OULU UNIVERSITY OF
APPLIED SCIENCES

Heikki Niemelä

ROBOT CONTROL WITH RASPBERRY PI

ROBOT CONTROL WITH RASPBERRY PI

Heikki Niemelä
Bachelor's thesis
Spring 2016
Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Information Technology

Author: Heikki Niemelä
Title of the bachelor's thesis: Robot Control with Raspberry Pi
Supervisor: Timo Vainio
Term and year of completion: Spring 2016 Number of pages: 31

The objective of this bachelor's thesis was to test if it would be possible to create a robot control system that controls each robot individually. The hardware used in testing was Raspberry Pi, another aim for this thesis was to find industrial hardware which could replace Raspberry Pi in the actual application. The used PLC program was CoDeSys since a cheap license for Raspberry Pi was recently released. This kind of solution could make developing robots easier and would greatly assist when many robots would be combined.

The test platform was built using Ginolis master and motor cards. These cards were connected to Raspberry. With two motor cards a test conveyer and a single EC flat motor was controlled. When this physical built was tested also virtual axes were added to the software. In this way it was found out the maximum number of motors that Raspberry Pi could control. Also, a hardware research for replacing Raspberry Pi was done. This research was completed by searching products from the Internet and by negotiating with SKS Group.

The research pointed out that Raspberry Pi works as a robot controlling PC. It was able to control 20 motors at the same time with a reasonable jitter. As for the research for hardware, there was available industrial hardware with similar capabilities to Raspberry Pi. Unfortunately, the price of all available solutions were over the budget of this thesis.

Keywords: Robotics, Automation, Raspberry Pi, PLC, Distributed control

PREFACE

The work was commissioned by Ginolis Ltd. The company provided a working environment and all the boards used in thesis. The work took place in the production facility of Ginolis.

From the company side the supervisor of the thesis was Jarmo Paloviita, Manager of the robot applications. He provided a great assistance with understanding current codes and solutions.

A special thanks has been rightfully gained by all the coworkers in Ginolis Ltd. This thesis was made possible by them and their explanations and teachings of their products and equipment.

Timo Vainio assisted as a tutoring teacher.

Oulu, 12.4.2015
Heikki Niemelä

CONTENTS

VOCABULARY

AND	Logic gate that gives a TRUE output only if it gets two TRUE inputs.
Array	Programming term for space where multiple variables are stored.
Topology	A way of building a network.
Interface	Graphical part of program for a user to be seen.
Variable	A slot that a program uses to save and read data from.
RJ-45	Commonly understood as a physical layer of Ethernet.
Cycle time	Time between starting of each loop.
Microcontroller	Small component with calculation capabilities.
Case structure	Programming structure which uses a certain part of program according to a changing number.
Bit	Term used for data that is one or zero.
Operational state	PLC devices use different states to indicate what is happening in a device. This state means that a device is ready to be used.
Library	In information technology there is a group of code that can be easily called.
Boolean	Variable that contains values TRUE or FALSE.
DWord	Variable that contains 32-bits which are divided into four groups.
Else if	Program structure that runs a certain part of a program if a certain condition is fulfilled.

Jitter	Difference between the wanted time and the actual time.
Oscilloscope	Device used to measure signals.
Ethernet	The most common networking standard used in local area networking.
C++	Programming language.
TCP/IP	Common networking protocol widely used in the Internet browsing.
Skript	Program that is executed in another program.

1 INTRODUCTION

Ginolis Ltd. Is a company that offers automated solutions to the production of diagnostics equipment. The strength of Ginolis automated solutions lay in high precision and modularity (1).

The object of this thesis was to test if the current robot control would be possible to replace with a CoDeSys solution. If the current TwinCat solution were pointed out to be replaceable, the CoDeSys solution would be tested with the application. The purpose of this testing was to create a system that offers a control for each robot individually instead of a single control PC for each production line.

A word robotic means mechanical devices that are programmed to do simple tasks. Nowdays robots are met even in households doing simple tasks such as vacuuming and lawn mowing. While household robots are rather new, sight robots have been doing simple tasks for decades in the field of automated productions. As told earlier these tasks are simple, such as a screw fixing or lifting objects. In these kinds of tasks robots are ideal since they do not get tired or frustrated while repeating simple but necessary tasks. Robots also have many other advantages compared to humans. They have a superior accuracy, speed and strength. These attributes give a possibility to use robots in huge amounts of tasks. And robots are constantly being developed more. So the amount of tasks that they can complete grows constantly larger.

If this thesis would point out that this concept works, the products of the company would gain an even greater modularity since each module would work with or without another module.

The challenges of this thesis lay in a real time function of the Raspberry Pi. Another challenge was an application which did not exist at the point when the purpose of the thesis was decided.

2 DEFINATION OF USED THCHNOLOGY

In this section used technologies and equipment are introduced. In order to build a working test platform, it was necessary to get familiar with these technologies.

2.1 Raspberry Pi

Raspberry Pi is a single board PC, which is usually used for educational purposes. Usually Raspberry Pi has a Linux based operation system. Overall Raspberry is claimed to do any task that a usual desktop PC is capable of doing (2.)

Raspberry has five different models and all available models are comparedwith each other in table below (Table 1). On the compartment a notable model is Raspberry Pi Zero, which is physically half the size of others. And it meant for projects that need to save space.

TABLE 1. Compartment of different Raspberry pi models (3).

Model	CPU	Wireless	Ethernet	Bluetoot	Ram	GPIO pins	Camera interface
Raspberry Pi 3 Model B	1.2GHz 64-bit quad-core ARMv8	Yes	1 Slot	Yes	1 GB	40	Yes
Raspberry Pi 2 Model B	**900MHz quad-core ARM Cortex-A7**	**No**	**1 Slot**	**No**	**1 GB**	**40**	**Yes**
Raspberry Pi 1 Model B+	700 MHz 7ARM116JZF-S	No	1 Slot	No	512MB	40	Yes
Raspberry Pi 1 Model A+	701 MHz 7ARM116JZF-S	No	No	No	256MB	40	Yes
Raspberry Pi Zero	1Ghz, Single-core	No	No	No	512MB	HAT 40	No

Raspberry Pi is not meant for industrial purposes, since usually an industrial PC is designed to cooperate in much harsher conditions and has a higher tolerance towards dirt, heat and cold. In this particular case Raspberry Pi was used in the test case scenario and it was never meant to be used in the actual production.

This platform was chosen to be a test platform because of its cheap price and because CoDeSys had recently released a very cheap license for Raspberry Pi.

The used version was a Raspberry Pi 2 model B. This version was chosen because at the beginning of the project its processor was the most powerful. Although Raspberry Pi 3 model B has a better processor, it was released after the testing in this thesis was started.

The used Raspberry version has a 900Mhz quad core ARM Cortex-A7 processor and 1 GB RAM. It also has 4 USB ports, one Ethernet port and a micro SD card slot (4). Naturally, the model B has all the same other connections as the previous versions have but they were not needed or used during this thesis.

2.2 Programmable Logic Controller (PLC)

PLC is a computer control system, which constantly measures input devices and makes decisions for output devises according to the written program (5.) PLC is a widely used system in terms of automated production.

The basic idea of PLC is that all the inputs and outputs are connected to the central processing unit. At the beginning the system measures input values. Then, the system runs a customized program made for that particular case. After that the system manages the output according to the program and results gained from the inputs. At the last state the program does housekeeping tasks such as communicating with devices and diagnosing hardware.

PLC programming can contain five different languages. The programming languages are a structure text (ST), a ladder diagram (LD), a function block diagram (FBD), an instruction list (IL) and a sequential function chart (SFC). These languages can roughly be divided into two groups, a written language and a graphical language. The structure text and instruction list are written langages. The ladder diagram, function block and sequential function charts are graphical languages.

During this thesis only the structure text, ladder diagram and function block diagram were used. The ladder and function block diagrams were used at the beginning of this thesis while testing the inputs and outputs of implementation. These codes had no effect on the result of this thesis. However, the structure text was more widely used and all the used programs were written using the structure text.

The ladder diagram is usually used in simple cases as shown in the example (figure 1). This example turns a digital output to be true if digital inputs 2 and 3 are true. It is also required that the AND function is enabled with the digital input 1. timer function is also used in the example. This turns the digital output 2 to be true after one second when the digital input 2 is turned true.

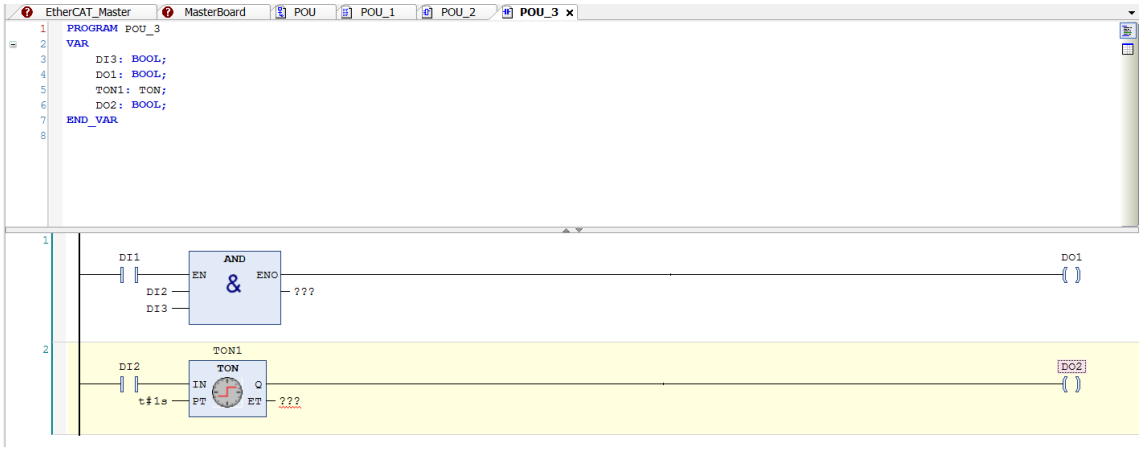

FIGURE 1. Example of a ladder diagram

The function block diagram is usually used to create different kinds of functions that are used in the whole PLC application. In the shown example (figure 2) the function is created. This function detects a rising edge from the digital input 1 and sets the variable 1 to be true after one second from the detection of the rising edge.

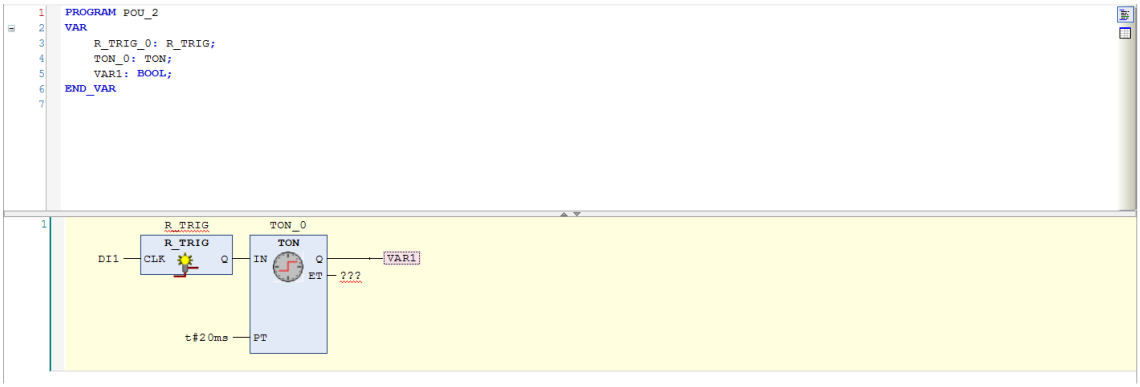

FIGURE 2. Example of a function block diagram

The most important language in this thesis was the structured text. It is the most used language because compared to other languages it allows more complex programs to be created. As shown in the example (figure 3), the function is exactly the same as in the ladder diagram example but the structure text allows output data to be stored in an array for a later use.

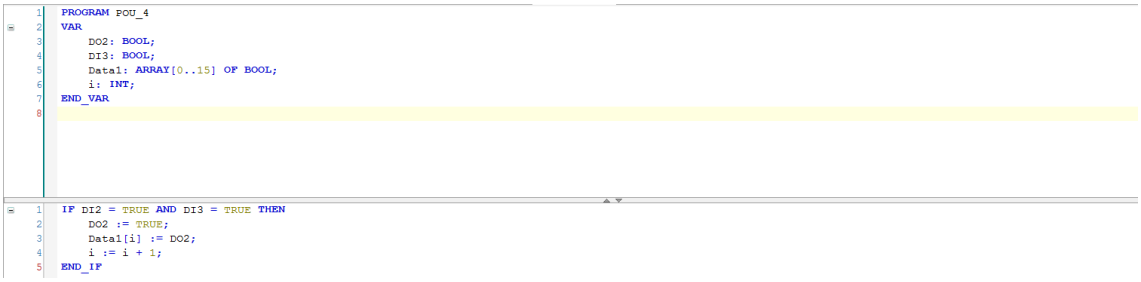

FIGURE 3. Example of a structure text

2.2.1 EtherCat

A company called Beckhoff started a project to develop EtherCat in 2004. Thus, Ethercat Technology Group was created. This croup is separated from Beckhoff but it does a really close cooperation with Beckhoff. EtherCat Technology Group has all the rights to the EtherCat.

EtherCat is a network protocol, which aims for a fast rate messaging. EtherCat could be defined to be flexible since it gives a possibility to connect multiple de-

vices to the network with a superb topology. Possible topologies are a string topology and a ring topology. The strength of these two EtherCat topologies lays in their self-establishing and flexibility.

EtherCat devices are connected to each other through RJ-45 ports. With two RJ-45 ports in each device devices can be connected to a string (figure 4). And messages are conveyed from device to another.

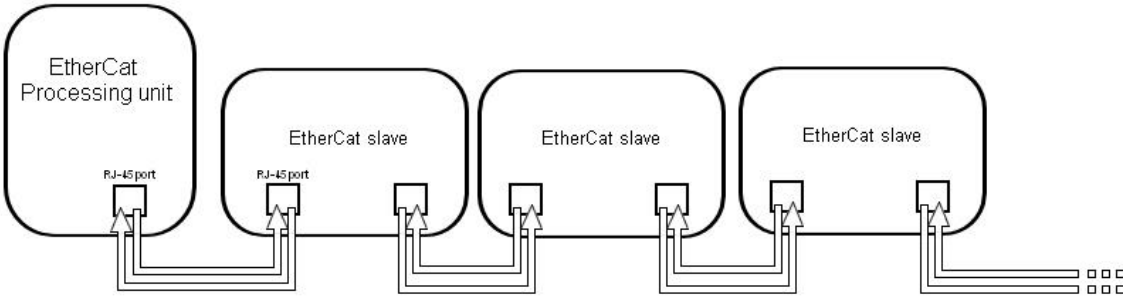

FIGURE 4 EtherCat string structure

Devices will initialize themselves when the system is started. A somehow unique feature in EtherCat is that when last the device does not detect the next device it will automatically terminates the network. The last device will also send messages back to the master through the same RJ-45 post as it received messages.

EtherCat also provides a possibility to the ring topology (figure 5). In the ring structure messages are always relayed forward until message reaches a processing unit.

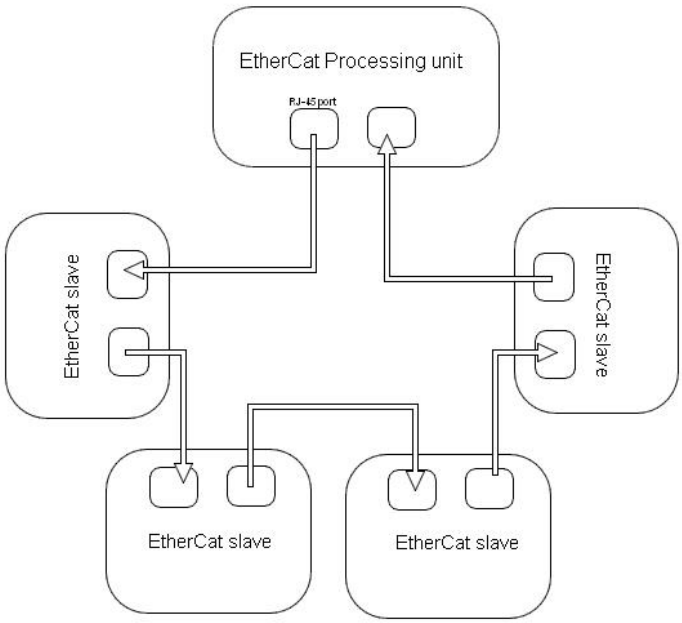

FIGURE 5. EtherCat ring structure

The strength of the ring structure lays in its fault tolerance. If a cable is broken or one of slave Ethernet ports are multifunctioning the processing unit will detect the problem and it will send messages to the slaves. In these messages there is information which creates two separate string structures and system itself can continue operating (6).

2.2.2 CoDeSys

CoDeSys is a software platform, which is targeting to sell software to the industrial automation companies and projects. This platform uses the PLC programming language and it offers several useful functionalities. The most important functionalities in this thesis are visualization and soft motion. Visualization is a system that creates a user interface which is visible to the user through a screen. And Soft motion is system that allows creation of axes. Without the soft motion calculation to run motor smoothly should be written in the program.

This section concentrates on the usage of CoDeSys, so that it could be easier to understand what was done during this thesis.

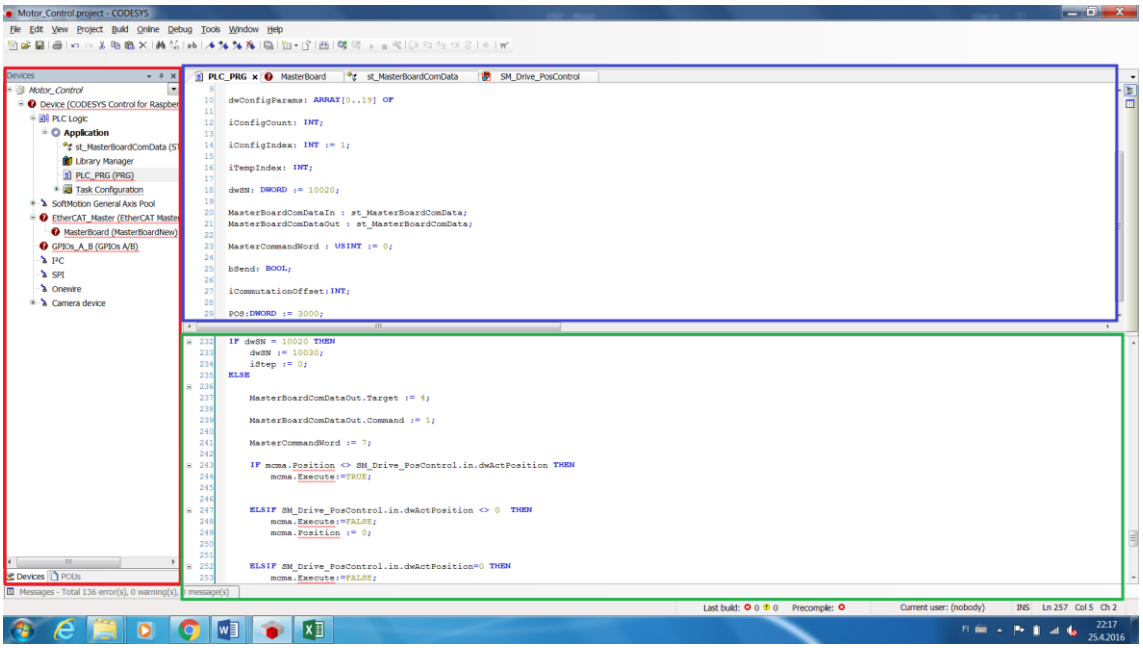

FIGURE 6. Basic view of CoDeSys

On the left side of the figure (figure 6) circled in red is a device tree. On this device tree there are all the used devices, programs and the used functionalities of CoDeSys. For instance, there is a soft motion general pool, under this pool all the used axes are defined. Thus, on this device tree it is possible to freely add or remove new devices, programs and functions. New devices can also be added by a scan function. This function will automatically scan all the connected devices and add them to the device tree.

On top of the figure circled in blue is a variable table. On this table all the used variables on the current program are defined. These variables are possible to link to any device but if two programs use the same variable, then variables need to be defined in global variables the under device tree.

Finally, at the bottom of the figure circled in green is a program table. This section contains the actual program selected from the device tree. On the view program there is the structure text but it could be as well any other available programming form. At the same time different kinds of programming languages can be used. All the wanted languages can be defined to the device tree.

2.3 Ginolis Master board

The ginolis master board is designed and produced by Ginolis Ltd. The master board is used as an EtherCat master. During this thesis only few connections of the master board was used. In reality the master board has ten digital outputs and ten digital inputs. These outputs and inputs are usually used to control sensors and to control simple devices like lights. In each input slot there is also ground and power pins for each sensor and each output slot contains also a ground pin. These pins exist to make wiring more simple for each product.

The master board also contains communication ports for different kinds of control cards. Each master board can give commands to ten different cards at the same time.

2.4 Conveyer

In this thesis a conveyer was used as an application (figure 7). This conveyer belt is out of usage and has already been replaced with a newer kind of technology but it is well suitable for the test application.

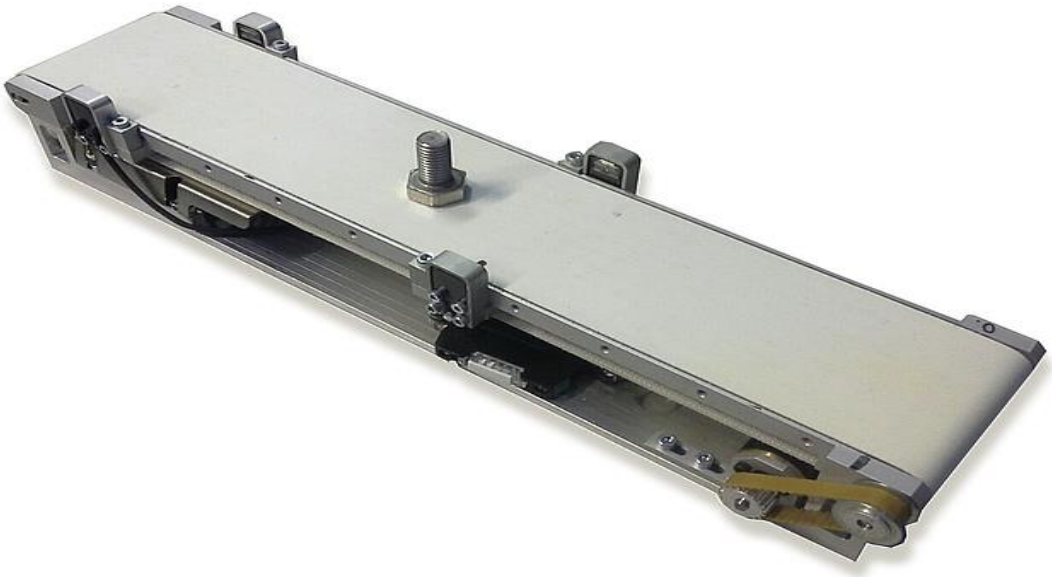

FIGURE 7. Conveyer

This conveyer contains the Maxon EC-max motor and two Omron E3T-FT13 sensors used for sensing motion on the conveyer. The bolt visible in the figure is a random object, which was used during testing and its only purpose is to be transported.

At the beginning of the thesis the implementation was planned to be completed with a cutter application. Due to mechanical problems, this cutter applications production was delayed and this conveyer was chosen to replace the actual application.

3 IMPLEMENTATION

This section contains a description about how the test platform was built. To achieve a functional test platform, the following steps were needed. Firstly, a physical platform was build. This build was really simple because it was built just and only for test purposes. After the basic connection between devices was built they needed the basic communication between each others. This communication was achieved with premade code for a PLC processing unit and with premade code for the Ginolis master and motor cards.

3.1 Physical platform

Raspberry Pi that contained CoDeSys was used as a PLC processing unit. Raspberry was connected to the Ginolis master card with an RJ-45 cable. Wiring from the Ginolis master card to the Ginolis motor card was done as pointed out in the Ginolis motor card datasheet. The motor was wired to the Ginolis motor card as pointed out in the datasheet of motor. This used motor was a Maxon EC-max motor.

After the basic communication was achieved and the first test was made, another Ginolis motor card was connected with the Ginolis master card. This motor card was used to run the conveyor, which was chosen to represent the application. All the connections were done with an exact method as previously mentioned. The only difference was two sensors from the conveyor. These sensors were connected to Ginolis master cards input slots. Current and ground were wired from the same slot because the card makes this kind of connection possible.

3.2 Programming

The used languages in this thesis were PLC and C++. PLC code was made for CoDeSys and C++ was used for both Ginolis cards. During this thesis the actual program writing was, however, a minor part. The main efforts with codes lay in understanding premade codes. Understanding was necessary since codes were not directly possible to plot into the system and needed slight changes.

Another reason for understanding was data sending. It was necessary to know which kind of data had to be sent so that Ginolis cards would function properly.

3.2.1 Motor and master card

Code for both Ginolis master and motor cards already existed since both cards are constantly in use. In normal usage Ginolis cards have a default cycle time of 2ms. This cycle time is defined in code so if needed, this cycle time can be changed. Changing a cycle time to be longer would rise an idle time of processor and so it could be able to calculate more axes. Since cards use PIC controllers, a PICkit was needed to program the cards. During this thesis the actual programming was done with the PICkit 3. PIC itself is a microcontroller, which contains a small memory for program to be saved in. And PICkits are programming devices which allow a program to be written in this small memory of PIC.

Ginolis master cards code itself was unchanged. However, for a card to be able to function, code was written in it with the PICkit. Another necessary part was XML file, which contains information for the card about its self. This information basically tells the card what are its functions and how the card is called. This information makes a connection to the card possible.

With the Ginolis motor card code was written with the PICkit just as with the master card. The only difference was that code had to be written twice to the card. At the first writing round an ID definition was enabled. This ID is unique to each card and is necessary since it is used to recognise which card each message is meant to. At the second time when writing the code, the ID definition was disabled and the actual program was taken in use.

3.2.2 Starting motor with CoDeSys

At the beginning the Ginolis master board was taught to CoDeSys. This was done by adding a specific masterboard.xml file to a list of known devices. After this the XML file master board was added to device tree of this project. After the master board had been added to the project, all necessary connections were

mapped to the master board. In other words, the program was taught that data in a certain variable is sent out from certain output. A similar task was done to all outputs and inputs in usage.

From this point on working was done more like traditionally recognized coding. All the coding was completed in the structured text, because other available languages were not useful with this kind of application. This also means that necessary changes in CoDeSys were done and the following code will command the Ginolis motor card.

For making the basic connection from Raspberry Pi to the Ginolis master board, premade code was available and used. This code was implemented with a case structure. This code works by running a case after a case so that at the end of each case the number, which chooses cases, is increased by one. Unless one case does not function as it should the case sets the choosing number to be 999. This means that the program itself is in an error state. As this was used in a test case, the error state did not do anything else than informed that something went wrong.

At the first case parameters for the Ginolis motor card were defined. The defined values were an ID number, amplifier values, used currents and a used pulse width. These values depend on the controlled motor. Amplifier values adjust the motor so that the actual movement is equivalent to the desired movements. The used currents values define current that is allowed to the motor so that the motor will not break down. Lastly, the pulse width is defined so that the motor has enough time to execute all the orders sent to it. At the second case the message of incoming parameters was sent to the motor card. At the third case parameter values were sent. The sent data is in two parts. The first part is three bursts of eight bits containing the ID of the receiving devise. The second part is four bursts of eight bits containing one defined parameter. At the third case it was checked that all the parameter values were sent and the message for the end of parameters was sent. After the third case has made sure that all the messages have been sent, the program is directed to a case number fifty.

This means that all the devices have reached an operational state and the basic communication between devices has been achieved.

After getting all the devices to the operational state, the motor was defined in CoDeSys. Unlike the master board, the motor card does not need a specific XML file. Controlling motors is a universal system in CoDeSys thus all the motors are controlled in the same way and the difference between motors affects only to the previously sent parameter values.

A position controlled drive was defined under an SoftMotion axis pool in the device tree. Without this definition of new axis, all the functions of the motor ought to have been programmed manually. But now all the calculation such as acceleration, deceleration and position was done in this axis. This made programming easier since now the motor control was done just by changing variable values.

Under the defined axis there is large amount of different libraries and all the libraries contain a lot of control variables. However, to get the motor working, only two libraries were needed. These libraries were Power and MoveAbsolute. The power library is a library where different functions of the motor are enabled. Without the usage of this library, the motor would not gain any current to run. The moveAbsolute library is a library where values for running the motor are defined.

In the actual program libraries were defined in variable sections. At the definition the beginning values of variables were also defined. Since the program is rather simple, Boolean variables enable bRegulatorOn and bDriveStart were defined true under the Power library. This means that the motor itself was able to run from the start of the program. Also, the MoveAbsolute library was defined. Under this library the defined variables were Execute, Acceleration, Deceleration, Velocity and Position. The only notable variable is the Execute variable, which is a Boolean variable, and it was defined as false because changes to the true motor take action according to other defined variables. DWord variables Acceleration, Deceleration, Velocity and Position were given some values since at this point they did not matter so much as the purpose was only to get

the motor running. These variables were changes later when the system was tested and a more accurate control was needed.

After everything above was completed and the program ran till the end, the motor was in a ready state and all what had to be done was manually force the Execute variable as true. This caused the motor to run in a different position using the defined variables.

3.2.3 Test program

To test Raspberry Pi's feasibility, it was decided to run two motors and a couple of sensors at the same time. This was done to confirm that controlling a physical motor was equivalent to controlling a virtual motor. This test platform was achieved by adding a conveyer to the build and by repeating everything from the chapter Starting a motor with CoDeSys. The only exception was that also two sensors had to be mapped to the Ginolis master card. In this way also the conveyer was set to a ready state.

To run two motors while using two sensors, a short program was written under a case 50 (figure 8). This program causes another motor to run till the position reaches 3000. When in that position, the motor returns to a position 0. Another motor runs forward until the sensor detects an object and then the motor changes direction until another sensor detects the same object.

```
243     IF mcma.Position <> SM_Drive_PosControl.in.dwActPosition THEN
244         mcma.Execute:=TRUE;
245
246
247     ELSIF SM_Drive_PosControl.in.dwActPosition <> 0  THEN
248         mcma.Execute:=FALSE;
249         mcma.Position := 0;
250
251
252     ELSIF SM_Drive_PosControl.in.dwActPosition=0 THEN
253         mcma.Execute:=FALSE;
254         mcma.Position := POS;
255     END_IF
256
257
258     IF mcma_1.Position <> SM_Drive_PosControl_1.in.dwActPosition THEN
259         mcma_1.Execute:=TRUE;
260     END_IF
261
262     IF DI1 = FALSE THEN
263         mcma_1.Execute:=FALSE;
264         mcma_1.Position := 0;
265     END_IF
266     IF DI2 = FALSE THEN
267         mcma_1.Execute:=FALSE;
268         mcma_1.Position :=5000;
269     END_IF
```

FIGURE 8. Program used to run motors.

The used variables in this code are Position, which is a desired position for the motor, and dwActPosition, which is the actual positon of the motor. This value updates as the motor changes its position. Also, the variable Execute was used to apply changes done to the desired position. These variables are saved to a different variable so that multiple motors could be controlled. These variables are named as mcma and mcp. The difference between motors are seen from a numbering system. When mcma is given values, the motor is controlled and when the changed values are under mcma_1, the motor conveyer is controlled.

Code itself firstly checks if the actual position and the desired position are different. If they are different code sets the motor to move to the desired position. Then the actual position of motors is checked with else if structure. If motor has reached its desired position, the desired position is changed to the opposite

end. Then on the next round of the loop, the first if case of the program, executes this change.

With conveyer codes, the basic working policy is exactly the same. The motor runs one direction until it reaches the desired position. The point desired position is changed and executed at the next round of loop. The only difference is that the running direction of the motor changes when the sensor value changes from true to false. This change means that sensor the has detected an object.

With this executed two motors started running separately. One motor ran back and forth between two points set with the variable. The conveyer started to move an object back and forth on it. At this point everything was set to start the actual measurements and test.

4 TESTING THE IMPLEMENTATION

The main focus of tests was on the processor usage since it was assumed to be the main reason why Raspberry Pi would not suit to this kind of robot control. Another test subject was jitter which told how much deviation there were in messaging times.

The processor usage was tested in two cases. The first case was the visual side of CoDeSys. This test was done by simply enabling or disabling visual while inspecting the processor usage on Raspberry Pi. This test was completed while two motors were running and without them.

Another case was to find out how many axes Raspberry was able to run. This was done by creating virtual axes. These virtual axes were linked to receive their positions and desired positions from the motor. In this way each virtual axis had to complete exactly same calculations as the axis that ran the motor. After each added axis, the program was executed and the processor usage was checked from Raspberry Pi. Also, the functioning of the motors was visually inspected since it was expected that if the motor would skip any steps due to lack of processor power, it could be possible to detect with a plain eye.

The last test subject was jitter. Jitter is showed directly from CoDeSys. But to make sure that this data was reliable, an oscilloscope was used. To get any data for the oscilloscope, one of Ginolis master cards outputs was set to change its value at each round of loop. When this data was supplied to the oscilloscope, an output should look like a square wave. And from the irregularity of the square wave, an actual jitter was possible to measure. This measurement, however, pointed out that data shown by CoDeSys is accurate.

4.1 Test results

Processor testing was started by starting CoDeSys and checking the processor usage level. It was confirmed that having just CoDeSys running but without doing anything, the processor usage level was 17% from the available capacity.

When the visual side of CoDeSys was started, the processor usage level raised up to 30%. This value however was not stabile and after inspecting the usage level, it was possible to say that the visual side uses approximately 10% of the available processor capacity.

After adding virtual axes on the program, the results from processor usage per axis were written down (Table 2).

TABLE 2. Processor usage with multiple axes

Motors	CPU%	Notes
0	17	OK
2	33	OK
3	36	OK
4	39	OK
5	41	OK
6	44	OK
7	46	OK
10	52	OK
20	68	OK
21	80	Slight Lag
22	82	Slight Lag
23	84	Lag
24	85	Lag

When zero motors were running 17% of the processor capacity was used. This is the same as CoDeSys were just idling. When two motors were started, the processor usage raised to 33%. This large raise is due to all the basic calculations from running CoDeSys since after this adding, one axis caused a just slight increase of the processor usage.

When 20 axes were running, the processor usage level was 68% and there were no problems. However, after adding the 21st axis, the processor usage level took a large leap up to 80% and the motor visibly started skipping steps. Although there was supposed to be still idle calculation power left it would seem that either Raspberry's information about the processor usage lever is inaccurate or Raspberry is actually not able effectively to use all its resources with CoDeSys.

After adding the 23rd axis, skipping steps were very clear, since these skips were so long that the motor completely stopped moving and soon started moving again. This caused acceleration and declaration to increase near infinity and from these sudden stops and starts the motor generated enough power to start move around.

4.2 Research for alternative hardware

Since Raspberry Pi is not suited for industrial purposes, a research for alternative hardware was done. The target for this research was to find out hardware whose capitalises are similar to Raspberry Pi but which would suit for industrial purposes. This means that hardware should comprehend dirt, heat and cold. Hardware should also operate without any fan which would make hardware silent. The last requirement was two Ethernet ports, which were required for EtherCat to reach its full functionality.

This research is also supposed to find out the lowest price for a robot control with CoDeSys. The price was an essential factor while choosing hardware, since it would be unreasonable to control each motor with the same price as the whole production line could be controlled. This would lead to a raise of the price of products of Ginolis Ltd.

From the beginning two choices were possible. It was possible to design and manufacture Linux based hardware just for this case. Or it was possible to order hardware from an external manufacturer. The first choice was to negotiate with an external manufacturer since ordering a hardware would save time and resources. These negations could also give a price range for the CoDeSys licence.

The research was started by searching suitable manufacturers from the Internet. This searching pointed out that there are plenty of single board platforms available for this kind of system. However most of the single board platforms within price range of 50-200€ do not have the required amount of Ethernet ports. Also while researching these single board platforms, the suitability for industrial purposes was unknown.

Another choice for what could be ordered was industrial PCs as it was expected that a PC produced for industrial purposes would be more expensive. But it was necessary to find out a price range of industrial PCs whit the CoDeSys licence. Since Ginolis Ltd. has been previously a customer of SKS Group, it was decided to send request of order for an industrial PC to the representor of SKS Group.

The answer for the request of order pointed out that SKS Group produces suitable industrial PCs. These PCs have a similar capability as Raspberry Pi as well as other required features such as two Ethernet ports and a real time function. However, the price of this kind of hardware was more expensive for a single motor control (7).

Shortly after the request of order, a possibility to have a meeting with the representor of SKS Group occurred. This meeting was hold since it was assumed that the representor had more knowledge of the CoDeSys licence policy. This information would give a price range for the license as possible manufactured hardware.

This meeting pointed out that the price of the CoDeSys licence is always negotiated separately with CoDeSys. Since the price for a license is determined by the price of the manufactured product. So the price of a licence for a high cost robot is higher than the current TwinCat licence. But at the same time licence price could be lower for low cost robots. The biggest set back from this meeting was plotting costs. Since the policy of CoDeSys is that when licence to new kind hardware is sold a representor of CoDeSys needs to plot a program to hardware (8).

5 CHALLENGES

5.1 Raspberry Pi

The biggest challenge with Raspberry Pi was a lack of Ethernet ports since there is only one Ethernet port in Raspberry. With the PLC implementation two Ethernet ports are needed since one Ethernet port is used by the EtherCat protocol. So to actually connect Raspberry Pi and the program, CoDeSys TCP/IP connection was needed. This problem was solved by implementing a WLAN module to the Raspberry Pi.

Another challenge, which needed a lot of research, was a fact that Raspberry Pi is not suitable for industrial solutions. Thus so a dilemma was ready when hardware similar with Raspberry was needed. Yet this similar hardware was supposed to be meant for industrial purposes. This research was made even more difficult with a demand that the price of new hardware was supposed to be similar to Raspberry Pi.

5.2 Programming

Programs were written with two languages. All the cards were already written with C++. And the actual code was written as PLC because PLC is a language that CoDeSys requires. This combination of two different languages caused some problems because codes in cards are not short skripts. And when two different languages were wanted to communicate with each other a lot of studying was necessary so that the author could understand what information the cards needed to function or even send an answer message to the Raspberry Pi.

5.2.1 Motor and master card

The codes for Ginolis mastercard and Ginolis motor card already existed and they were used in this implementation. However, to use and implement these codes some changes were needed since TwinCat revives the sent information in a slightly different way than CoDeSys. Another problem with Ginolis cards were the lack of datasheets. All the details of the functions of the cards seemed

to be gained only by testing or by asking from someone who had worked with the cards longer a period of times.

5.2.2 CodeSys code

A base of the PLC code was copied from TwinCat PC. These codes were designed to connect the Ginolis master board to TwinCat. So with small changes code was plotted to the CoDeSys project. Although this seems a simple job, it caused problems since the author did not have any experience with PLC coding.

6 CONCLUSION

The aim of this thesis was to test Raspberry Pi's capability of robot control. The combination of CoDeSys and Raspberry Pi could lead to individual robot control for each robot in the production line. The inspiration for thesis came from a very cheap license CoDeSys for Raspberry Pi.

A test application which simulates the actual application was built with a conveyer and a separate single motor. This kind of application was proven to be possible to run with Raspberry Pi. Although due to the lack of real time functionality of Raspberry Pi, this application had a 100µs jitter. This jitter would most likely be much lower with another hardware.

With the running application the maximum amount of running axes was also tested. This was simply done by adding virtual axes and using them in calculations. Raspberry Pi ran without any problems until the 21st virtual axis was added. This means that Raspberry Pi can run calculations for 20 axes at the same time, which is enough axes for most of the applications. Over all, the result of this thesis is that Raspberry Pi or similar hardware is able to control a single robot.

While choosing usable hardware for the application it was found out that any reasonable solution would exceed the budget plan for this thesis. If a company did not have an existing solution for robot control, this kind of solution would work well. Most likely the best solution could be to design and produce own hardware with similar capabilities to Raspberry Pi 2 module B. Also, the CoDeSys license is required. This licence and plotting costs should be negotiated directly with CoDeSys. In this way costs of hardware would be possible to reduce to a level where each robot is controlled separately.

If a company has been recently established or a company is planned to be established, the solution of this thesis should be taken under a careful consideration, since plotting costs that prevented testing CoDeSys with another hardware would eventually be divided between the produced robots.

It could have been really interesting and rewarding to actually create a new kind of robot control system, which would lower production costs. But the decision of not to forcefully continue developing the system despite the costs, was the most reasonable decision to do. Although, this result is a little bit bothering, it is a really good lesson of making reasonable decisions.

As a learning process completing this thesis has been very rewarding because before this thesis, the author did not have any experience with robotics. Basically, the only familiar part was a PIC controller and everything else had to be studied or experimented. Completing this thesis also gave an idea of the usefulness of robotics. And naturally this led to a great eagerness to work with robotics.

REFERENCES

1. Ginolis. 2015. Date of retrieval 16.2.2016

http://ginolis.com/.

2. Raspberry Pi. Date of retrieval 12.4.2016

https://www.raspberrypi.org/help/what-is-a-raspberry-pi/.

3. Raspberry Pi. Date of retrieval 27.4.2016

https://www.raspberrypi.org/products/.

4. Raspberry Pi 2 Model B. Date of retrieval 12.4.2016

https://www.raspberrypi.org/products/raspberry-pi-2-model-b/.

5. PLC introduction. AMCI. Date of retrieval 12.4.2016

http://www.amci.com/tutorials/tutorials-what-is-programmable-logic-controller.asp.

6. EtherCat. Date of retrieval 13.4.2016

http://www.rtaautomation.com/technologies/ethercat/.

7. Saarikko, Pekka 2016. Offer 3668091. Email message. Receiver: Heikki Niemelä Date of retrieval 19.2.2016

8. Saarikko, Pekka 2016. Sales manager. Oulunsalo. Meeting 23.2.2016