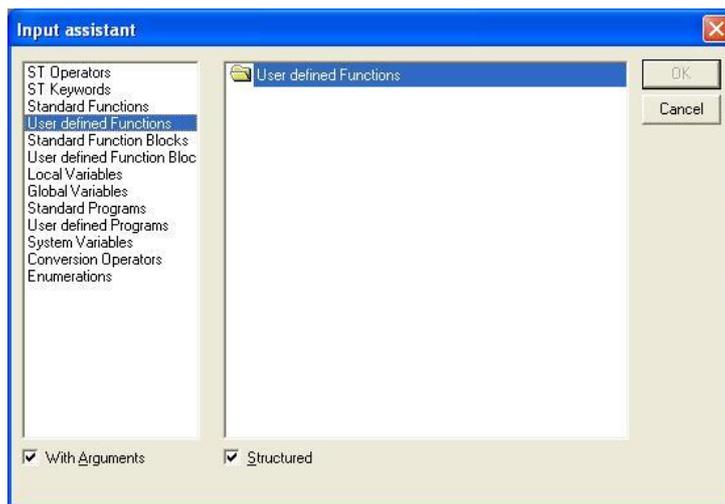
**Figure 37: Help menu**

It is also possible to find in the libraries standard function blocks. This means that timers, triggers, counters and bistable functions are available from the standard library without any necessity to create them on your own.

To use these functions, press 'F2' at any moment inside the window of the program (it doesn't matter if it is in the variable or in the program description side). A new window will appear with these appearances (depends of where F2 is pressed, the window is different), just select 'Standard Function Blocks' on the menu on the left side and then choose between all the available function blocks.



**Figure 38: Library**



**Figure 39: Library**

As you can see in the last window, by clicking 'F2' while the program is being written, it is easy to get access to any variable, function block or anything useful for the program.

Finally, a simple example of a Structured Text program, with a counter which is possible to increase or decrease, and then this counter is converted to a Byte to be sent as a binary number.

```

MAIN (PRG-ST)
0001 PROGRAM MAIN
0002 VAR
0003     increase_button AT %I*:BOOL;
0004     decrement_button AT %I*:BOOL;
0005     signal_output_dig AT %Q* :BYTE;
0006     signal_output_anal AT %Q* :INT;
0007     i: INT := 0;
0008     flag: BOOL;
0009     reset_button: BOOL;
0010
0011 END_VAR
0012
0013 IF (increase_button AND (NOT flag)) THEN
0014     i := i + 1;
0015     flag := TRUE;
0016 ELSIF (decrement_button AND (NOT flag)) THEN
0017     i := (-1)+i;
0018     flag := TRUE;
0019 ELSIF reset_button THEN
0020     i := 0;
0021 ELSIF ((NOT increase_button AND NOT decrement_button) AND flag) THEN
0022     flag := FALSE;
0023 END_IF;
0024
0025 signal_output_anal := i*1000;
0026 signal_output_dig := INT_TO_BYTE (i);
0027
0028

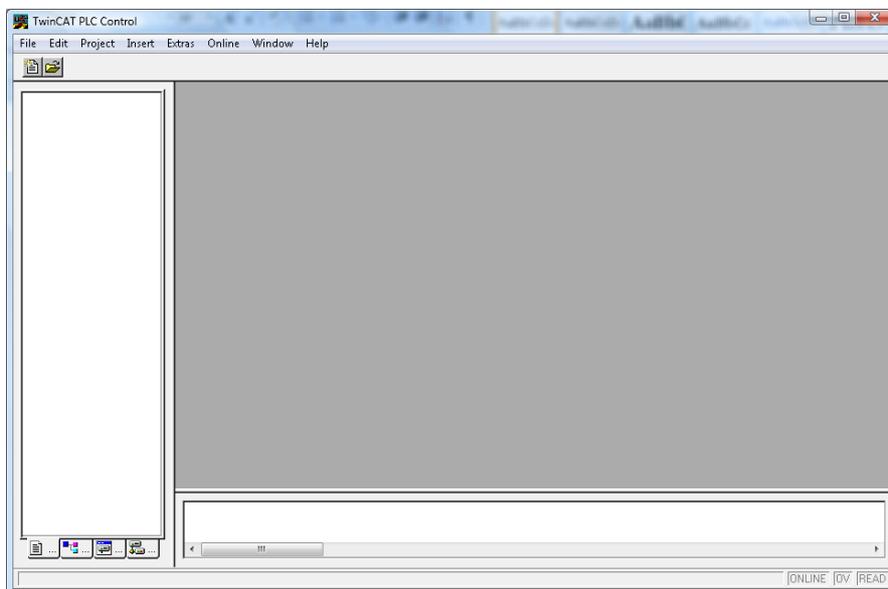
```

**Figure 40: Example of ST program**

## 4.2 TwinCAT

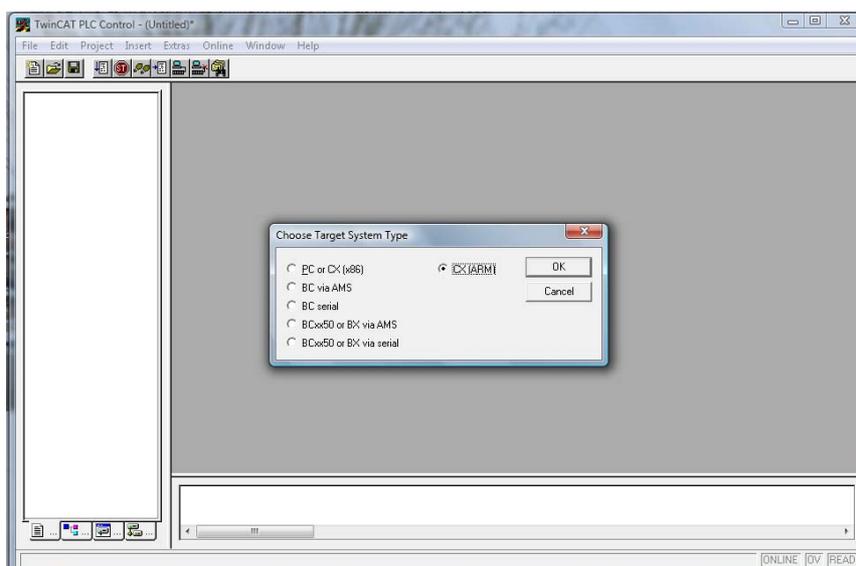
### 4.2.1 TwinCAT PLC control

The application we have used to develop Structured Text programs inside TwinCAT software is TwinCAT PLC Control. Its appearance is:



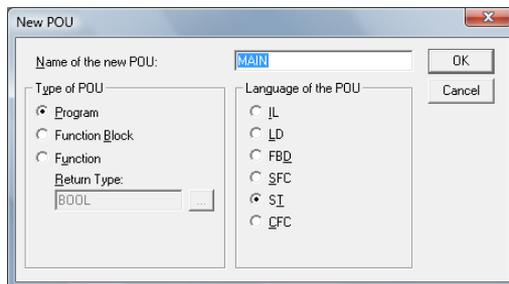
**Figure 41: TwinCAT PLC Control**

To start programming a Structured Text program we select “New” and then we can select which target system type are we going to use. In our case it is “CX (ARM)”.



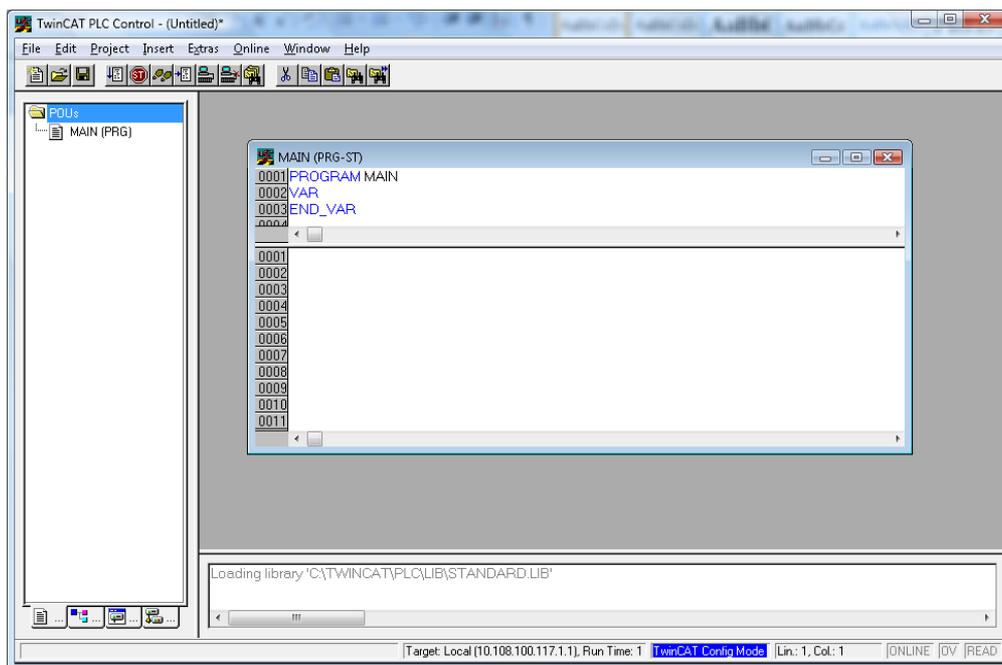
**Figure 42: Choose Target**

Last step before start programming is to choose the type and language of the POU. Structured Text (ST) is the language chosen.



**Figure 43: Choose language**

Finally, we get to the workspace. As we can see, it is very similar to other languages we know. Maybe, the main difference we have is that the declaration of variables is separated from the program description.

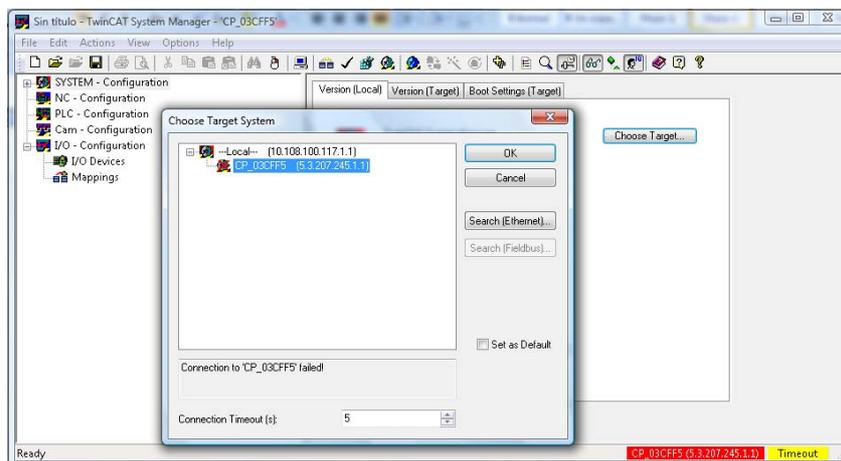


**Figure 44: Starting with the program**

## 4.2.2 TwinCAT system manager

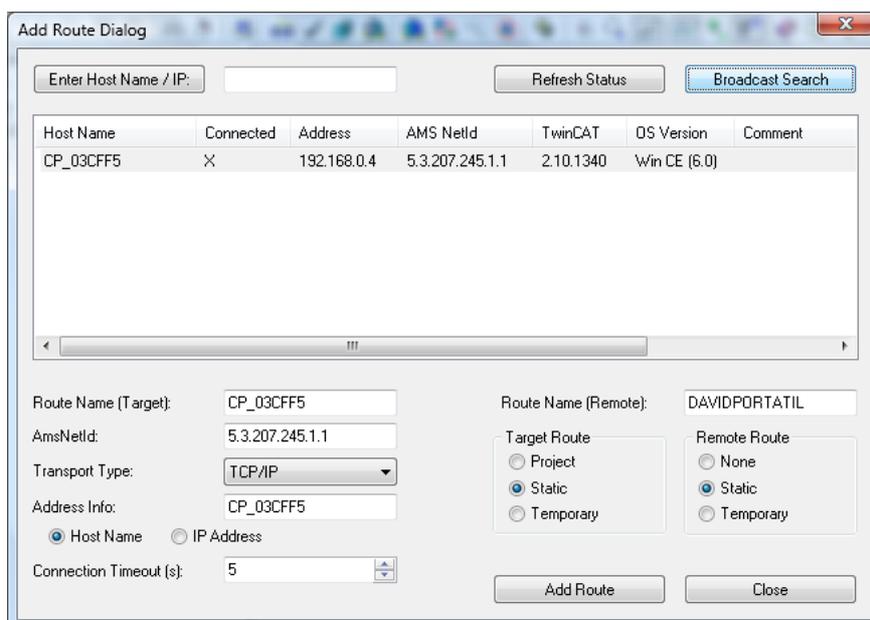
When all previous work is done, it is time to upload the written program to the PLC. To connect our PC with the PLC inside the Touch Panel module, System Manager is the software we need.

Once the program is opened, click on “Choose” Target and then choose “Search (Ethernet)”.



**Figure 45: Choose Target System**

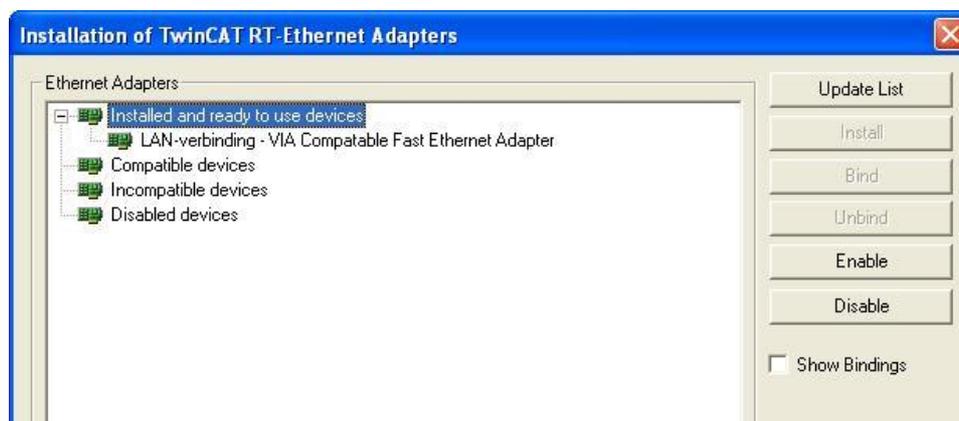
Now, with the Ethernet wire connected to the PC and to the Touch Panel, click on “Broadcast Search” until the Touch Panel appears in the list. Once it appears, click on “Add Route” and the PC will get connected to the Touch Panel.



**Figure 46: Add Route**

At this point, to get connection to the touch panel can take some time. It depends of if it is the first time to connect that computer with the PLC and also the internet IP configuration on the computer and the PLC.

Sometimes, when you try to connect for the first time, there is also a problem just to find the PLC. If the PLC is not visible after several attempts to search it or maybe it appears but the connection is not established, the compatibility of the Ethernet card off the computer should be checked. To check the compatibility, click on “Options” on the top bar and then select “Show Real Time Ethernet Compatible Devices...”. The window shown is this one:

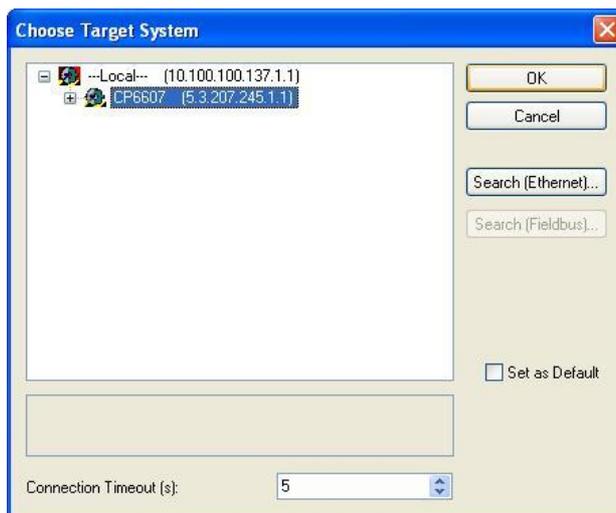


**Figure 47: Installation of TwinCAT RT-Ethernet Adapters**

In case a compatible or incompatible device is found, it would be turn to “installed and ready to use” just selecting it and then, click on “Install” button.

If the computer has been connected to the PLC several times, but it takes some time to connect to it again. Probably is due to the internet protocol (TCP/IP) configuration. If the computer and the PLC are both configured with fixed IP addresses, the time to connect should be shorter. Other way, if the configuration is on automatic IP address, it will take more time, because both devices have to wait until they get their address. Due to this reason, if an Ethernet switch is used between the computer and the PLC, the time to connect could be increased.

Once every occasional problem to get connection with the PLC has been solved, it is the moment to select the PLC as the master device. It is only necessary to select the PLC (CP6607) on the ‘Choose Target System’ window and click “OK”.

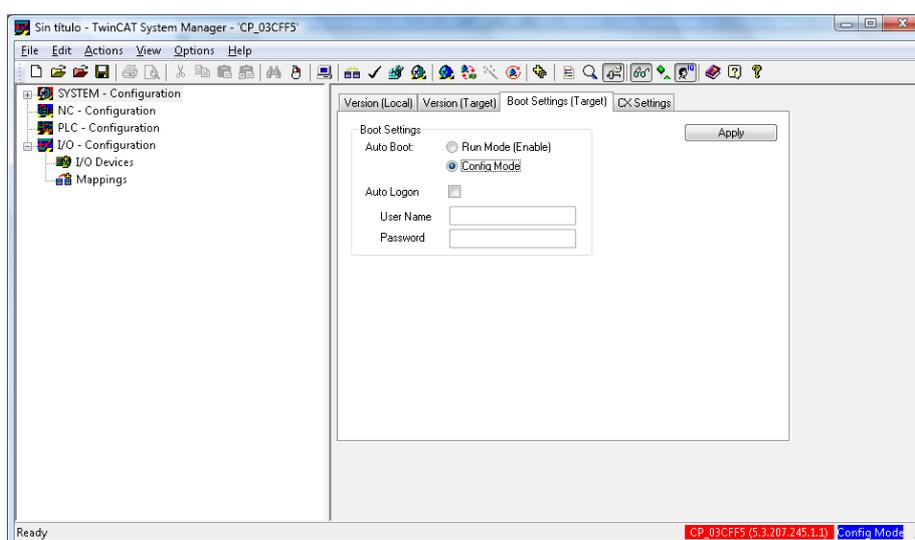


**Figure 48: Choose Target System**

From now, the control device will be the Touch Panel. It is possible to do some configuration about how it should work.

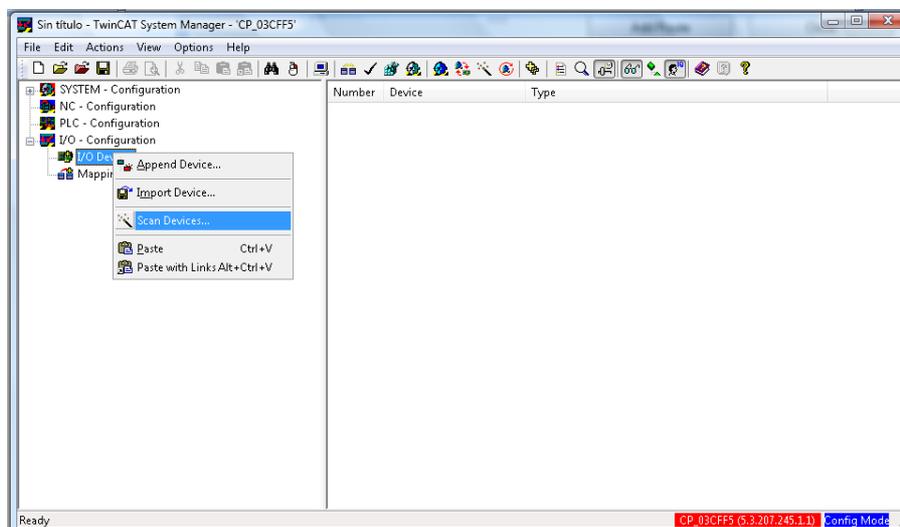
One important setting is to choose if we want the PLC to work in Config Mode or in Run Mode every time the Touch Panel is rebooted. It must be in Config Mode all the time the program is just a test program, and use Run mode only if it is confirmed (by testing it) that the program doesn't have any problem in its working process.

Also, if the intention is to run a program from the reboot of the Touch Panel, is not enough to set Auto Boot in Run Mode. With the program created in PLC Control, it is also necessary to create a Boot Project.



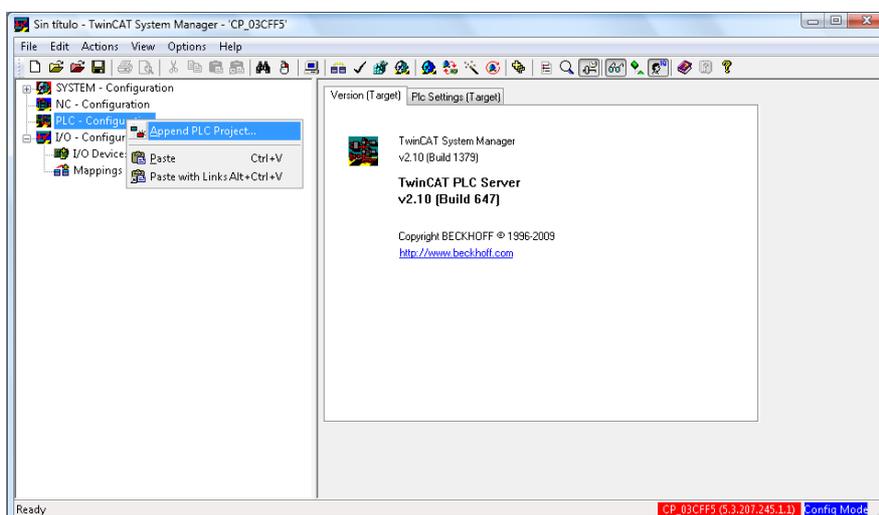
**Figure 49: Boot settings**

When the PLC is connected and all the settings are corrected, it is time to look for all the devices on the network. By clicking on “Scan Devices”, the program searches automatically for all the devices on the network.



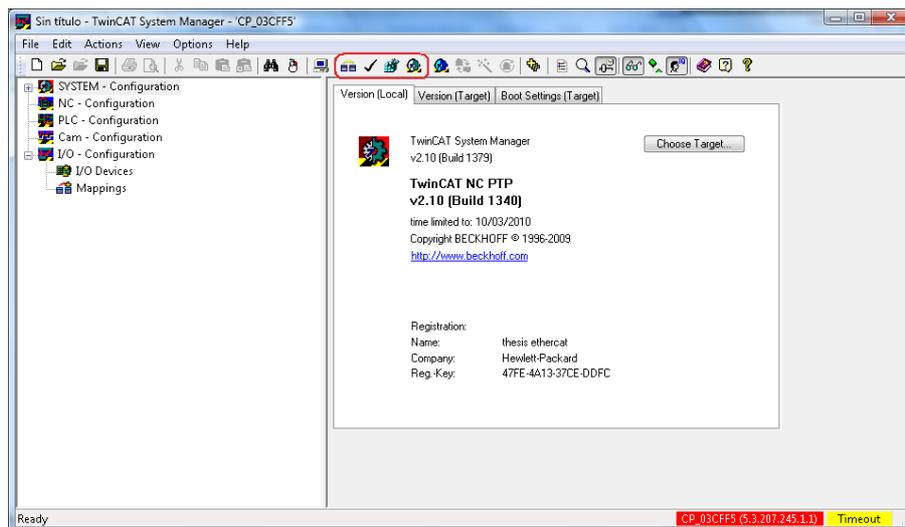
**Figure 50: Scan devices**

One of the latest steps will be to append a PLC project. So, click on “Append PLC Project”, choose the project, and then it will be possible to connect the internal variables to the external devices.



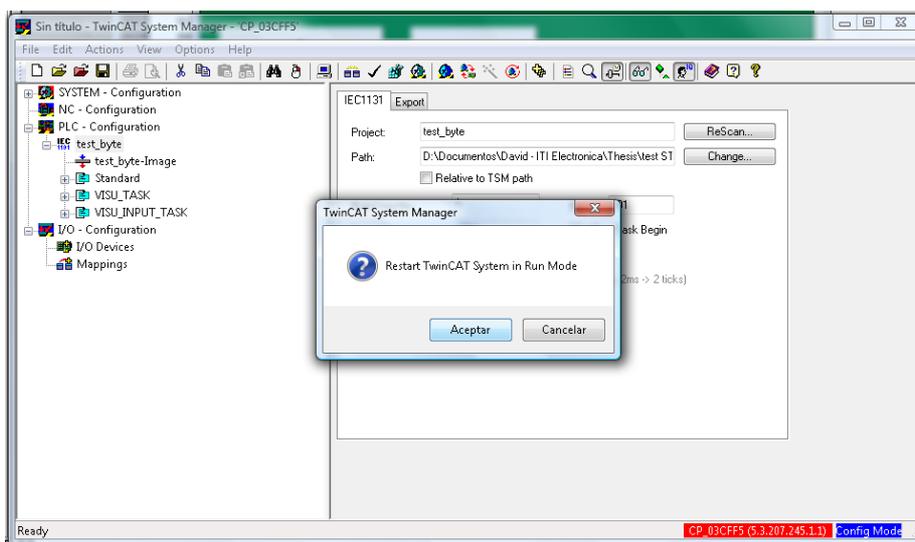
**Figure 51: Append PLC project**

When the variables are configured to be linked, click on the mapping icon (first one in the red rectangle), then it is good to check the configuration (second icon) and finally click on “Activate Configuration” (third icon) and the variables of the PLC project can use data read from Input devices or write into the Output devices.



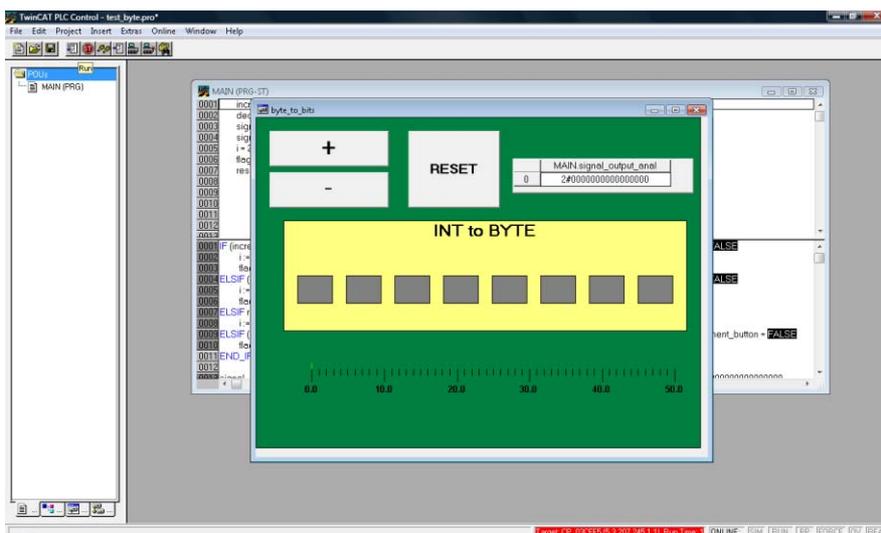
**Figure 52: Activate configuration**

Last step is to restart the TwinCAT system in Run Mode, and the PLC will start working.



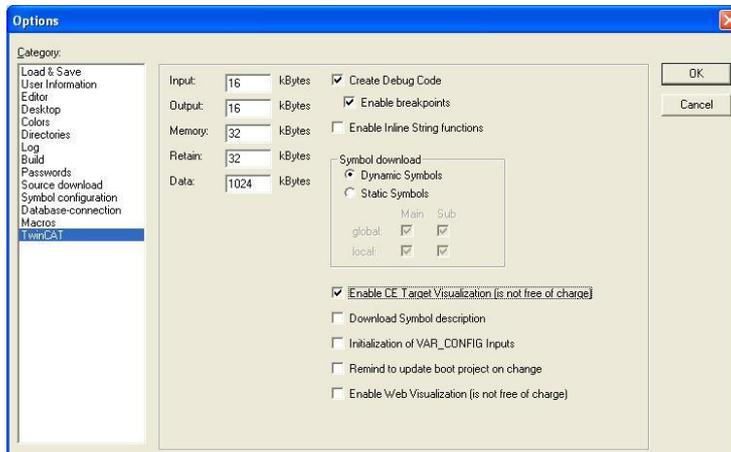
**Figure 53: Restart in Run Mode**

Finally, to run the PLC Project into the PLC, open again PLC Control software with the project which will be used and click the icon to “Login” into the Ethernet network and once it is connected click on “Run” and the program will start to work in the PLC.



**Figure 54: Visualisation**

Notice, if a visualization has been created to use with the program. It will be obligatory to configure it on the PLC Control tool. The option to add the visualization every time we run the program, is located in “Project”, and then select “Options...” and a new window will appear. In that window, select the last option in the menu on the left (TwinCAT) and choose “enable CE Target Visualization”.

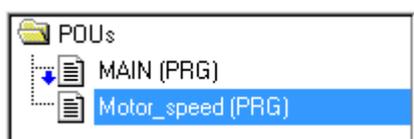


**Figure 55: Options**

### 4.3 The Code

You can find our code in the chapter Appendices (Appendice A). Here in this paragraph we are going to explain only the most important pieces of the code.

We have two separated programs, running independently from each other: 'MAIN' and 'Motor\_speed' as you can see in the figure.



**Figure 56**

The Motor\_speed has a higher priority than the main program because the motor\_speed has to count every pulse coming in from the optical speed sensor. If this program misses one pulse, then the speed of the motor is not correctly calculated.

To calculate the frequency of the motors, we count the pulses from the optical speed sensor during a period of one second. This value is equal to the rotations per second or Hertz.

Because the value from the analog input module of the PLC has a resolution of 12 bits, we first have to convert the value that we read out from the PLC into the corresponding voltage. We do this by multiplying the input value by 0,0003.

$$\text{Voltage (V)} = \text{measured value} * 0.0003$$

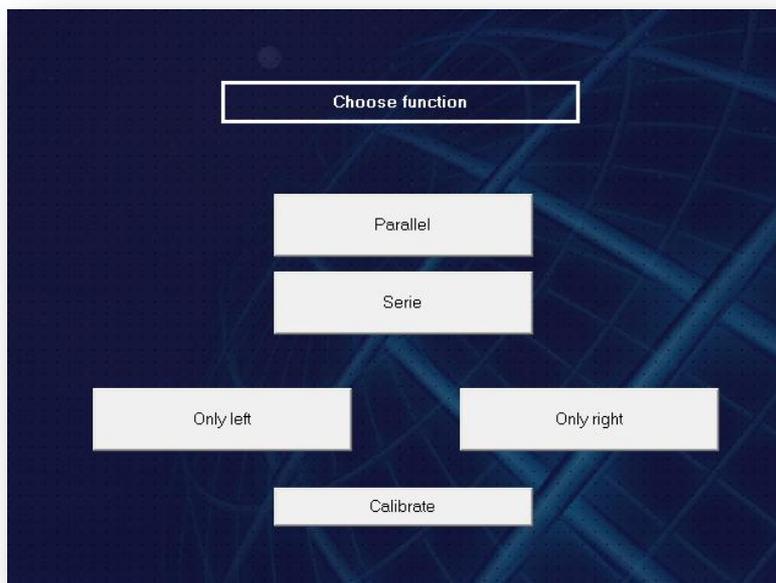
To calculate the temperature, we multiply the measured voltage by  $\frac{100}{6}$ .

$$\text{Temperature (}^{\circ}\text{C)} = \frac{\text{Voltage} * 100}{6}$$

The pressure sensors, we have to calibrate them to get a very precise calculation. We calibrate them by measuring the pressure when there is no pressure difference. To calculate the real pressure, we subtract this calibrated voltage from the measured voltage and then divide it by 0,045.

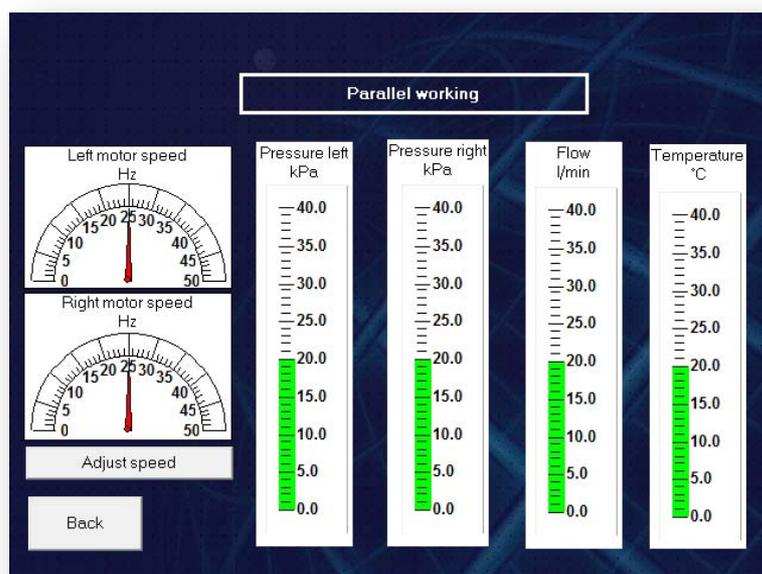
$$\text{Pressure (kPa)} = \frac{\text{measured value} - \text{calibrated value}}{0.045}$$

For the touch panel, we made a user-friendly visualization that looks like this:



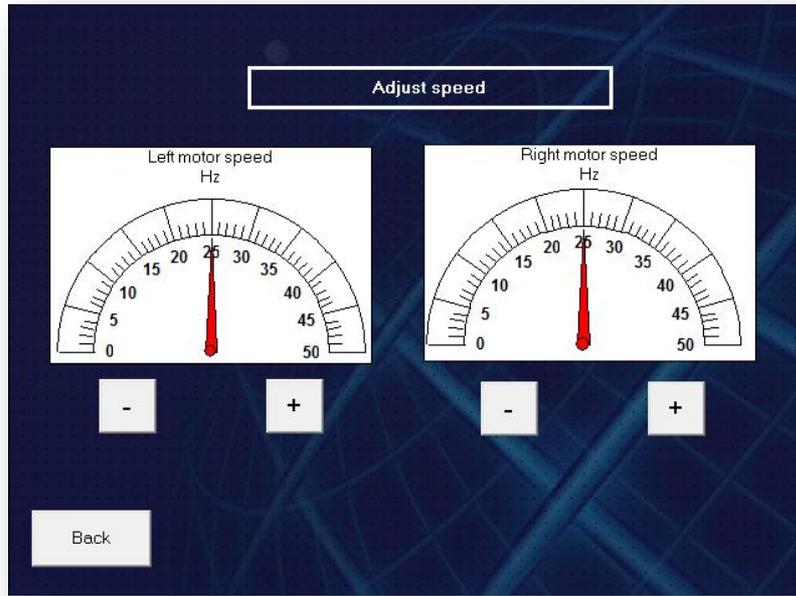
**Figure 57: Main window of the visualisation**

In this main screen you have to choose the set-up of the pumps or you can choose to calibrate the pressure sensors. If you have chosen the right set-up, you get this screen where you can read out every the values of all the sensors. To control the speed, just press the button 'Adjust speed'.



**Figure 58: Monitoring the pumps**

The speed adjust window looks like this:



**Figure 59: Adjust speed of the pumps**

## 5 Conclusions

Basically, the goal of the project was to connect the sensors to the slave devices, which are connected with the PLC through an EtherCAT network. Then, we had to program the PLC using Structured Text language. And finally export the data from the Structured Text program to a LabView application, which would be also a HMI shown on the touch panel.

But, to carry out the project, we had made several changes to the initial approach to avoid the different troubles we had found during these months.

The initial work was to study two separately areas, EtherCAT and Structured Text was one of those, and the other was to get used to the pump installation. Here, the pump installation was the first point where we had to face an issue. The problem was that all Beckhoff devices work with 24V voltage supply, and the sensors of the automation system were ready to work with 5V. The solution we put into practice was to redesign the circuits to make them compatible with the 24V supply.

Another complication which came up with the connection between sensors and slave devices was that there are more analog signals than analog inputs available. To solve it, as we have already explained in this paper, we decided to demultiplex the analog signals into the two analog inputs that we have on the EL3002 device. However, we had also problems with the demultiplexer board because the chips we were using were broken. Once we changed them, the signals became as expected.

Continuing with the communication between slaves and sensors/transducers of the pump installation, we also saw the necessity to find a solution for the wiring. To use individual wires for every connection it is not an option, because the amount of wires would be so high for just 6 or 8 external devices. The solution we found better for our purpose was to install RJ45 connectors on every sensor/transducers box and made in every box the connections it needed. Up in the installation structure, close to the slave devices, we placed a box with the demultiplexer board inside it. In this box we installed, as well, the RJ45 terminals to connect from here to rest of boxes placed over the pump installation. Then, we built our own cables with Ethernet cables and RJ45 connectors. With this solution, the appearance of the installation is improved and also is easier to connect or disconnect any of the elements. However, any people using the installation must not confuse the EtherCAT cables with the cables to connect the sensors/transducers, although their appearances are the same.

In the previous paragraph, when we mentioned transducers, we referred to the stepper motors we use to manage the voltage of the variable transformers. The first option we had to control those stepper motors was a Beckhoff device, the KL2531. This device is a stepper motor terminal (as it has been described before) which should be able to control the stepper motor directly. But finally we did not use it. The reasons for this decision were that the stepper motors we are using work with 12V (instead of the 24V signal we would get from the KL2531) and that we did not find enough information about how it works. The solution we used to control the motors was to use the EtherCAT SPI slave module, developed by Olivier Ottenburgs, as we mentioned before.

Step by step on our project, the next challenge was to get connection between our computer, the touch panel and the slave devices.

Between the touch panel and the slave devices, normally there is not any problem and the touch panel will find the EtherCAT terminal most of the times. But, to get connection between the touch panel (the PLC) and the computer can be difficult sometimes. The easiest and fastest way to get connection is using fix IP on both, the computer and the touch panel. This solution can be useful if there is only one computer connected to the PLC, but, if there are several computers connected it is a better option to configure the computers and the PLC with automatic IP address. One of the consequences of using automatic IP address is that it takes more time to get connection between the computers and the touch panel. Another consequence, and maybe the most important one, is that the PLC is not ready to work with more than one computer (at the same time), so it is easy to cause an error on its operation system and make the PLC get frozen.

When we have done all the previous work, that means, to complete the new circuits, connect them to the slaves and finally get connection all over the network created between the computer, touch panel and slaves. We still found some other troubles.

The intention of this project was to control automatically the pump installation. So, the speed (frequency) of the motor must be correctly measure in order to be sure of how much water is being pumped. At this point, we designed the circuit for the optical speed sensor that had been described before in this paper. After that, we tested it and its output signal was correct when we measured it before it enters the digital input in the EL1008 device. But, when we use that signal in our program, we do not read the correct value. Until this moment, we are still not sure about the real cause. Our theory is that the cycle time used on PLC Control is not fast enough, or maybe, the EtherCAT fieldbus is introducing a delay to the signal.

Finally, we also changed the software used to develop the Human-Machine interface (HMI). At the beginning of the project, an interface from Labview was expected to be used. But a research from our partner, Dieter Vanrykel, concludes that it was not possible to use LabView because of problems with ADS interfacing. There would be a possible solution for it, but this I refer to our partners that worked on it. We still needed to create an interface to use on the touch panel and as we have been working with TwinCAT software, we decided to use the tool inside this software to develop it.

## 6 Appendices

### Appendice A: The Code

#### MAIN

PROGRAM MAIN

VAR

```

    waarde1 AT %I*:INT;
    waarde2 AT %I*:INT;
    mpx AT %Q*:BOOL;
    btn_bck:BOOL;
    cal_val1:LREAL;
    cal_val2:LREAL;
    cal_val3:LREAL;
    temp:LREAL;
    pres1:LREAL;
    pres2:LREAL;
    pres3:LREAL;
    calibrate:BOOL;
    i:INT;
    j:LREAL;

```

```

    btn_left: BOOL;
    inc_left :BOOL;
    dec_left :BOOL;
    b1_left:BOOL;
    b2_left:BOOL;
    b3_left:BOOL;
    b4_left:BOOL;
    a_left AT%QX0.0:BOOL;
    b_left AT%QX0.1:BOOL;
    c_left AT%QX0.2:BOOL;
    d_left AT%QX0.3:BOOL;
    k :INT;
    btn_right: BOOL;
    inc_right :BOOL;
    dec_right :BOOL;
    b1_right:BOOL;
    b2_right:BOOL;
    b3_right:BOOL;
    b4_right:BOOL;
    a_right AT%QX0.0:BOOL;
    b_right AT%QX0.1:BOOL;
    c_right AT%QX0.2:BOOL;
    d_right AT%QX0.3:BOOL;
    l :INT;

```

END\_VAR

IF calibrate = TRUE THEN

IF i < 100 THEN

```

    mpx := TRUE;
    cal_val2 := INT_TO_LREAL(waarde1)*0.0003;
    i := i+1;

```

ELSIF i < 200 THEN

```

    mpx := FALSE;
    cal_val1 := INT_TO_LREAL(waarde1)*0.0003;
    cal_val3 := INT_TO_LREAL(waarde2)*0.0003;

```

```

        i := i+1;
    ELSE
        i := 0;
    END_IF;
ELSE
    IF i < 100 THEN
        mpx := TRUE;
        pres2 := (INT_TO_LREAL(waarde1)*0.0003-cal_val2)/0.045;
        temp := (INT_TO_LREAL(waarde2)*0.0003)*100/6;
        i := i+1;
    ELSIF i < 200 THEN
        mpx := FALSE;
        pres1 := (INT_TO_LREAL(waarde1)*0.0003-cal_val1)/0.045;
        pres3 := (INT_TO_LREAL(waarde2)*0.0003-cal_val3)/0.045;
        i := i+1;
    ELSE
        i := 0;
    END_IF;
END_IF;

IF btn_left THEN
    IF k < 1 THEN
        k := k +1;
    ELSE
        k := 0;
        IF inc_left THEN
            IF (NOT(a_left) AND NOT(b_left) AND NOT(c_left) AND NOT(d_left))
THEN
                b1_left:= TRUE;
                b2_left:= TRUE;
                b3_left:= FALSE;
                b4_left:= FALSE;
            ELSIF (a_left AND b_left AND NOT(c_left) AND NOT(d_left)) THEN
                b1_left:= FALSE;
                b2_left:= TRUE;
                b3_left:= TRUE;
                b4_left:= FALSE;
            ELSIF (NOT(a_left) AND b_left AND c_left AND NOT(d_left)) THEN
                b1_left:= FALSE;
                b2_left:= FALSE;
                b3_left:= TRUE;
                b4_left:= TRUE;
            ELSIF (NOT(a_left) AND NOT(b_left) AND c_left AND d_left) THEN
                b1_left:= TRUE;
                b2_left:= FALSE;
                b3_left:= FALSE;
                b4_left:= TRUE;
            ELSIF (a_left AND NOT(b_left) AND NOT(c_left) AND d_left) THEN
                b1_left:= TRUE;
                b2_left:= TRUE;
                b3_left:= FALSE;
                b4_left:= FALSE;
            ELSE
                b1_left:= FALSE;
                b2_left:= FALSE;
                b3_left:= FALSE;
                b4_left:= FALSE;
            END_IF;
        ELSIF dec_left THEN

```

```

THEN
    IF NOT(a_left) AND NOT(b_left) AND NOT(c_left) AND NOT(d_left)
        b1_left:= TRUE;
        b2_left:= FALSE;
        b3_left:= FALSE;
        b4_left:= TRUE;
    ELSIF a_left AND NOT(b_left) AND NOT(c_left) AND d_left THEN
        b1_left:= FALSE;
        b2_left:= FALSE;
        b3_left:= TRUE;
        b4_left:= TRUE;
    ELSIF NOT(a_left) AND NOT(b_left) AND c_left AND d_left THEN
        b1_left:= FALSE;
        b2_left:= TRUE;
        b3_left:= TRUE;
        b4_left:= FALSE;
    ELSIF NOT(a_left) AND b_left AND c_left AND NOT(d_left) THEN
        b1_left:= TRUE;
        b2_left:= TRUE;
        b3_left:= FALSE;
        b4_left:= FALSE;
    ELSIF a_left AND b_left AND NOT(c_left) AND NOT(d_left) THEN
        b1_left:= TRUE;
        b2_left:= FALSE;
        b3_left:= FALSE;
        b4_left:= TRUE;
    ELSE
        b1_left:= FALSE;
        b2_left:= FALSE;
        b3_left:= FALSE;
        b4_left:= FALSE;
    END_IF;
ELSE
    b1_left:= FALSE;
    b2_left:= FALSE;
    b3_left:= FALSE;
    b4_left:= FALSE;
END_IF;

a_left:=b1_left;
b_left:=b2_left;
c_left:=b3_left;
d_left:=b4_left;
END_IF;
END_IF;

IF btn_right THEN
    IF I < 1 THEN
        I := I +1;
    ELSE
        I := 0;
        IF inc_right THEN
            IF (NOT(a_right) AND NOT(b_right) AND NOT(c_right) AND
NOT(d_right)) THEN
                b1_right:= TRUE;
                b2_right:= TRUE;
                b3_right:= FALSE;
                b4_right:= FALSE;
            ELSIF (a_right AND b_right AND NOT(c_right) AND NOT(d_right))
THEN

```

```

        b1_right:= FALSE;
        b2_right:= TRUE;
        b3_right:= TRUE;
        b4_right:= FALSE;
THEN      ELSIF (NOT(a_right) AND b_right AND c_right AND NOT(d_right))

        b1_right:= FALSE;
        b2_right:= FALSE;
        b3_right:= TRUE;
        b4_right:= TRUE;
THEN      ELSIF (NOT(a_right) AND NOT(b_right) AND c_right AND d_right)

        b1_right:= TRUE;
        b2_right:= FALSE;
        b3_right:= FALSE;
        b4_right:= TRUE;
THEN      ELSIF (a_right AND NOT(b_right) AND NOT(c_right) AND d_right)

        b1_right:= TRUE;
        b2_right:= TRUE;
        b3_right:= FALSE;
        b4_right:= FALSE;
ELSE
        b1_right:= FALSE;
        b2_right:= FALSE;
        b3_right:= FALSE;
        b4_right:= FALSE;
END_IF;
ELSIF dec_right THEN
NOT(d_right) THEN      IF NOT(a_right) AND NOT(b_right) AND NOT(c_right) AND

        b1_right:= TRUE;
        b2_right:= FALSE;
        b3_right:= FALSE;
        b4_right:= TRUE;
        ELSIF a_right AND NOT(b_right) AND NOT(c_right) AND d_right THEN
        b1_right:= FALSE;
        b2_right:= FALSE;
        b3_right:= TRUE;
        b4_right:= TRUE;
        ELSIF NOT(a_right) AND NOT(b_right) AND c_right AND d_right THEN
        b1_right:= FALSE;
        b2_right:= TRUE;
        b3_right:= TRUE;
        b4_right:= FALSE;
        ELSIF NOT(a_right) AND b_right AND c_right AND NOT(d_right) THEN
        b1_right:= TRUE;
        b2_right:= TRUE;
        b3_right:= FALSE;
        b4_right:= FALSE;
        ELSIF a_right AND b_right AND NOT(c_right) AND NOT(d_right) THEN
        b1_right:= TRUE;
        b2_right:= FALSE;
        b3_right:= FALSE;
        b4_right:= TRUE;
ELSE
        b1_right:= FALSE;
        b2_right:= FALSE;
        b3_right:= FALSE;
        b4_right:= FALSE;

```

```

                END_IF;
            ELSE
                b1_right:= FALSE;
                b2_right:= FALSE;
                b3_right:= FALSE;
                b4_right:= FALSE;
            END_IF;

            a_right:=b1_right;
            b_right:=b2_right;
            c_right:=b3_right;
            d_right:=b4_right;
        END_IF;
    END_IF;

```

### **Motor speed**

```
PROGRAM Motor_speed
```

```
VAR
```

```

    dig_in1 AT%I*:BOOL;
    dig_in2 AT%I*:BOOL;
    count1:LREAL;
    count2:LREAL;
    timer1:TP;
    timer2:TP;
    freq1:LREAL;
    freq2:LREAL;
    flank1:R_TRIG;
    flank2:R_TRIG;
    timer: BOOL;

```

```
END_VAR
```

```

timer1(IN:=NOT(timer1.Q),PT:=T#1100ms);
flank1(CLK:=dig_in1);
flank2(CLK:=dig_in2);

```

```

IF(timer1.ET <= T#1000ms)THEN
    IF(flank1.Q) THEN
        count1 := count1 + 1;
    END_IF;
    IF(flank2.Q) THEN
        count2 := count2 + 1;
    END_IF;
ELSE
    freq1:=count1;
    freq2:=count2;
END_IF;

```

```

IF(NOT(timer1.Q)) THEN
    count1 := 0;
    count2 := 0;
END_IF;

```

## 7 Literature and references

Pump installation:

[www.armfield.com](http://www.armfield.com)

Structured Text:

[www.eod.gvsu.edu/~jackh/books/PLCs/chapters/PLC\\_st.pdf](http://www.eod.gvsu.edu/~jackh/books/PLCs/chapters/PLC_st.pdf)

EtherCAT:

[www.EtherCAT.org/](http://www.EtherCAT.org/)

<http://en.wikipedia.org/wiki/EtherCAT>

[www.EtherCAT.org/pdf/english/pcc0107\\_safety\\_over\\_EtherCAT\\_e.pdf](http://www.EtherCAT.org/pdf/english/pcc0107_safety_over_EtherCAT_e.pdf)

Beckhoff

[www.beckhoff.com/](http://www.beckhoff.com/)