Dublin Institute of Technology

School Of Electrical & Electronic Engineering

Hochschule Merseburg

University of Applied Sciences

Bachelor of Engineering Technology: Electrical and Control Engineering (DT009)

2014/15

Communications between Programmable Logic Computer and Vision Sensors with working Robotic Arm and Human Machine Interface

Student Names: **Andrew East**

Project start date: **5th March 2015**

Project end date: **26th May 2015**

Declaration

I hereby certify that the material, which is submitted in this thesis/project, is entirely our own work and has not been submitted for any academic assessment other than as part fulfilment of the assessment procedures for the program Bachelor of Engineering Technology in Electrical and Control Engineering (BEng Tech) (DT009).

Signature of Students:

_____ _____

Date: _____

Abstract

In our Control and Automation project we used a Mitsubishi RV-E2 robotic arm, controlled by a CR-E116 controller box, to move sixteen different puzzle tiles. On each tile was four different pictures of either a boat, cross, tree or volcano in any order. The goal was to match each picture to the same picture on a different tile in a 4x4 sequence. Each piece was moved within the robot's working area to different predetermined positions. We used a 'teach in' function on a P6TB-TE pendant that came with the robotic arm to save these positions to the robot's memory.

The tiles first position was held in front of an object recognition sensor, IFM O2D220, supplied to us by IFM. The tile was placed down and the arm moved out of view of the sensor. The sensor was triggered when the arm moved out of position by an output bit from the CR-E116 program. It then read the contours of the tile by matching the pictures to a previously saved application. . When triggered the sensor sent a string to a Siemens Simatic S7-1200 PLC. The string was cut down and the relevant information was converted into an integer. This integer was saved to an internal data register and the robotic arm was then signalled to move on to the next location.

The tiles next position was held in front of a multicode reader, used as a QR code scanner, also supplied to us by IFM. On the back of each puzzle tile was a different QR code numbered 01-16, representing the first tile position to the last tile position. When the tile was moved into position by the robotic arm, the sensor (IFM O2I100) was triggered by a second output bit of the program. It then sent another string to the PLC which was again cut down and converted into another integer.

The integers from both sensors were then compared and if it was a match, the tile was successfully identified. The identified tile was then picked back up by the robotic arm and moved to the correct position in the 4x4 puzzle. The robotic arm would then run this cycle another fifteen times until all the tiles were placed in the correct order and the puzzle was complete.

Introduction

The goal of this thesis is to complete a working machine which is capable of solving a puzzle. No matter what order the puzzle tiles are stacked, the machine should have the ability to determine the correct final positions.

We hope to achieve completion of this build with the use of sensors given to us by IFM. IFM is a large international company, stemming from Germany, which has been pioneering the automation industry with quality made products since it was founded in 1969. They specialise in sensors and monitoring systems.

Our project will incorporate various manufacturer's products interlinking using transmission control protocol (TCP). TCP enables two hosts to establish a connection and exchange streams of data. TCP guarantees delivery of data and also guarantees that packets will be delivered in the same order in which they were sent.[1]

To programme the Siemens S7-1200 we plan to use TIA Portal. This software made by Siemens holds itself as the key to unlocking the full potential of Totally Integrated Automation (TIA). The pioneering engineering framework optimizes all planning, machine and process procedures and offers a standardized and integrated operating concept. It seamlessly integrates controllers, distributed I/O, HMI, drives, motion control and motor management into a single engineering environment.[2] It is our hopes that this is true and we successfully manufacture a machine capable of our objective.

[1] "What is Transmission Control Protocol (TCP)? Webopedia." 2002. 28 April 2014 <http://www.webopedia.com/TERM/T/TCP.html>

[2] "TIA Portal - Totally Integrated Automation Portal - The Future ..." 2010. 28 April 2014 <http://www.industry.siemens.com/topics/global/en/tia-portal/pages/default.aspx>

Table of Content

Chapter 1 Project Objectives & Background 1

1.1 Concise Statement of Project Objectives 1

1.2 Detailed Assessment of the Objectives 1

1.3 Component information 2

1.3.1 IFM O2D220 2

1.3.2 IFM O2I100 4

1.3.3 Siemens Simatic S7-1200 5

1.3.4 Siemens Touch HMI KTP600 Basic Colour PN 6

1.3.5 Mitsubishi RV-E2 Arm & CR-E116 7

1.3.6 The Puzzle 8

Chapter 2 Familiarisation with the TIA Portal 9

2.1 Ladder 9

2.1.1 Traffic Light Example (LAD) 9

2.2 Function Block Diagram 11

2.2.1 Car Wash Example (FBD) 12

2.3 Structured Control Language 17

2.3.1 Car Park Example (SCL) 17

Chapter 3 Mitsubishi Movemaster RV-E2 Arm & CR-E116 Contol Box 19

3.1 Introduction 19

3.1.1 Remote 19

3.1.2 Program (COSIPROG) 19

3.1.3 Commands 20

3.2 Laser Test 22

3.3 Suction Head Test 22

3.4 Programming Each Position 23

Chapter 4 IFM Sensors **24**

4.1 IFM O2I100 Identification Sensor **24**

4.1.1 Creating QR Codes 24

4.1.2 Software 24

4.1.3 Connecting Hardware and Software 26

4.1.4 Configuration 30

4.1.5 Test 32

4.2 IFM O2D220 Object Recognition Sensor **33**

4.2.1 Software 33

4.2.2 Connecting Hardware and Software 35

4.2.3 Scanning The Contours Of Each Tile 36

4.2.4 Test 38

Chapter 5 Interlinking All Components **39**

5.1 IFM O2I100, IFM O2D220 & Siemens S7-1200 **39**

5.1.1 Installation 39

5.1.2 Programming 40

5.2 Siemens S7-1200 & Mitsubishi RV-E2 **43**

5.3 Siemens S7-1200 & Siemens Touch HMI KTP-600 **46**

5.4 Conclusion **48**

Bibliography / References **49**

List of Figures

Figure 1	**Side View IFM O2D220**	**2**
Figure 2	**O2D220 Elevations**	**2**
Figure 3	**O2D220 Range & Depth Information**	**3**
Figure 4	**O2I100 Photo & Example**	**4**
Figure 5	**O2I100 Elevations**	**4**
Figure 6	**Siemens Simatic S7-1200**	**5**
Figure 7	**PLC Configuration in TIA Portal**	**5**
Figure 8	**Siemens Touch HMI**	**6**
Figure 9	**HMI Configuration in TIA Portal**	**6**
Figure 10	**Mitsubishi RV-E2 Robotic Arm**	**7**
Figure 11	**Pneumatic Suction Head**	**7**
Figure 12	**The 16 Piece Puzzle**	**8**
Figure 13	**Traffic Light Example Used**	**9**
Figure 14a	**Ladder Logic Example**	**10**
Figure 14b	**Ladder Logic Example**	**11**
Figure 15	**Car Wash Example Used**	**12**
Figure 16	**HMI Used with Car Wash Example**	**12**
Figure 17a	**Function Block Diagram Example**	**13**
Figure 17b	**Function Block Diagram Example**	**14**
Figure 17c	**Function Block Diagram Example**	**15**
Figure 17d	**Function Block Diagram Example**	**16**
Figure 17e	**Function Block Diagram Example**	**17**
Figure 18	**Car Park Example Used**	**18**
Figure 19	**Structured Control Language Example**	**18**
Figure 20	**P6TB-TE Teach Pendant**	**19**
Figure 21	**RV-E2 Joint Jog Operation**	**19**
Figure 22	**COSIPROG Program Example**	**20**
Figure 23	**Examples Of Commands Being Used**	**20**
Figure 24	**Mitsubishi RV-E2 Laser Test**	**22**
Figure 25	**Final Position List**	**23**
Figure 26	***(http://goqr.me/) QR Code Generator Website***	**24**
Figure 27	**Basic User Inerterface (Dualis)**	**25**
Figure 28	**Monitor Mode**	**26**
Figure 29	**Operating Guide "Dualis Multicode Reader O2I100"**	**27**
Figure 30	**Setting Ethernet Connection**	**28**
Figure 31	**Connections**	**28**
Figure 32a	**Option 1 - Bookmarks**	**29**

Figure 32b	**Option 2 - Enter IP**	**29**
Figure 32c	**Option 3 - Search IP**	**29**
Figure 33	**Configuration Page**	**30**
Figure 34	**QR Code Passing A Test**	**32**
Figure 35	**Basic Interface (E2D200)**	**34**
Figure 36	**Operating Instructions (O2D220)**	**35**
Figure 37	**Model Definition Of A Tile**	**36**
Figure 38	**Test Model & Set Parameters**	**37**
Figure 39	**Final Test Match**	**38**
Figure 40	**Test Set Up**	**38**
Figure 41	**Profinet Interface**	**39**
Figure 42	**O2I100 Function Block**	**40**
Figure 43	**O2D220 Function Block**	**40**
Figure 44	**Function Blocks In TIA Portal**	**40**
Figure 45	**Char to String SCL & FB O2I100 Example**	**41**
Figure 46	**Char to String SCL O2D220 Example**	**42**
Figure 47	**S7-1200 Wiring Diagram**	**43**
Figure 48	**CR-E116 Controller Box Connection Block**	**43**
Figure 49	**Matching The Outputs**	**44**
Figure 50	**Final COSIPROG Program**	**45**
Figure 51	**HMI Pages**	**46**
Figure 52	**HMI KTP-600 in TIA Portal**	**47**

List of Tables

Table 1	**Command Types**	**21**
Table 2	**Interface Elements & Contents (Dualis)**	**25**
Table 3	**E2D200 Toolbar Items**	**33**
Table 4	**Interface Elements & Contents (E2D220)**	**34**

Chapter 1 Project Objectives & Background

1.1 Concise Statement of the Project Objectives

The primary objective of our Control and Automation project is to interlink various components to assemble a difficult puzzle. It consists of a Siemens Simatic S7-1200, a Siemens Touch HMI, an IFM O2D220 (02D/RPKG/K) sensor for contour scanning, an IFM O2I100 (02/RO5-G/D/RS232) sensor for QR code scanning and a Mitsubishi CR-E116 control box and RV-E2 robotic arm.

The robotic arm will lift each of the sixteen puzzle tiles and redirect them to each camera. Both cameras will read each card, send the relevant information to the PLC and then the PLC will direct the robotic arm to where the tile must be placed.

1.2 Detailed Assessment of the Objectives

The overall objective of the project is for the Mitsubishi RV-E2 to complete the puzzle correctly. We believe this can be achieved if we:

1. Scan the contours on each of the cards with the IFM O2D220.
2. Create individual QR codes and attaching them to the reverse of each card.
3. Configure the IFM O2I100 to read the QR codes
4. Link both sensors to the Siemens Simatic S7-1200 via RJ45.
5. Create or source SCL code to convert the relevant information, i.e. converting chars to strings.
6. Create a program for the Mitsubishi RV-E2 which can be directed by the Siemens Simatic S7-1200 to the correct positions of the puzzle.

1.3 Component Information

1.3.1 IFM O2D220 Object Recognition Sensor

Figure 1 - Side view IFM O2D220

The IFM O2D220 (02D/RPKG/K) is a sensor that we are going to use to check each tile of the puzzle to identify where it should be placed to rebuild it in the correct order. The sensor can identify the contours of each tile when scanned and saved in its applications.
It has many industrial applications including quality control in which each object being checked is scanned for any imperfections.

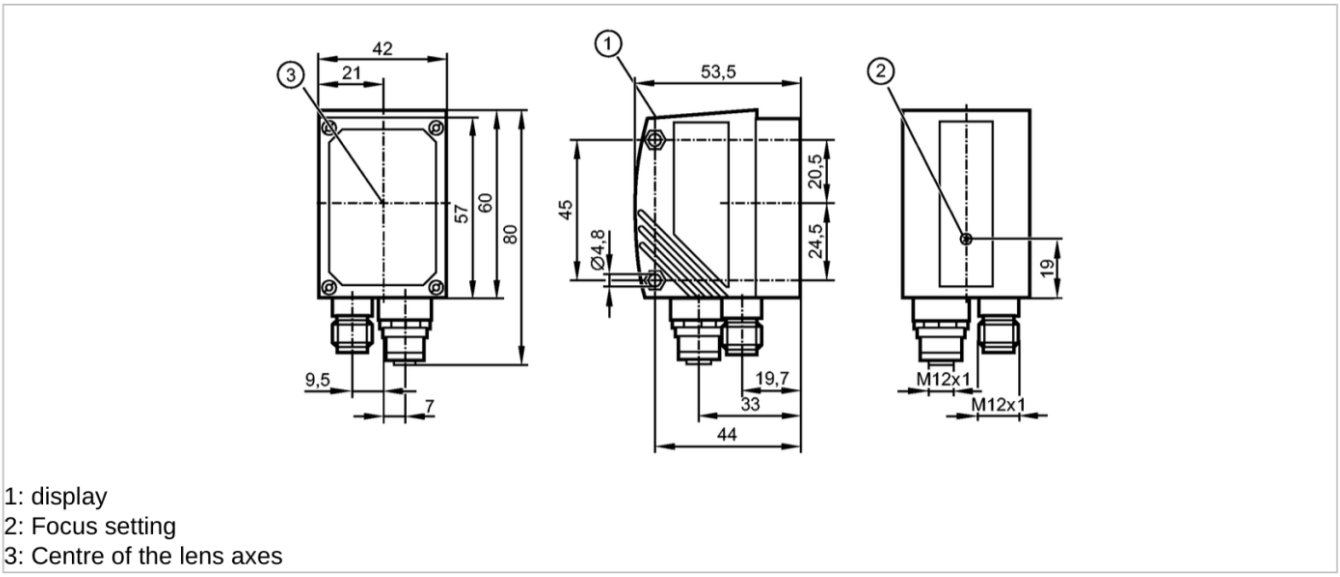


Figure 2 - O2D220 Elevations

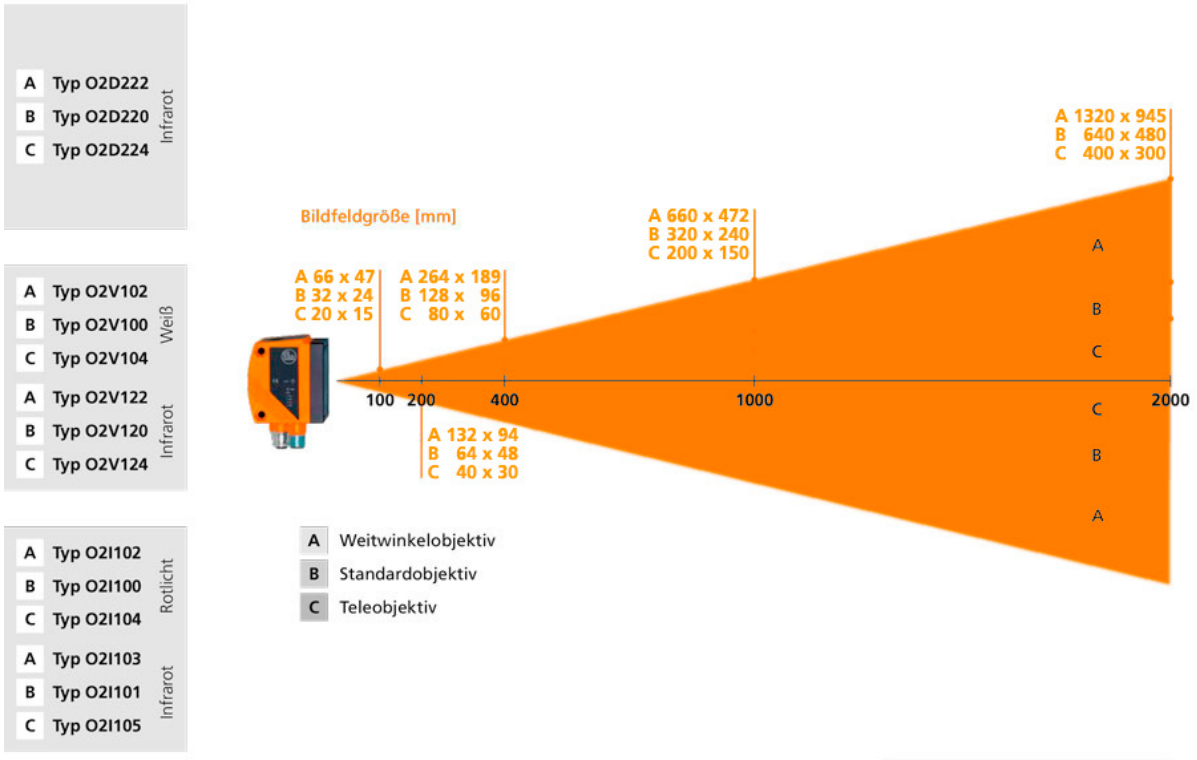


Figure 3 - O2D220 Range & Depth Information

1.3.2 IFM O2I100 Identification Sensor

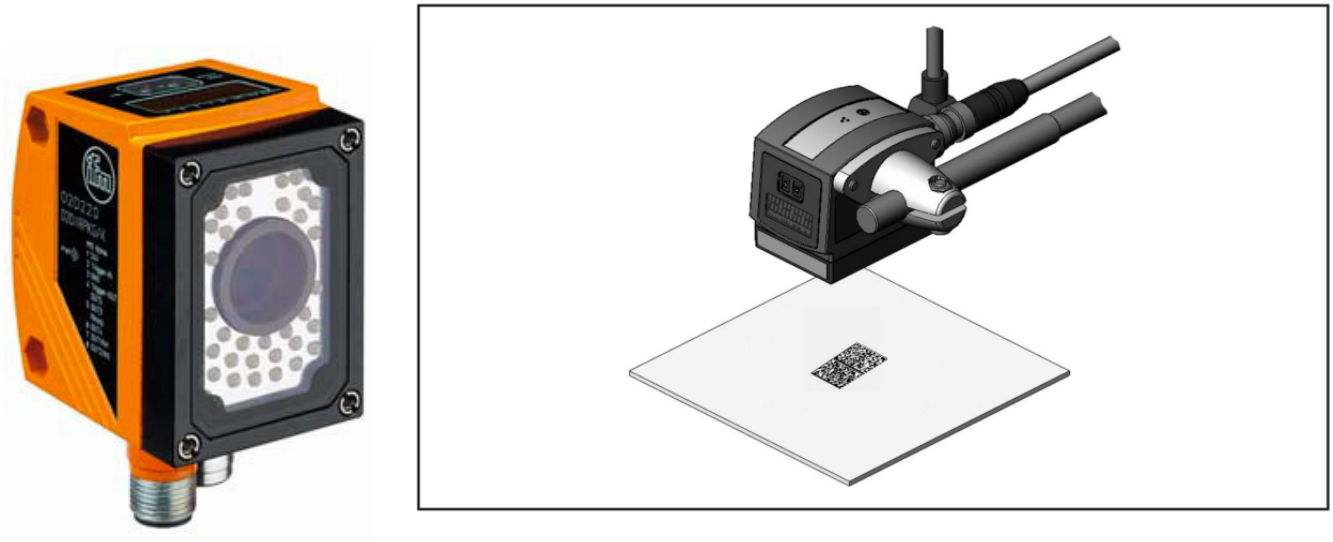


Figure 4 - O2I100 Photo & Example

The IFM O2I100 multicode reader is a sensor used to detect 2D codes and 1D barcodes. Features include integrated, configurable code evaluation, process interfaces RS-232, Ethernet TCP/IP and EtherNet/IP, parameter setting interface Ethernet TCP/IP and UDP/IP, internal illumination red light (625 nm) or infrared (850 nm), internal or external triggering.[3]

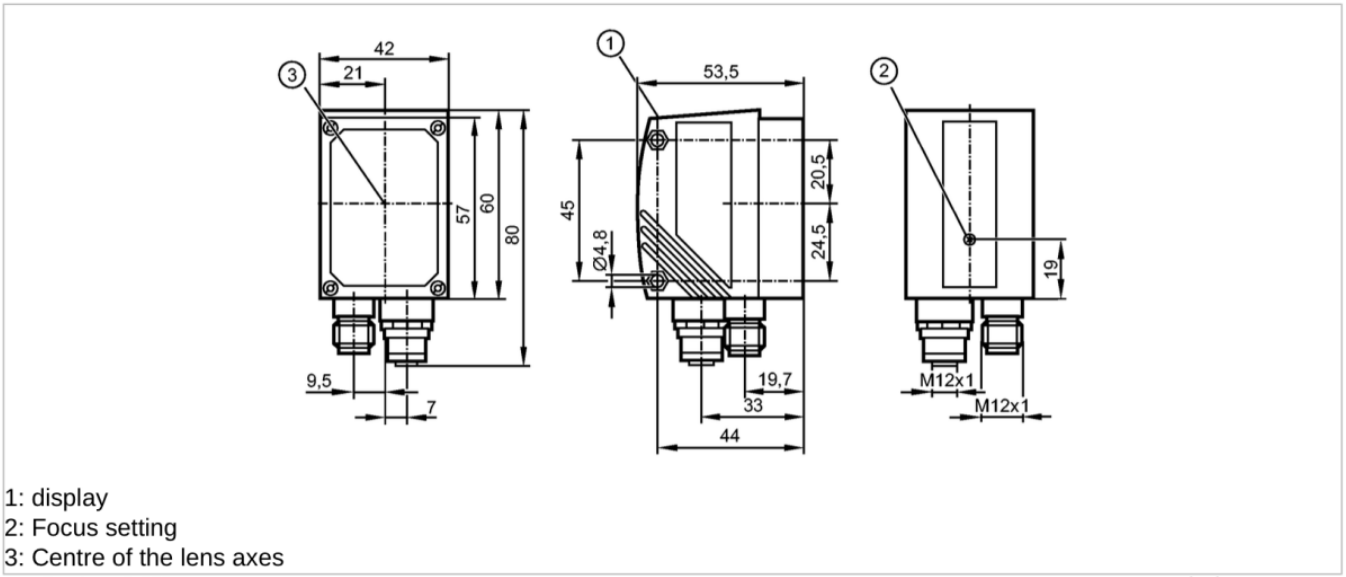


Figure 5 - O2I100 Elevations

[3] "Operating instructions Multicode Reader O2I10x - ifm." 2011. 28 April 2014 <https://www.ifm.com/mounting/704247UK.pdf>

1.3.3 SIEMENS SIMATIC S7-1200

Figure 6 Siemens Simatic S7-1200

The Siemens Simatic S7-1200 Basic Controller has a comprehensive range of technological functions and integrated IOs as well their particularly compact space-efficient design. They are the intelligent choice for all the everyday automation tasks with a small project scope.

A decisive advantage is the integration of all SIMATIC Controllers in the Totally Integrated Automation Portal (TIA Portal): All the SIMATIC controllers thus access a common data base, a uniform operating concept and central services.

The functionality of the SIMATIC S7-1200 controllers is derived consistently from the SIMATIC S7-1500 control systems developed for more complex tasks. This ensures uniform procedures and thus maximum efficiency in engineering, operation and maintenance as well as during changeovers.[4]

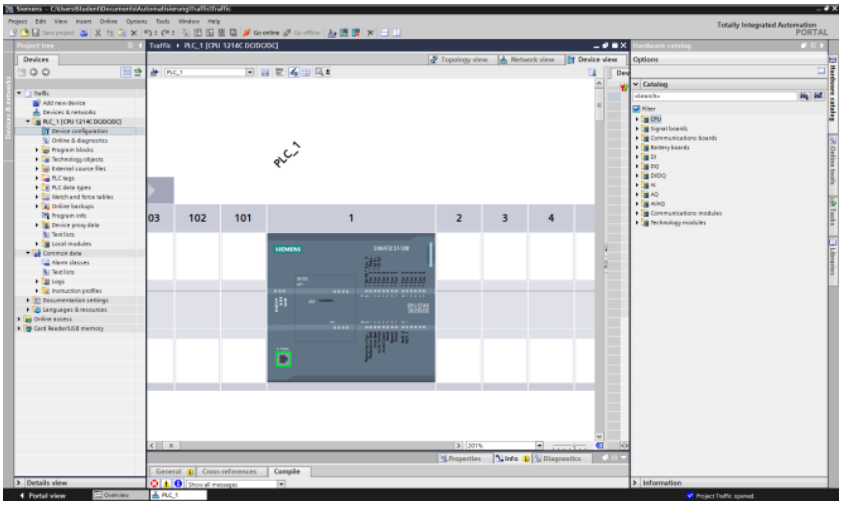


Figure 7 PLC Configuration in TIA Portal

[4] "http://www.simatec.ir/ 2015-01-14T04:54:21+00:00 monthly ..." 2014. 28 April 2014 <http://www.simatec.ir/sitemap.xml>

1.3.4 Siemens Touch HMI KTP600 Basic Colour PN

Figure 8 Siemens Touch HMI

The KTP600 is a 6 inch panel with 6 function keys and colour. It is a cost-effective device for small-scale HMI tasks for the use in PROFINET networks (RJ-45 Ethernet interface), particularly suitable for cramped space conditions directly at the machine especially in combination with SIMATIC S7-1200 controllers but also with other controllers.[5]

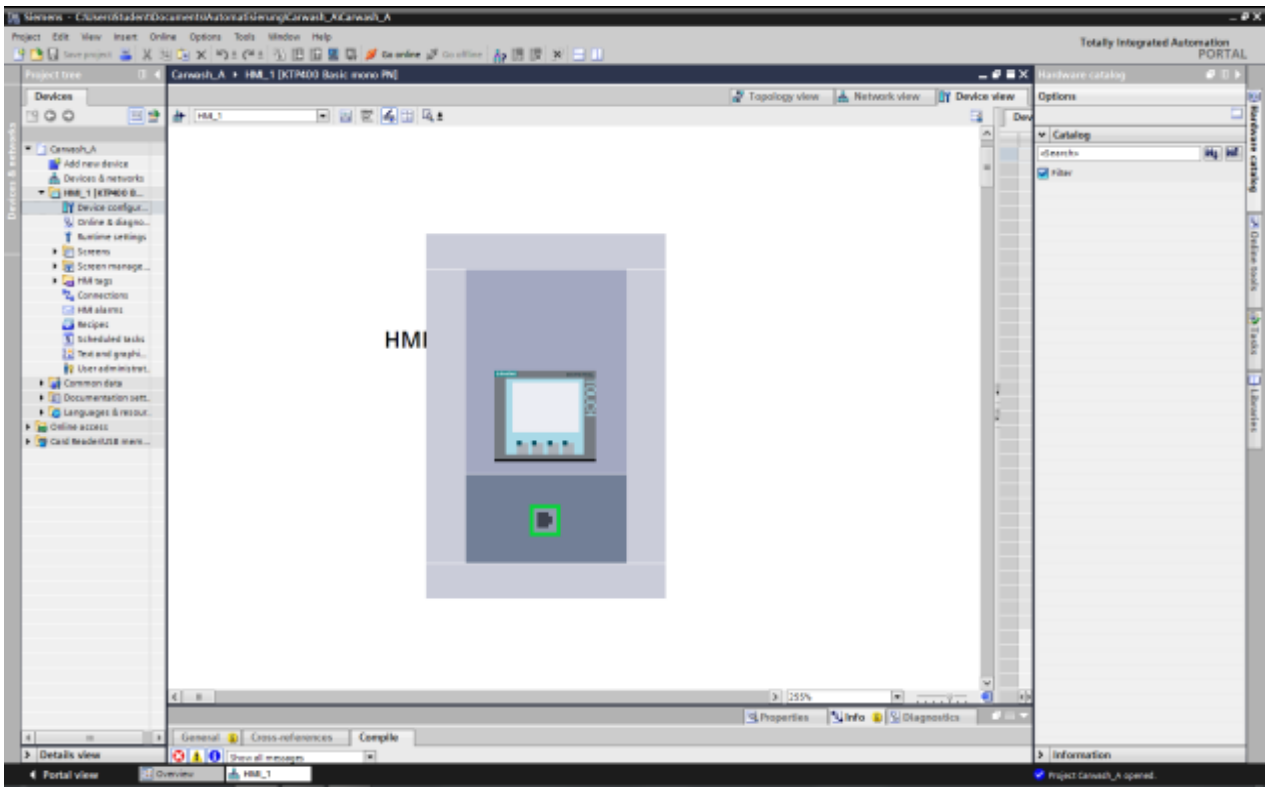


Figure 9 HMI Configuration in TIA Portal

[5] "1st Generation Device Overview - Operator devices - Siemens." 2014. 28 April 2014 <http://w3.siemens.com/mcms/human-machine-interface/en/operator-interfaces/basic-panel/devices-first-generation/pages/default.aspx>

1.3.5 Mitsubishi RV-E2 Arm & CR-E116

The Mitsubishi RV-E2 robotic arm is completely user friendly, featuring self-monitor and self-diagnosis function. It is controlled by the CR-E116 controllor box. It can be programmed with the teach pendant as well as with a PC, which dramatically reduces debugging time. It has a lifting capacity of 2.0kg and has excellent resultant speed and repeatability.[6]

We will use a pneumatic suction head attachment on the robot for picking up and placing the tiles. Industrial applications include picking and placing, deburring, and dispensing of materials. It is used in the medical, entertainment, educational and manufacturing fields.

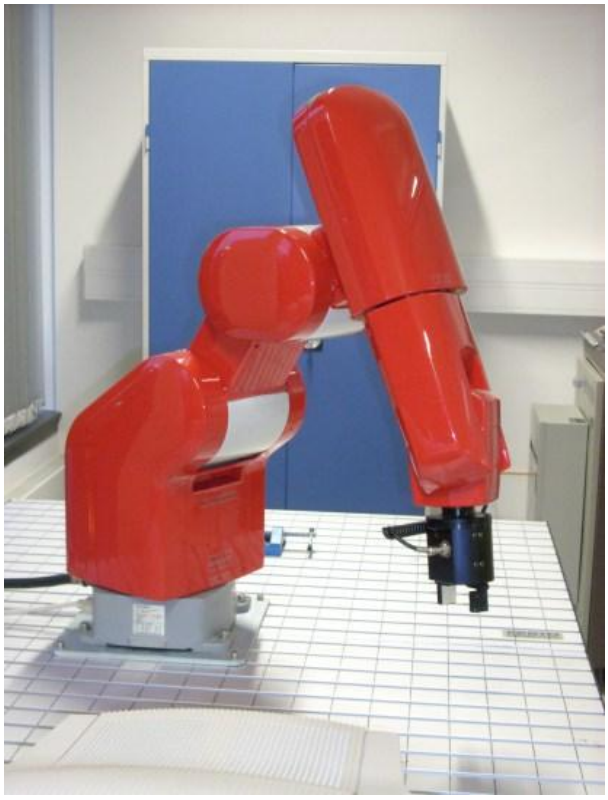


Figure 10 Mitsubishi RV-E2 Robotic Arm

Figure 11 Pneumatic Suction Head

[6] "Mitsubishi Robot RV-E2 Sale - Mitsubishi Robots Sale." 2010. 28 April 2014 <http://www.mitsubishirobots.com/RV-E2.html>

1.3.6 The Puzzle

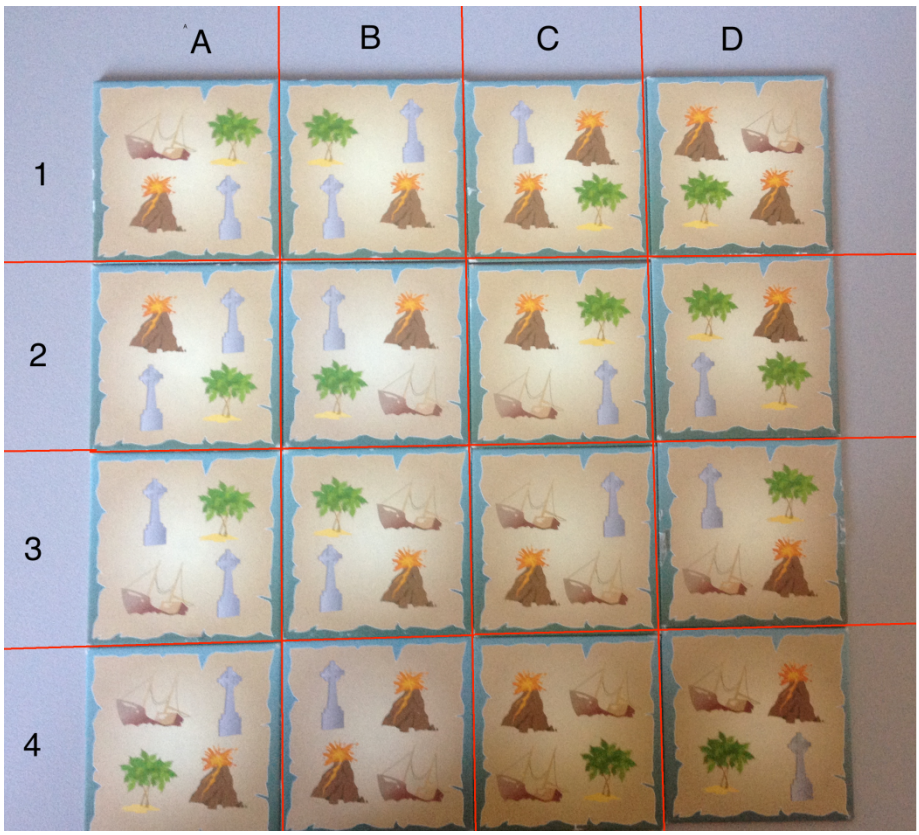


Figure 12 The 16 Tile Puzzle

This is the sixteen tile puzzle fully assembled in the correct order. Each tile consists of four images – either a Boat, Cross, Tree or Volcano. Each tile can only be placed next to a tile of the same image, side by side or top and bottom. There is only one correct way to assemble this puzzle making it very difficult to complete in a short space of time as one mistake early on will make it near impossible.

Chapter 2 Familiarisation with the TIA Portal

TIA Portal has three different methods for writing code. These include Ladder (LAD), Function Block Diagram (FBD) and Structured Control Language (SCL). For this project we needed to be familiarise ourselves with each of these methods for writing code so we could successfully complete our objectives. We were given three different tutorials to use the three different types of programming languages. These tutorials included simulating a set of traffic lights, a car wash and a car park. We completed each of these using LAD, FBD and SCL on each of the tutorials.

2.1 Ladder Coding

Unlike other methods of writing code, everything happens at once using Ladder logic. Normally in programming languages things happen in order, following each line from the top to the bottom until the end of a loop. There are no variables in ladder logic, instead there are registers. It works by using relay conditions (normally opened/closed) which in turn then energise or de-energise the output coils.[7]

2.1.1 Traffic Lights (LAD)

The traffic lights tutorial contained a set of traffic lights and a pedestrian crossing. The crossing could be activated by pressing S1. This would stop the traffic and allow pedestrians to cross the road. Another mode included night mode which had flashing amber lights for traffic which was activated by pressing S2. S0 was for turning off the lights and S3 turning back on to normal mode which was traffic on then when traffic stopped pedestrians could cross.

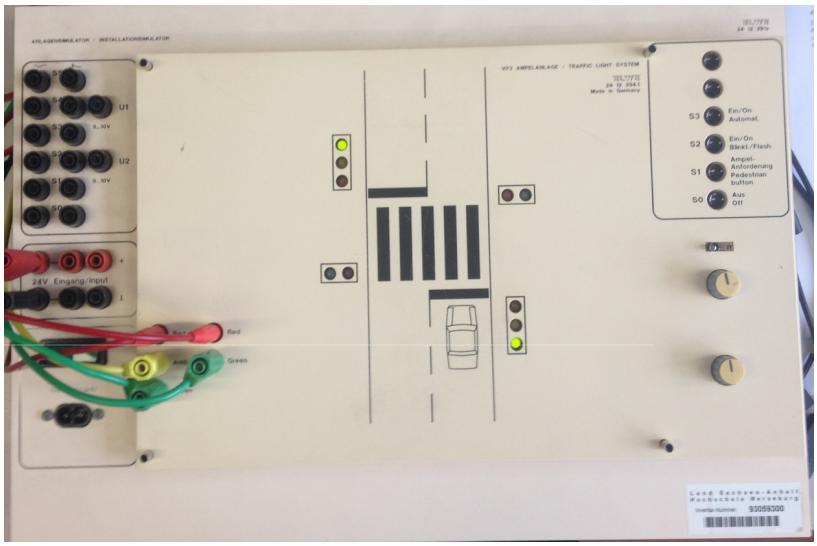


Figure 13 Traffic Light Example Used

[7] "Intro to Ladder Logic - Transwiki." 2015. 29 Apr. 2015 <https://wiki.xtronics.com/index.php/Intro_to_Ladder_Logic>

Here is an example of the code we used:

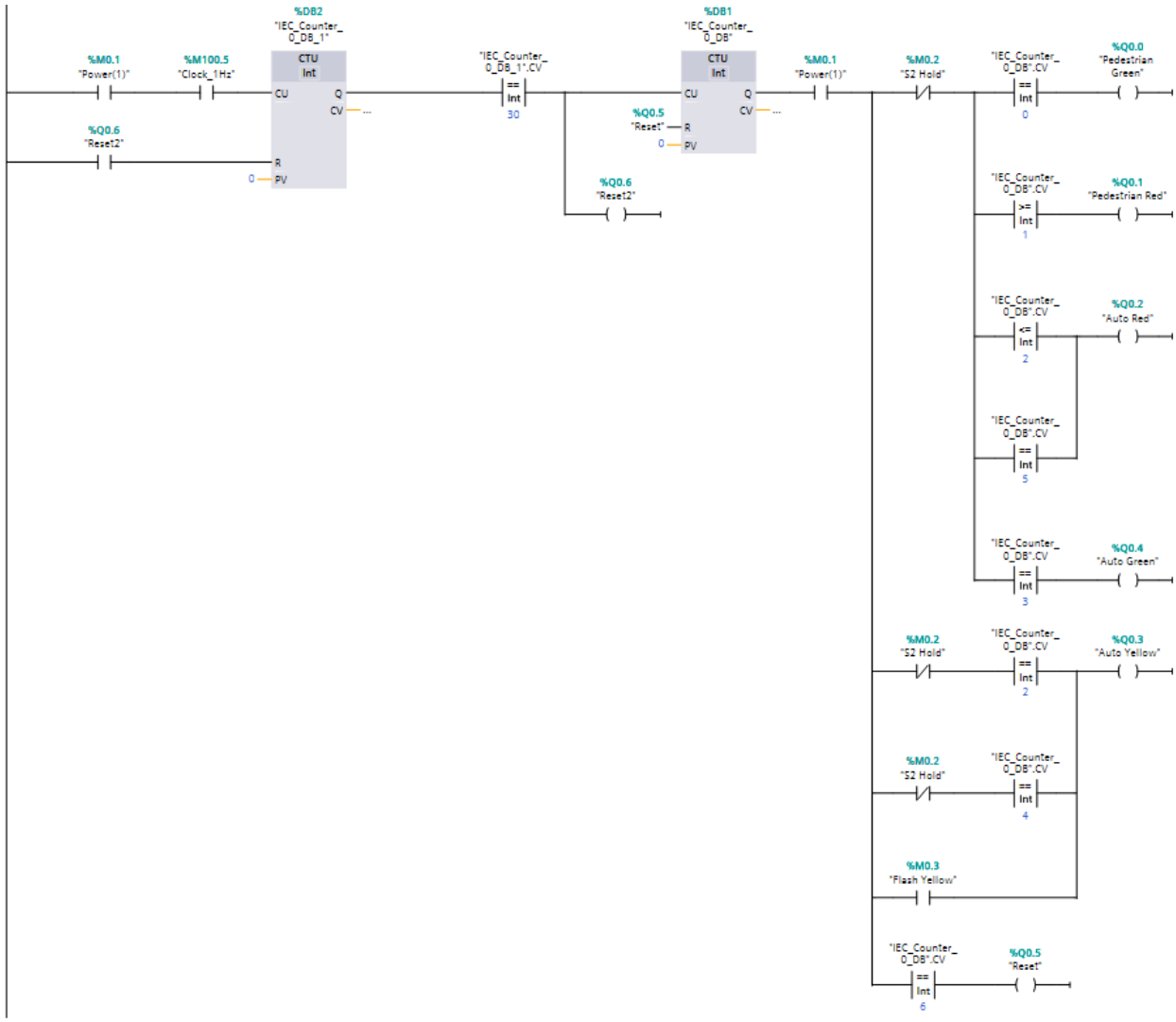


Figure 14a Ladder Logic Example

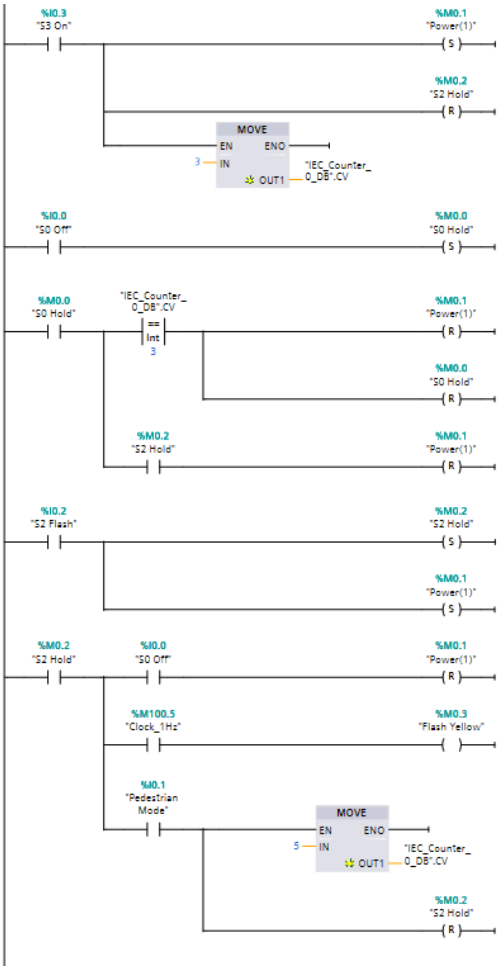


Figure 14b Ladder Logic Example

Here is a link to our video on youtube of the example we completed:
https://www.youtube.com/watch?v=iSSyraW59mo

2.2 Function Block Diagram

A generalised function block consists of input variables, output variables, through variables, internal variables, and an internal behavior description of the function block. Input variables can only be written from outside of an FB. From inside they can only be read. Output variables can be read and written from inside of an FB and only be read from outside. Through variables are special shared variables. If through variables of different FB instances are connected, they do all access the variable connected to the first input of the chain.[8]

[8] Zhang, Wei, Wolfgang A Halang, and Christian Dietrich. "Specification and verification of applications based on function blocks." *Component-Based Software Development for Embedded Systems* (2005): 8-34.

2.2.1 Car Wash Example (FBD)

Our second tutorial was a car wash, which included a wash mode, a dry mode and an emergency stop mode. For this tutorial we decided to add the HMI and used it for our inputs to the PLC. When wash mode is active the horizontal sweeper must find the level at which to clean and all sweepers must rotate while car is being detected. In dry mode, the horizontal bar must follow the car to simulate air being blown, and in emergency mode all motors must come to a complete halt and the plc may only continue operation by a manual reset.

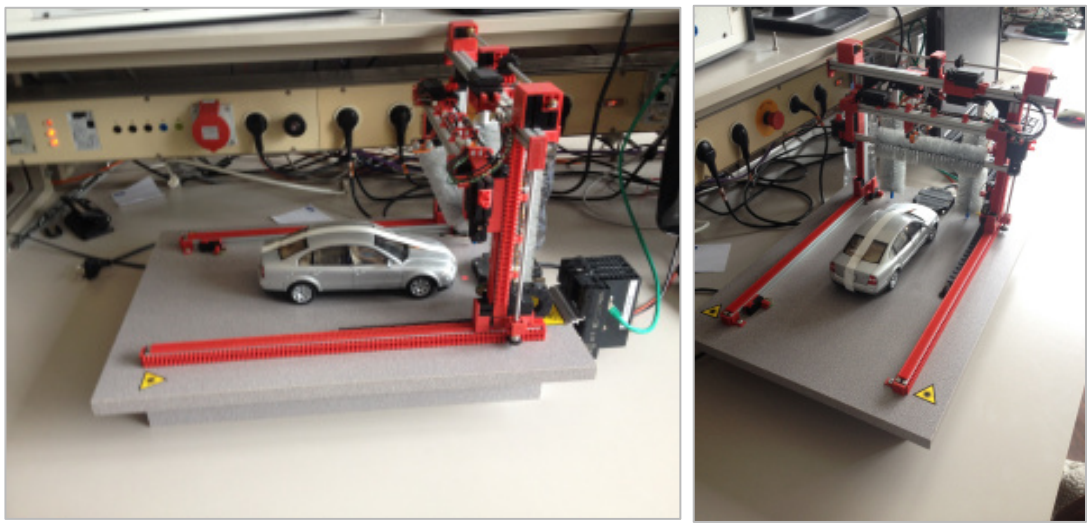


Figure 15 Car Wash Example Used

Figure 16 HMI Used With Car Wash Example

Here is the FBD code we used to to complete this example:

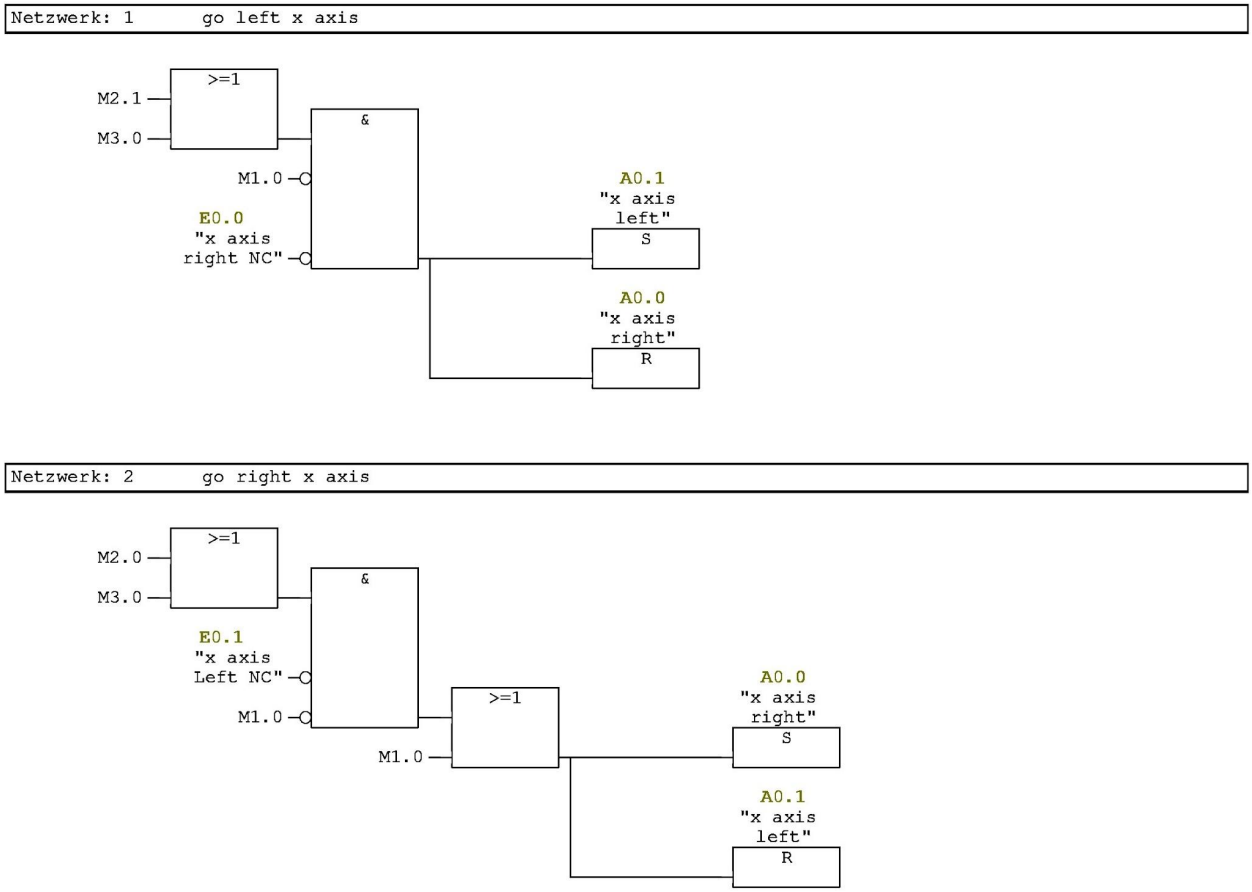


Figure 17a Function Block Diagram Example

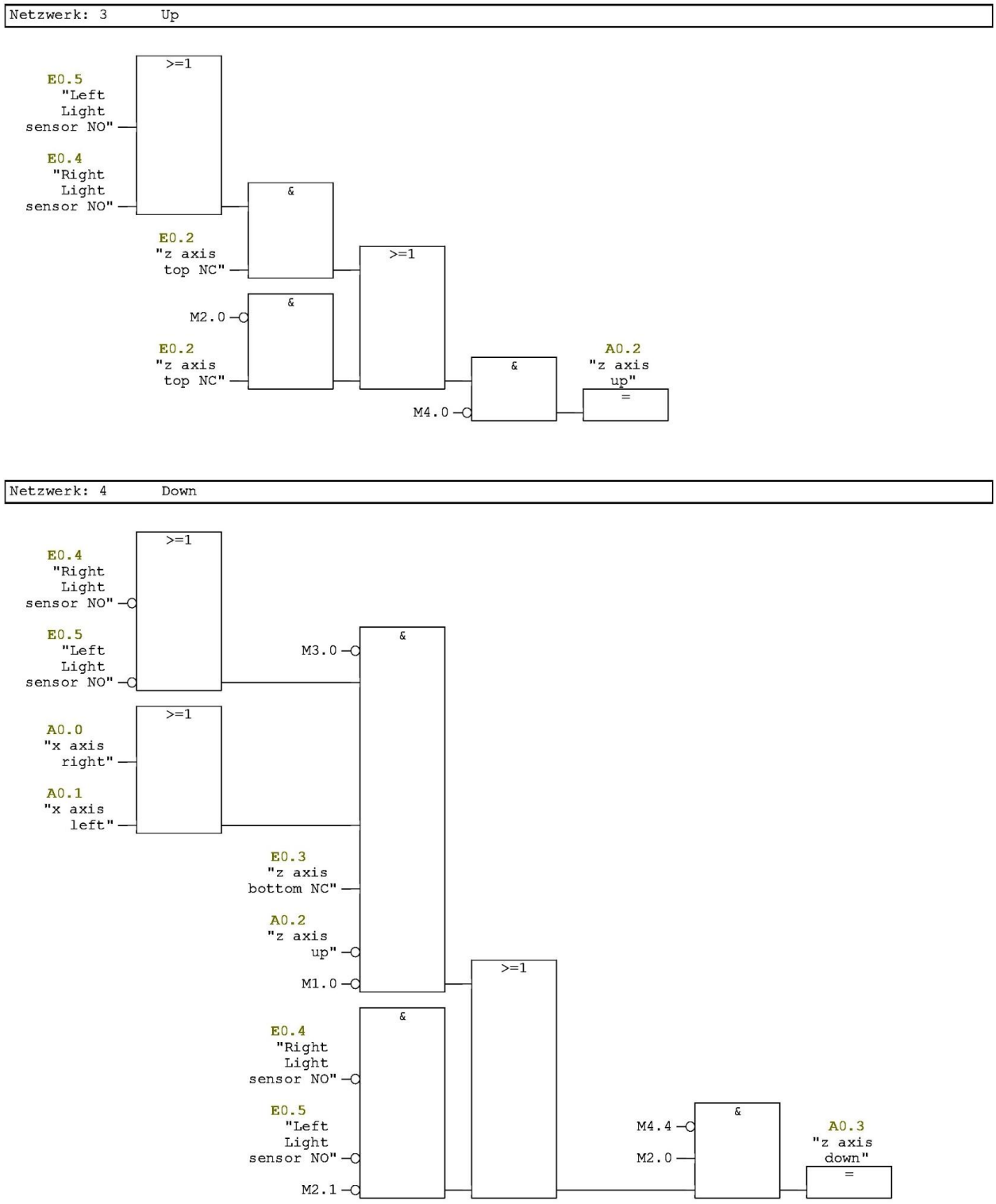


Figure 17b Function Block Diagram Example

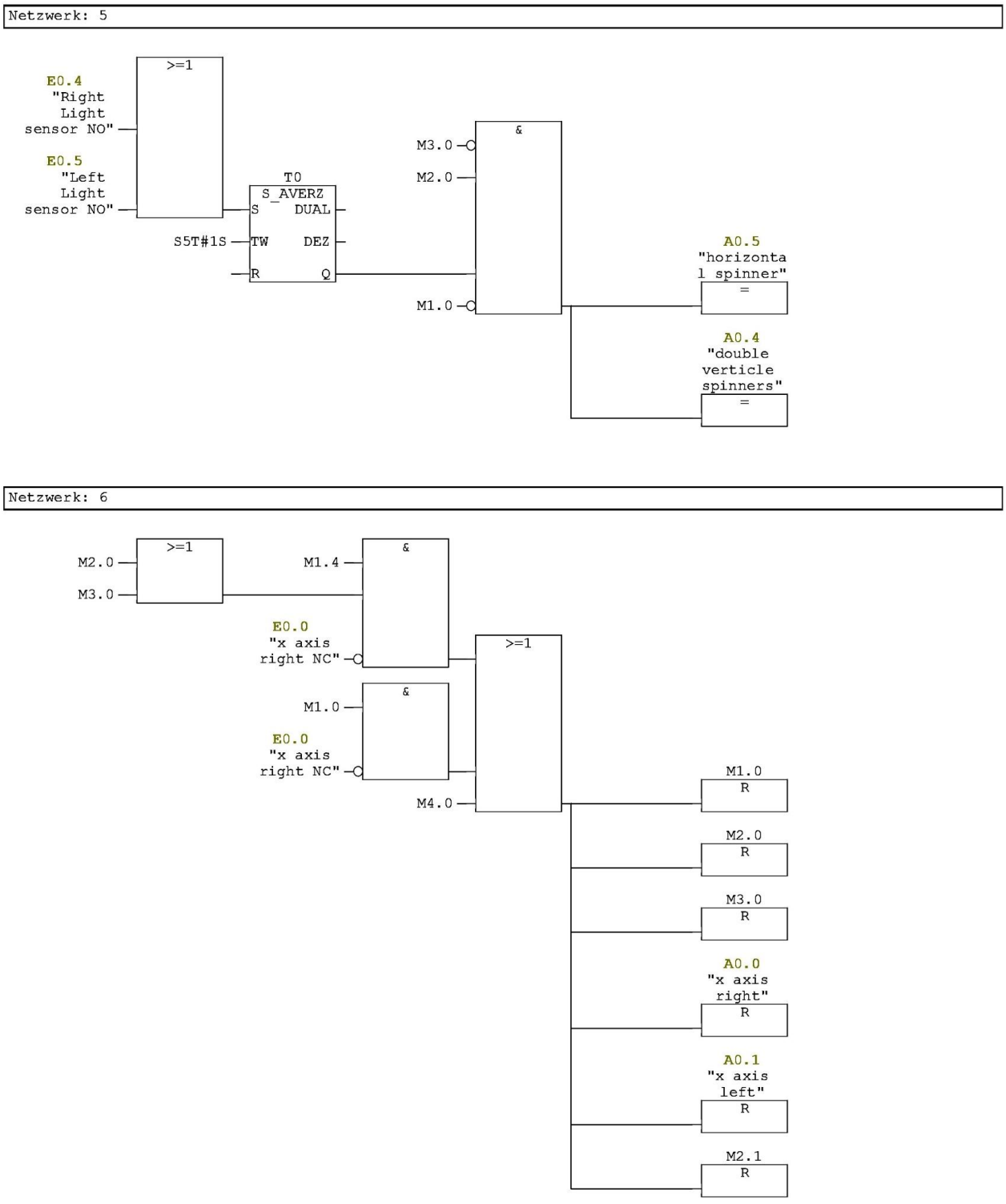


Figure 17c Function Block Diagram Example

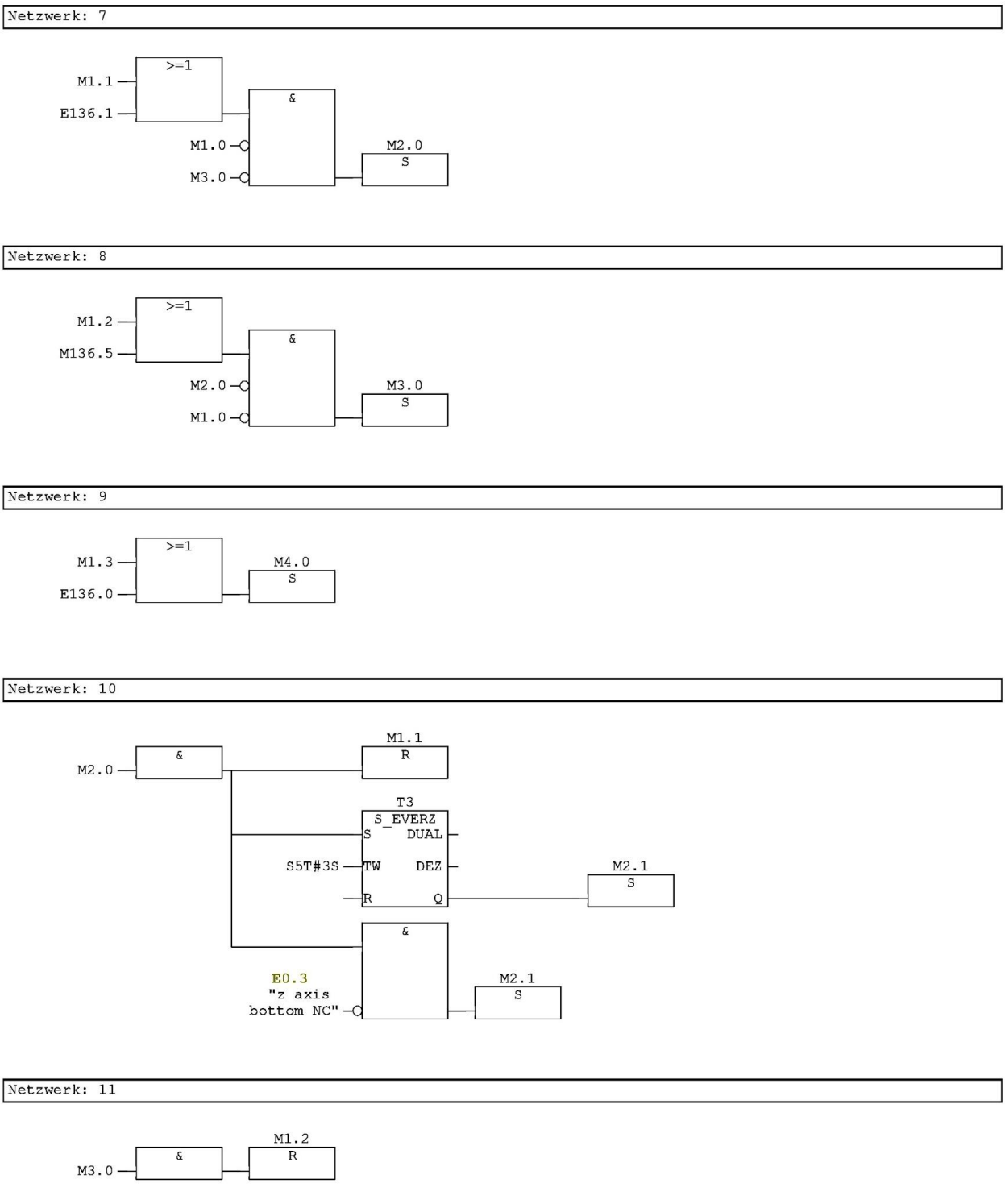


Figure 17d Function Block Diagram Example

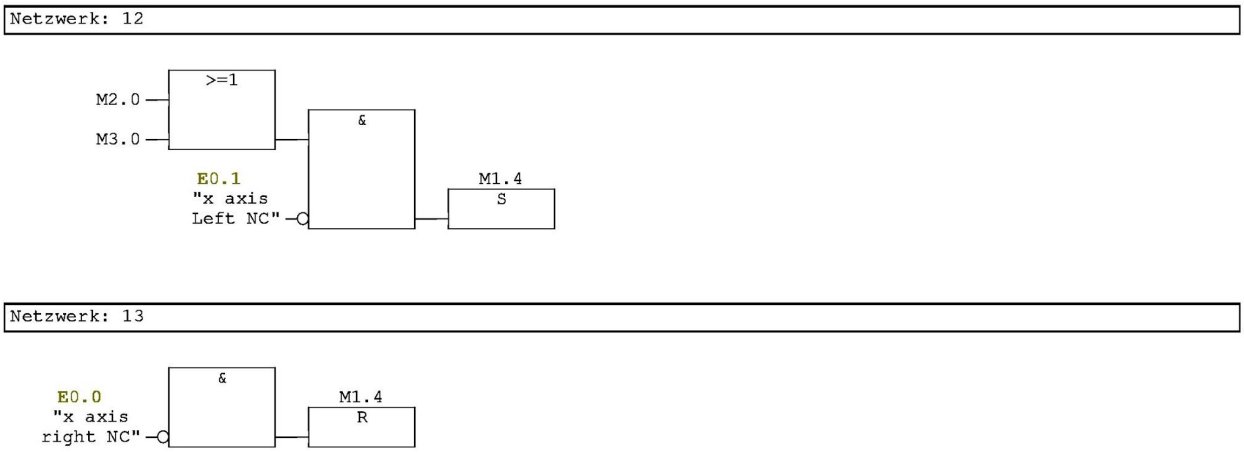


Figure 17e Function Block Diagram Example

Here is a link to our video on youtube of the example we completed:
https://www.youtube.com/watch?v=xzhQzjLjFfs

2.3 Structured Control Language

SCL (Structured Control Language) is a high-level textual programming language which is based on PASCAL. It is also based on a standard for PLCs (programmable logic controllers). PASCAL is a textual programming language largely used by beginners or as a teaching language. It is a very easy to understand language and it also a good language to learn so you get a foundation of the basics of programming.
In addition to high-level language elements, SCL also includes language elements typical of PLCs such as inputs, outputs, timers, bit memory, block calls, etc. In other words, SCL complements and extends the STEP 7 programming software and its programming languages Ladder Logic and Statement List.[9]

2.3.1 Car Park Example (SCL)

Our final example was a car park tutorial, the example included a car park full (red) and space (green) lights, a ticket machine with (red and green) lights and an exit (red) light and timers to allow time for entering and exiting. There was also an emergency mode, in which no one was allowed to enter and everyone was able to exit. All of which were implemented using SCL code.

[9] "Structured Control Language (SCL) for S7-300 ... - Siemens." 2015. 29 Apr. 2015 <https://cache.industry.siemens.com/dl/files/188/1137188/att_27471/v1/SCLV4_e.pdf>

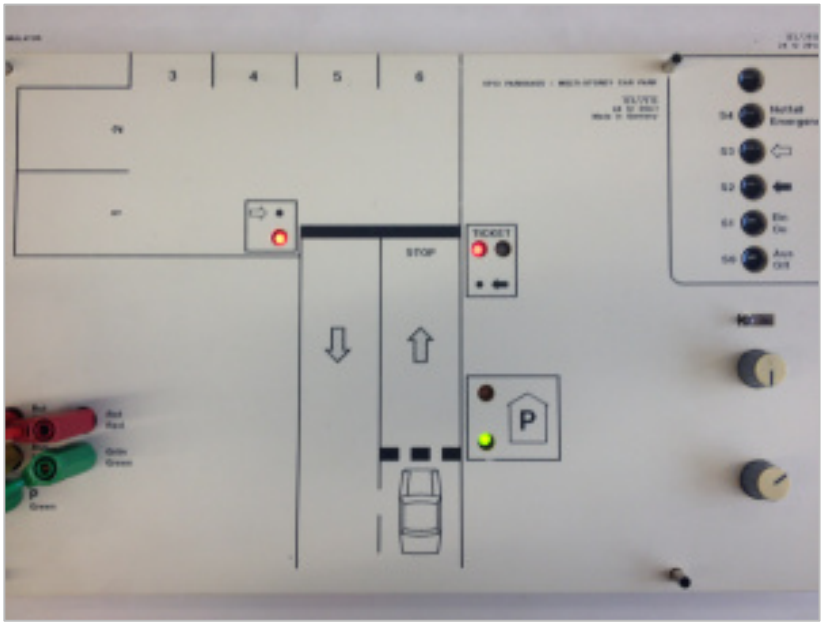


Figure 18 Car Park Example Used

Here is the SCL code we used for this example:

```
IF "s1_on" = 1 THEN
    #on := 1;
    #off := 0;
    #emergency := 0;
END_IF; // On mode

IF "s0_off" = 1 THEN
    #on :=0;
    #off := 1;
    #emergency := 0;
END_IF; // Off mode

IF "s4_emergency" = 1 THEN
    #emergency := 1;
END_IF; // Emergency mode
"IEC_Counter_0_DB".CTUD(CU := #count_up,
                        CD := #count_down,
                        PV := 0,
                        CV => #counter_value);
// entance lights
IF "IEC_Counter_0_DB".CV > 6 THEN
    "IEC_Counter_0_DB".CV := 6;
END_IF;
IF "IEC_Counter_0_DB".CV < 0 THEN
    "IEC_Counter_0_DB".CV := 0;
END_IF;
IF "IEC_Counter_0_DB".CV = 6 THEN
    "parking_red" := 1;
    "parking_green" := 0;
ELSE
    "parking_green" := 1;
    "parking_red" := 0;
END_IF;

IF #on = 1 THEN
    "ticket_red" := 1;
    "exit_red" := 1;
    "ticket_green" := 0;
    IF #entering = 1 AND "IEC_Counter_0_DB".CV < 6 THEN
        "ticket_green" := 1;
        "ticket_red" := 0;
    END_IF; // entering lights
    IF #exiting = 1 THEN
        "exit_red" := 0;
    END_IF;// exiting light
    IF "s2_entry" = 1 AND "ticket_red" = 1 THEN
        #count_up := 1;
    END_IF;
    IF "s3_exit" = 1 AND "exit_red" = 1 THEN
        #count_down := 1;
    END_IF;
    IF "s2_entry" = 0 THEN
        #count_up := 0;
    END_IF;
    IF "s3_exit" = 0 THEN
        #count_down := 0;
    END_IF;
END_IF;

IF #emergency = 1 THEN
    "parking_red" := 1;
    "parking_green" := 0;
    "ticket_red" := 1;
    "ticket_green" := 0;
    "exit_red" := 0;
    #count_up := 0;
    #count_down := 0;
END_IF; // when in emergency mode...

IF #off = 1 THEN
    "parking_red" := 0;
    "parking_green" := 0;
    "ticket_red" := 0;
    "ticket_green" := 0;
    "exit_red" := 0;
    #count_up := 0;
    #count_down := 0;
END_IF; // when in off mode...

"IEC_Timer_0_DB".TOF(IN := #count_up,
                     PT := #delay_timer0,
                     Q => #entering);
"IEC_Timer_0_DB_1".TOF(IN:=#count_down,
                       PT:=#delay_timer1,
                       Q=>#exiting);
```

Figure 19 Structured Control Language Example

Here is a link to our video on youtube of the example we completed:
https://www.youtube.com/watch?v=V9TFI9MPBK

Chapter 3 Mitsubishi Movemaster RV-E2 Arm & CR-E116 Contol Box

3.1 Introduction

We were given the Mitsubishi Movemaster RV-E2 Robotic Arm and CR-E116 Control Box as part of our project to pick up tiles, move them to respective positions to be scanned and identified and finally, be placed in the correct position as part of the puzzle. In order to familiarise ourselves with this robotic arm we attempted some tutorials and trials. But first we must understand the Teach Pendant, the COSIPROG program and the commands.

3.1.1 Remote (Mitsubishi P6TB-TE Teach Pendant)

Figure 20 P6TB-TE Teach Pendant

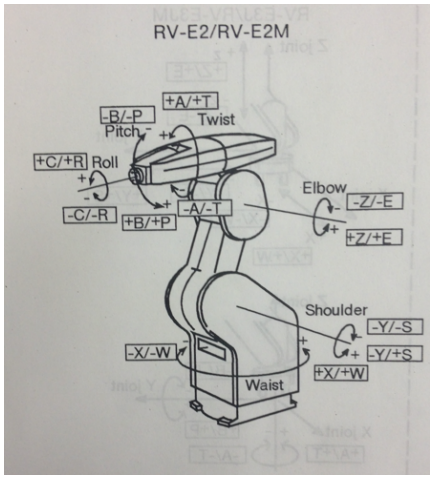


Figure 21 RV-E2 Joint Jog Operation

This device creates a robot program by teaching (moving the robot and memorizing the positions). The jog feed and position display requirements for teaching can be done easily. With large keys and a nice LCD display, simple programs can be created, edited and controlled by the teach pendant.[10]

3.1.2 Program (COSIPROG)

COSIPROG is a technology developed specifically for the education and training in the field of robotics. It is a universal tool that runs on Windows and has features such as windowing, a convenient user interface and multitasking. COSIPROG allows you to create programs and position lists in other programming languages and then connect and sync to the robot's teaching box.

[10] "MOVEMASTER." 2011. 30 Apr. 2015 <http://dim.usal.es/eps/im/roberto/pesados/rob/RV-E2_especifications_manual.pdf>

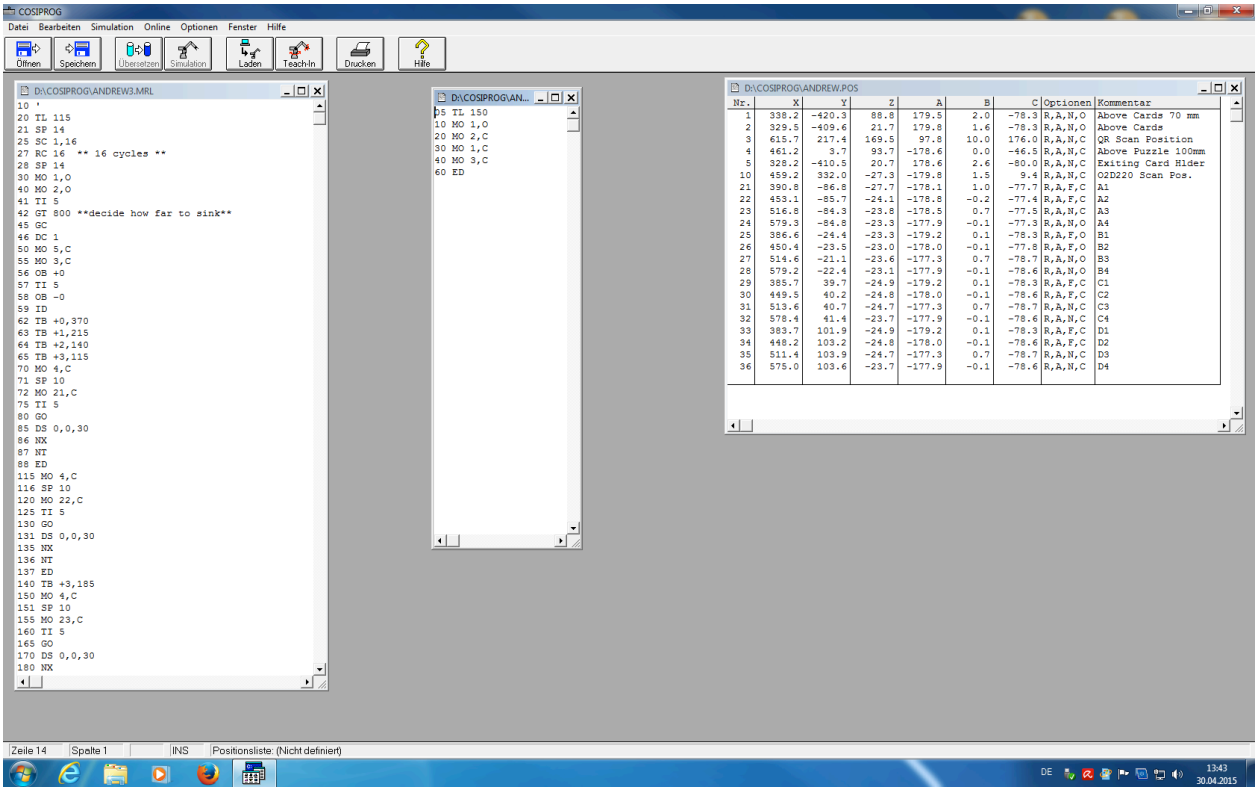


Figure 22 COSIPROG Program Example

3.1.3 Commands

There is a wealth of commands available and provided for the Movemaster command method. These commands include position and motion control commands, program control commands, hand control commands, I/O control commands, comunication commands and other commands like setting parameters, selecting programs, resetting alarms and describing comments.[11]

```
10 '
20 TL 115
21 SP 14
25 SC 1,16
27 RC 16  ** 16 cycles **
28 SP 14
30 MO 1,O
40 MO 2,O
41 TI 5
42 GT 800 **decide how far to sink**
45 GC
46 DC 1
50 MO 5,C
55 MO 3,C
56 OB +0
57 TI 5
58 OB -0
```

```
59 ID
62 TB +0,370
63 TB +1,215
64 TB +2,140
65 TB +3,115
70 MO 4,C
71 SP 10
72 MO 21,C
75 TI 5
80 GO
85 DS 0,0,30
86 NX
87 NT
88 ED
115 MO 4,C
116 SP 10
120 MO 22,C
```

Figure 23 Example Of Commands Being Used

[11] "mitsubishi robot movemaster manuals - Mitsubishi RV-3AL." 2008. 30 Apr. 2015 <http://www.mitsubishirobot.com/ta112.html>

Command	Input Format	Function
Data Read	DR [register no]	Reads the number of an internal register
End	ED	Terminates program
Grip Close	GC	Closes grip of hand
Grip Open	GO	Opens grip of hand
Increment Counter	IC [counter no]	Adds one to the value of the specified counter
Input Direct	ID	Fetches input data and checks it
Move	MO <position no>	Moves grip to position
Nest	NT	Moves to nest position and carries out origin setting
Output Bit	OB <+/-> <bit no>	Sets the output state of a specified bit
Repeat Cycle	RC <no of cycles>	Repeats the loop a specified no. of times
Set Counter	SC<counter><value>	A specified value is set to a specified counter
Speed	SP <speed>	Sets the operating speed
Test Bit	TB <+/-> <bit no> <line no>	Jumps to set line if the bit is returned a match when tested
Timer	TI <time count>	Time delay
Tool	TL <tool length>	Defines the length of the tool

Table 1 Command Types

3.2 Laser Test

In order to test and practice with the programming environment of the RV-E2 robotic arm we attached a laser to the head of the arm and attempted various tutorials. The idea was to start with simple tasks using the robot's many different commands. We began with the laser following simple lines on the working surface by using the robot's remote to 'teach in' the positions manually and then writing the program language on the computer. We then moved on to more difficult tutorials where the laser followed a cut pipe around its edge, this allowed us to improve our accuracy and get more familiar with its more difficult commands.

Figure 24 Mitsubishi RV-E2 Laser Test

Here is a link to our video on youtube of the example we completed:
https://www.youtube.com/watch?v=G_9YuJUAHW4

3.3 Suction Head Test

We attached a suction head to the Mitsubishi RV-E2. The suction head was attached through the RV-E2 to a compressor, and the head can be programmed to be opened or closed. This head is what we used for picking up and setting down the puzzle tiles. Our tests included moving tiles out of the tile holder and setting down/picking up tiles around the working surface. If the tiles were not placed in the correct position, even being off by the slightest millimeter, it caused them to overlap onto each other meaning each tile had to be picked up/set down the exact same way each time. This meant each card had to be carefully moved around by the robotic arm. We had issues with the power of the air from the compressor moving tiles as the robotic arm passed over them but by controlling the pressure and keeping the arm further above the tiles it prevented this from being a major issue. We also used time delays to allow time for the arm to stabilize, and the draw straight (DS) command to lift the arm vertically by approximately 25mm to clear all the previously set down cards.

3.4 Programming Each Position

Each position had to be manually located and programmed. The easiest way for us to do this was by using the "teach in" function on the CR-E116 control box. The locations were saved to a position list and using the program "COSIPROG" we wrote code to place each tile in its final position. To start we found the first four positions for tiles A1 – A4. Using these positions it was easier to find the rest by moving the robotic arm at an axis level to the first tiles. We also created positions for taking the tiles out of the tile holder and setting them down on the working surface. These positions were carefully programmed into the robot's position list, so when the robotic arm carried out an action it would keep clear of both the previously placed tiles or tile holder to avoid moving them and disrupting their positions. The tiles also had to be shown to each of the IFM sensors. First a tile was moved out of the holder and moved to a position under IFM O2D220. The robotic arm was then taken out of the way so the sensor could read the contours of the tile. Once the contours were scanned it moved to IFM O2I100 where it waited for the QR code to be read. Once a signal was received from the sensor, the tile was then ready to be moved again. If both sensor outputs were matched the tile was identified and set in the correct position. In total we used 22 positions and all are listed in the below figure.

Nr.	X	Y	Z	A	B	C	Optionen	Kommentar
1	338.2	-420.3	88.8	179.5	2.0	-78.3	R,A,N,O	Above Cards 70 mm
2	329.5	-409.6	21.7	179.8	1.6	-78.3	R,A,N,O	Above Cards
3	615.7	217.4	169.5	97.8	10.0	176.0	R,A,N,C	QR Scan Position
4	461.2	3.7	93.7	-178.6	0.0	-46.5	R,A,N,C	Above Puzzle 100mm
5	328.2	-410.5	20.7	178.6	2.6	-80.0	R,A,N,C	Exiting Card Hlder
10	459.2	332.0	-27.3	-179.8	1.5	9.4	R,A,N,C	O2D220 Scan Pos.
21	390.8	-86.8	-27.7	-178.1	1.0	-77.7	R,A,F,C	A1
22	453.1	-85.7	-24.1	-178.8	-0.2	-77.4	R,A,F,C	A2
23	516.8	-84.3	-23.8	-178.5	0.7	-77.5	R,A,N,C	A3
24	579.3	-84.8	-23.3	-177.9	-0.1	-77.3	R,A,N,O	A4
25	386.6	-24.4	-23.3	-179.2	0.1	-78.3	R,A,F,O	B1
26	450.4	-23.5	-23.0	-178.0	-0.1	-77.8	R,A,F,O	B2
27	514.6	-21.1	-23.6	-177.3	0.7	-78.7	R,A,N,O	B3
28	579.2	-22.4	-23.1	-177.9	-0.1	-78.6	R,A,N,O	B4
29	385.7	39.7	-24.9	-179.2	0.1	-78.3	R,A,F,C	C1
30	449.5	40.2	-24.8	-178.0	-0.1	-78.6	R,A,F,C	C2
31	513.6	40.7	-24.7	-177.3	0.7	-78.7	R,A,N,C	C3
32	578.4	41.4	-23.7	-177.9	-0.1	-78.6	R,A,N,C	C4
33	383.7	101.9	-24.9	-179.2	0.1	-78.3	R,A,F,C	D1
34	448.2	103.2	-24.8	-178.0	-0.1	-78.6	R,A,F,C	D2
35	511.4	103.9	-24.7	-177.3	0.7	-78.7	R,A,N,C	D3
36	575.0	103.6	-23.7	-177.9	-0.1	-78.6	R,A,N,C	D4

Figure 25 Final Position List

Chapter 4 IFM Sensors

4.1 IFM O2I100 Identification Sensor

4.1.1 Creating QR Codes

A QR code (quick response code) is a type of 2D bar code that is used to provide easy access to information.[12] It is an image of black and white squares containing information like Url links, geo coordinates, and text. One of the functions of the O2I100 identification sensor is to read these QR codes, which is why we decided to use it for our project. We needed sixteen different QR codes for each of the puzzle tiles. There are lots of ways to create these codes, using a website we found on the internet we created a QR code for each tile with just a numeric value of "01-16".

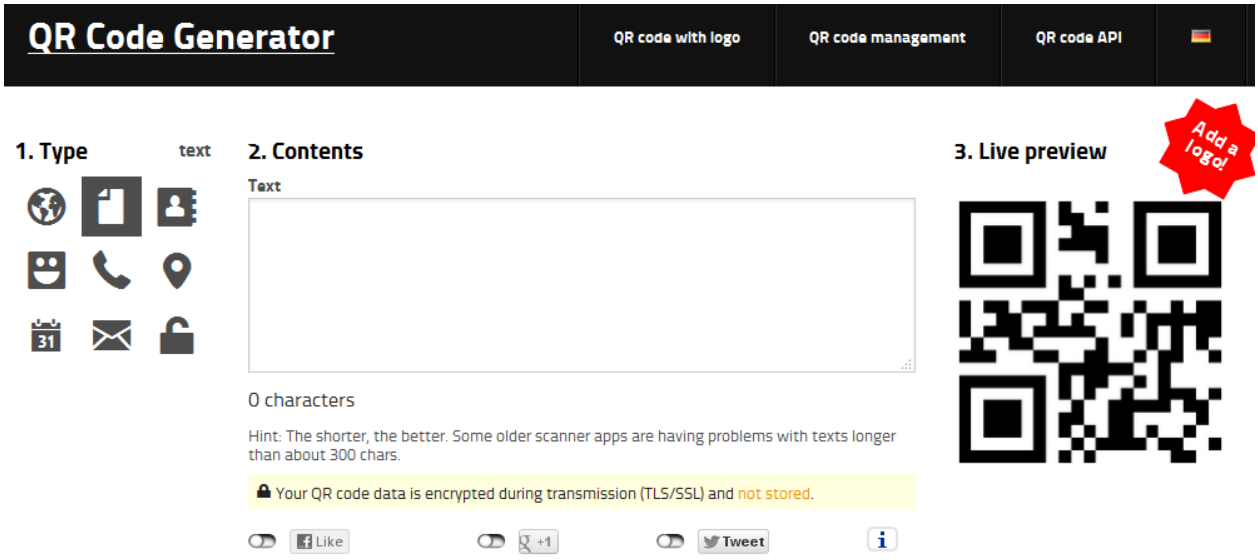


Figure 26 (http://goqr.me/) QR Code Generator Website

4.1.2 Software

In conjunction with a multicode reader O2I100 the PC operating program, Efector Dualis Multicode Reader parameter setting software, provides the following options that allow you to create, administer, name and/or group application-specific configurations. The program also allows real-time monitor mode for set-up and service purposes and an ability to save service reports for statistical evaluations.[13] Once installed on the computer, and the device is connected to the computer via ethernet RJ45 connection one can proceed to the configuration of the device, where the device and program may be set up via TCP/IP. For our project we used version 1.4, build 102.

---

[12] "What is QR code (quick response code)? - Definition from ..." 2012. 4 May. 2015 <http://whatis.techtarget.com/definition/QR-code-quick-response-code>

[13] "E2I200 / V1.3 - ifm." 2010. 4 May. 2015 <http://www.ifm.com/mounting/704743UK.pdf>

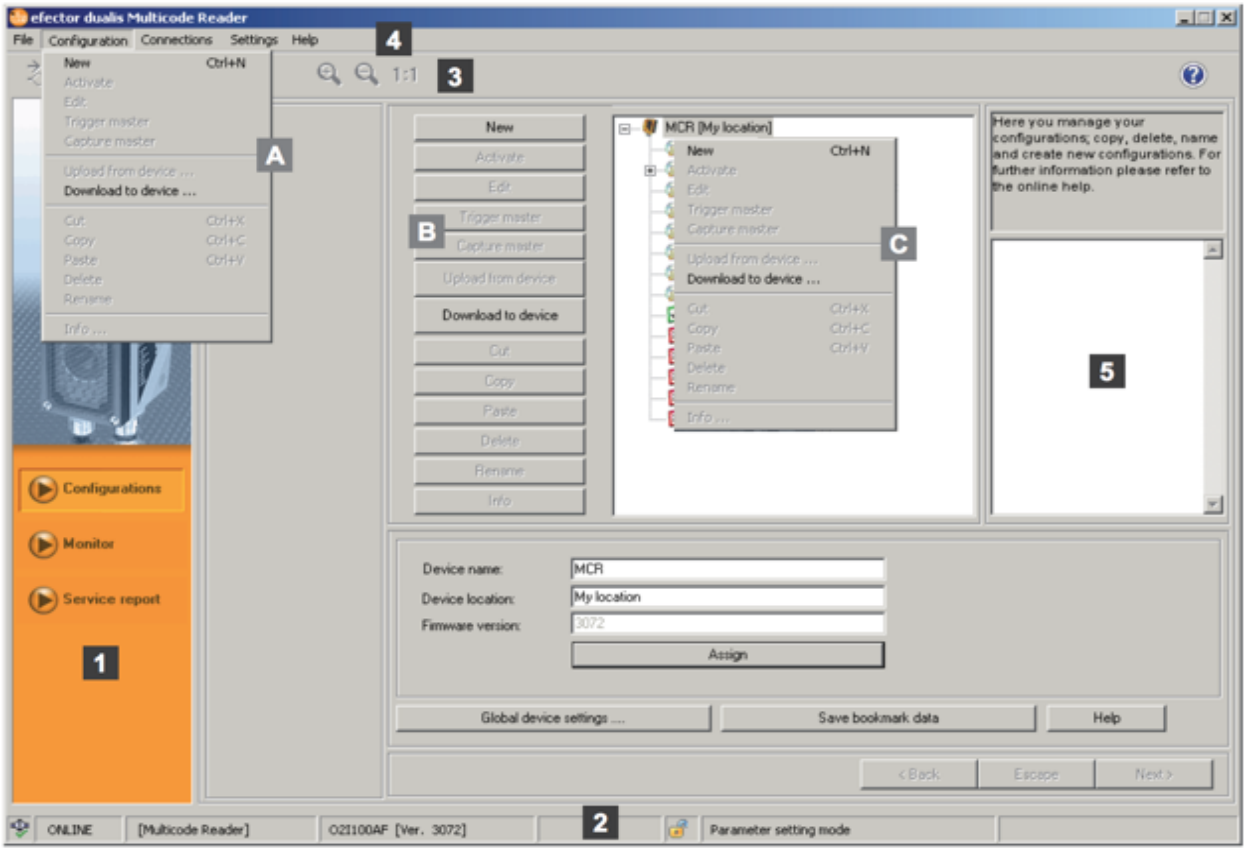


Figure 27 Basic User Interface (Dualis)

Pos.	Display / operating elements	Contents
1	Mode	• Configurations Create, administer or group configurations. By changing into this mode, the device will stop the read mode. • Monitor Device will run independently with saved and activated group or configuration. The read operation can be observed. • Service report By changing into this mode, the device will stop the read operation. The results, statistics and captured images can be activated and/or saved.
2	Status bar	• Network status of the device (OFFLINE/ONLINE) • Device name • Article number/production status/firmware of the connected device • Password protection on/off (lock symbol) • Program status (current program function)
3	Tool bar	Buttons (e.g. "connect" or "disconnect") Commands that cannot be selected are displayed in grey.
4	Menu bar	Pulldown menus with program functions.
5	Result field	• Read result e.g. number of found codes, code content, read time, total decoding time
A/B/C	Selection variants	Identical commands can be selected in different ways: (depending on the program function). A = selection via pulldown menu in the menu bar B = selection via button C = selection via context menu (click with right mouse button)

Table 2 Interface Elements & Contents (Dualis)

Within this software there also a monitor mode function and in this mode the operation of the device is observed. Each time the divice is triggered, the image captured is transferred to the operating program, displayed and evaluated. The respective read result is displayed in the result field.

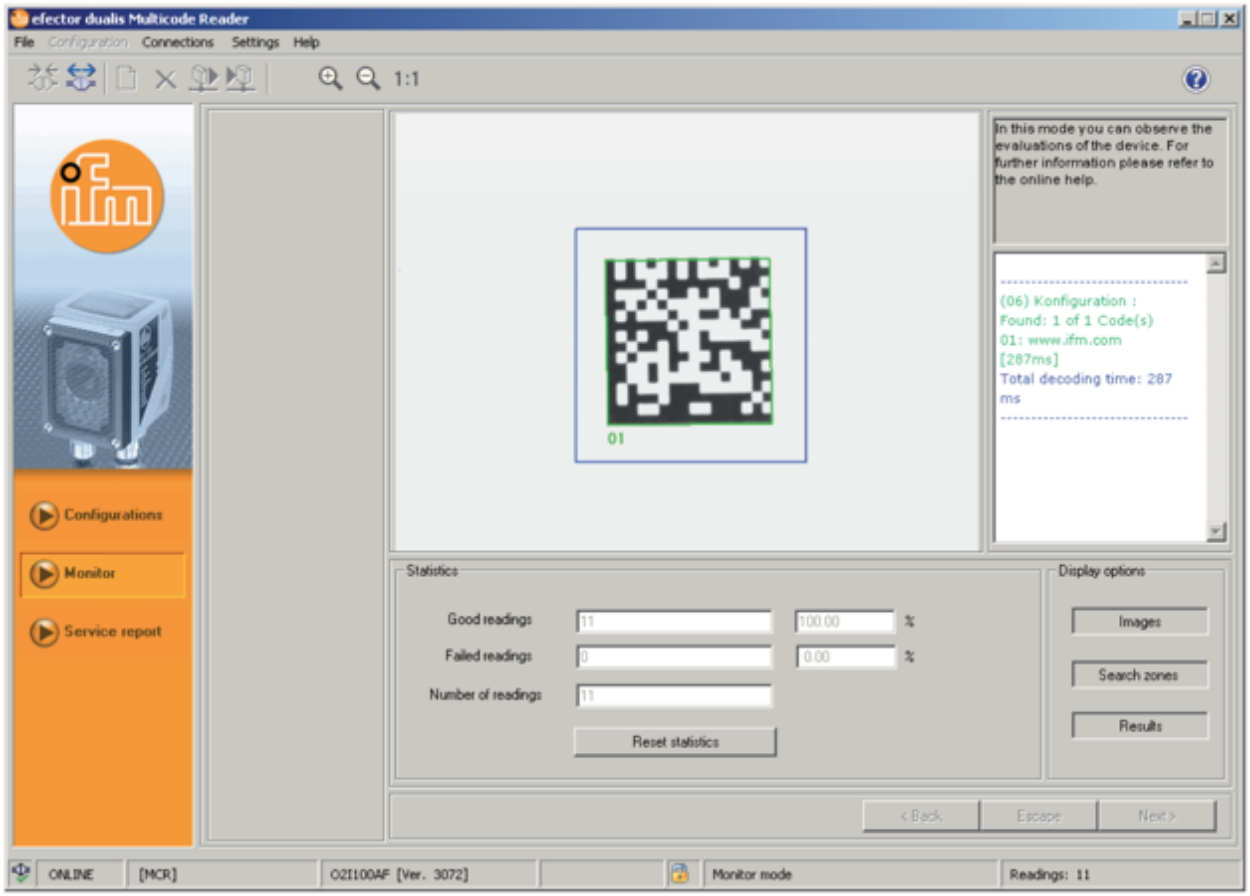


Figure 28 Monitor Mode

Should the read results be saved or assessed, continue with a click on [Service report].

4.1.3 Connecting Hardware and Software

In order to connect the device to the software, first the IP address of the IFM O2I100 must be verified and set.

First select the parameter "IP" (IP address) with [MODE/ENTER] and [SET]. The IP address is processed automatically and shown in 4 groups (A, B, C, D). Then verify the IP address and set with [SET], if necessary.[14]

[14] "E2I200 / V1.3 - ifm." 2010. 4 May. 2015 <http://www.ifm.com/mounting/704743UK.pdf>

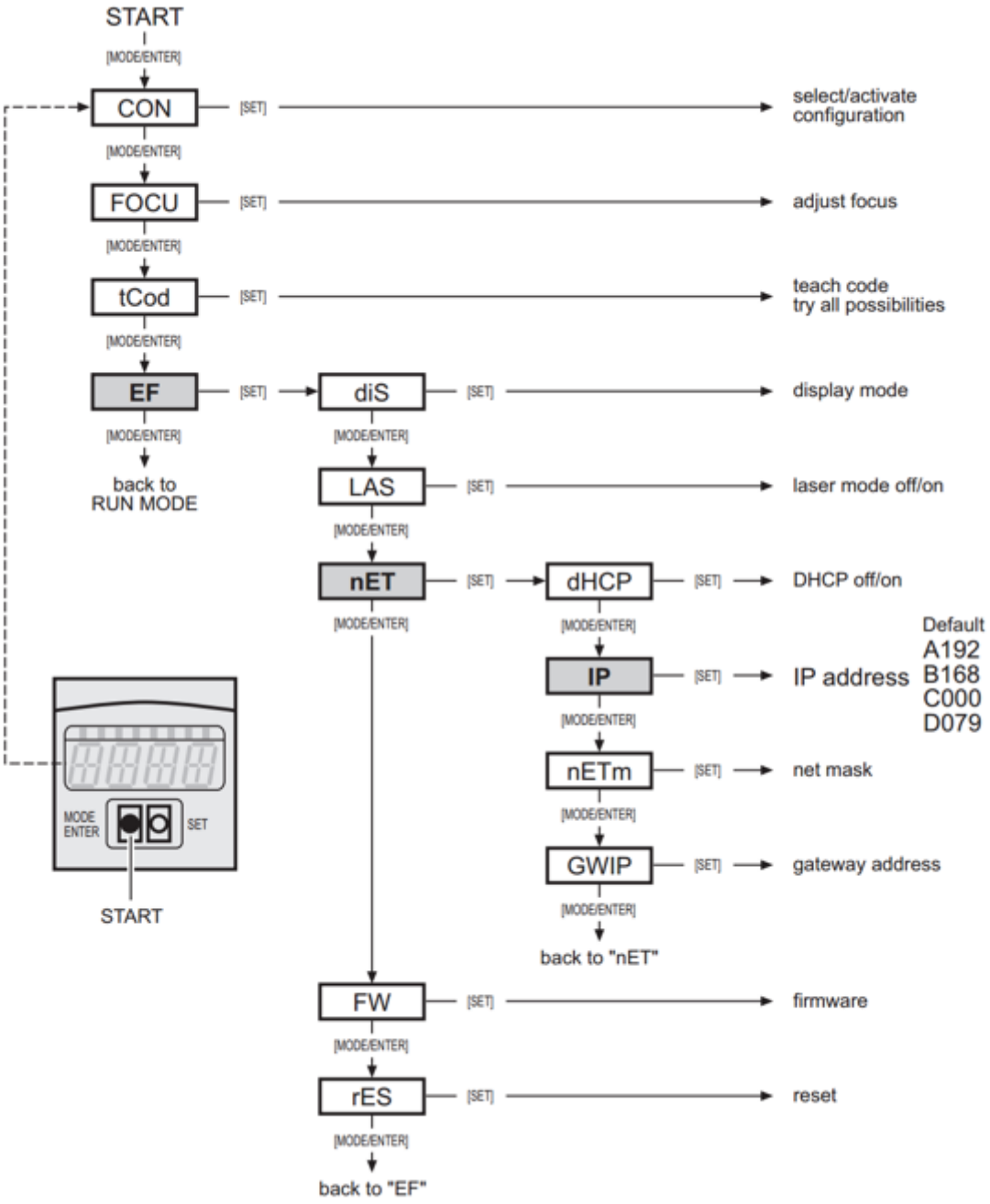


Figure 29 Operating Guide "Dualis Multicode Reader O2I100"

Our next step was to establish the ethernet/IP connection. Firstly click on [Global device settings...], then select [Process interface], then click "EtherNet/IP" in the pulldown menu.

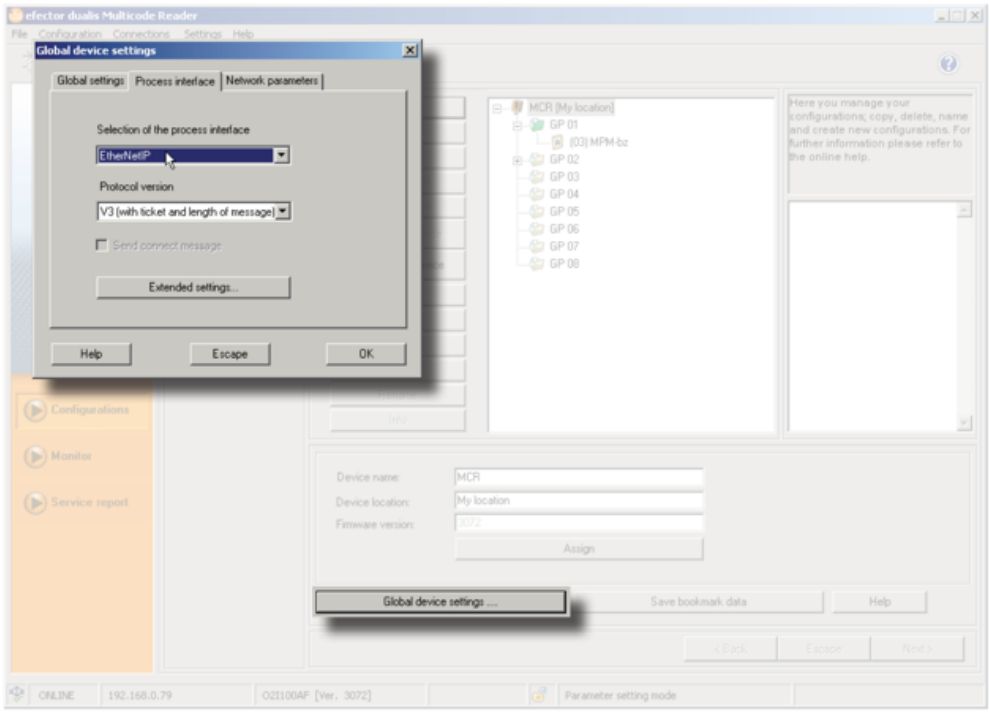


Figure 30 Setting Ethernet Connection

Next we connect the device to the program via IP. First we must click [connections] and then click [IP address…].

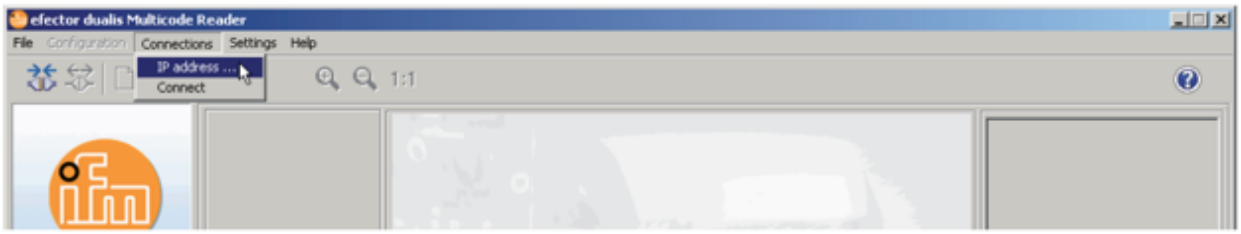


Figure 31 Connections

This can be done three alternative ways. The first is using a bookmark where you navigate through previously saved settings, selecting the device you wish to connect to. The second option is manually enter the known IP address of the device and clicking connect. And finally the third option is a search where you enter IP parameters to search between and available sensors are shown on a list and form here you simply select the sensor you wish to connect. This is done by first clicking [find device] and the then [add] and finally [start search]. You can then select the device you wish to connect.

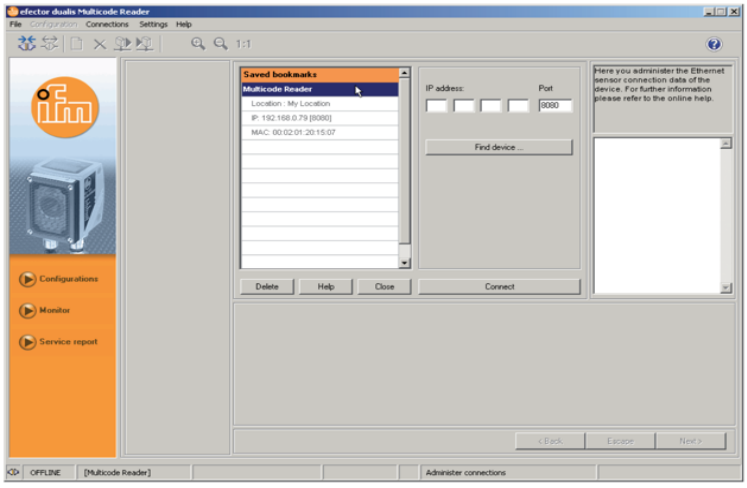


Figure 32a Option 1 - Bookmarks

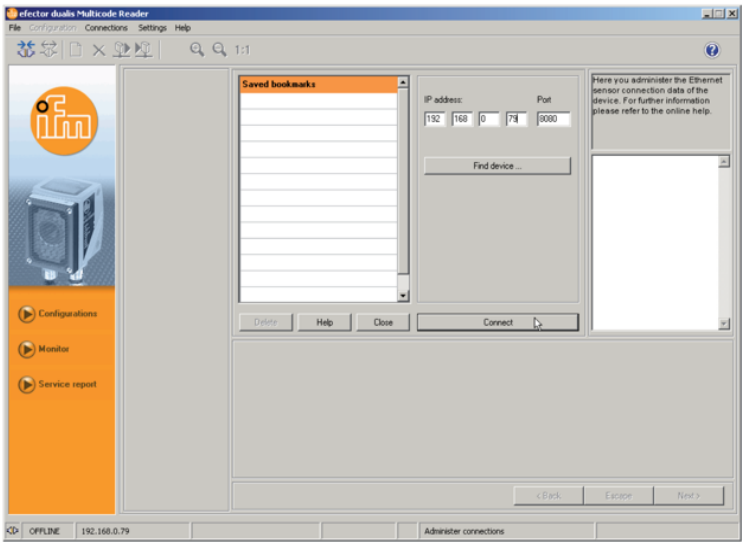


Figure 32b Option 2 - Enter IP

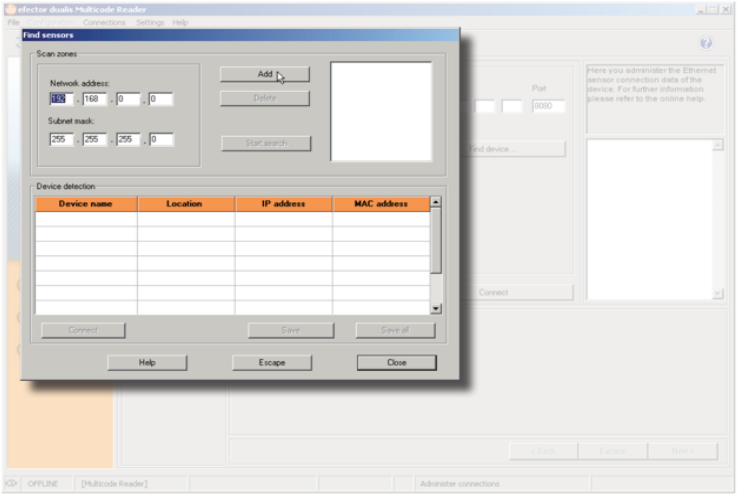


Figure 32c Option 3 - Search IP

4.1.4 Configuration

The device can store up to 32 configuration files (parameter sets). A configuration contains all application-relevant parameters allowing the device to execute the read mode independently. For creating a configuration the user is guided via a predefined navigation. The following settings and indications are polled and defined step by step:

1. Image Quality / Trigger configuration Internal/external illumination, exposure time, parameters for the image quality, trigger type, trigger window.
2. Define code definitions, code recognition criteria, filter functions for the image pre-processing, code-specific optimisation parameters.
3. Process interface Information about the process data, distinction between read operation/comparison/pattern recognition, character strings.
4. Overall function test Final function test with the defined specifications.[15]

Our next step is to activate configuration mode, this is done by clicking on [Configurations]. Then you must acknowledge the warning notification with [OK].

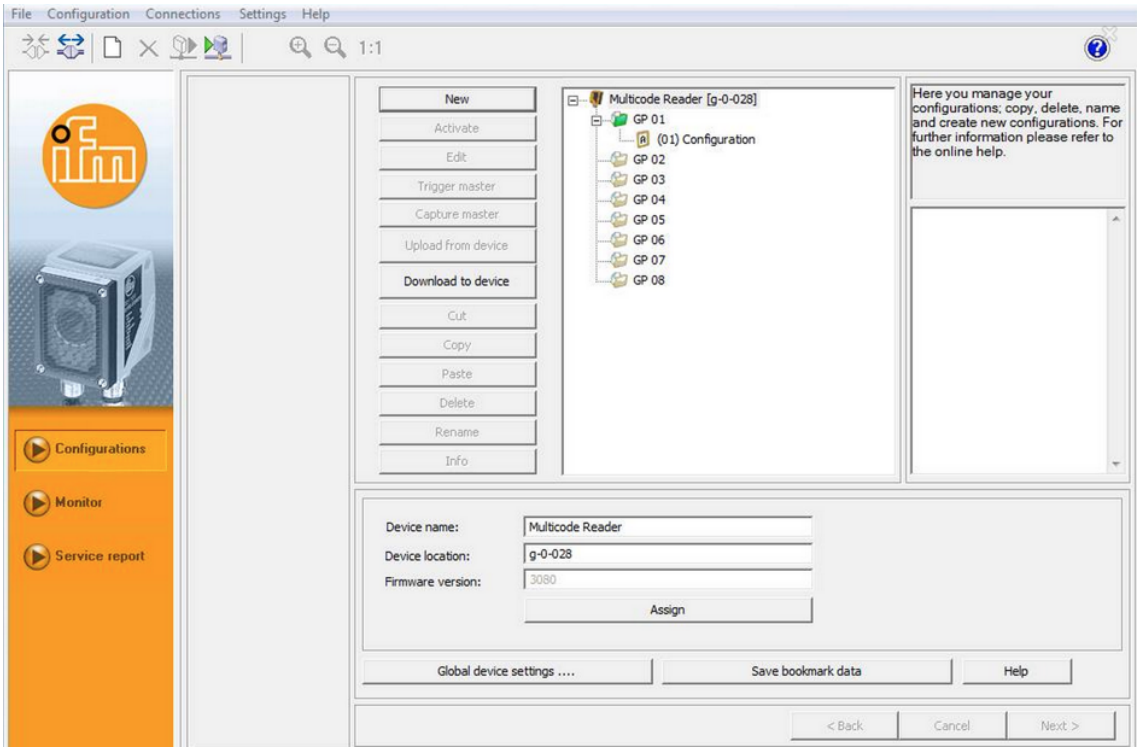


Figure 33 Configuration Page

[15] "E2I200 / V1.3 - ifm." 2010. 4 May. 2015 <http://www.ifm.com/mounting/704743UK.pdf>

By clicking on [New], you can enter a new configuration with a device name and number. Hit [Ok] to save and move on to the image quality / tigger configuration. We used the internal illumination and set the exposure time to best suit the light of the room. A focus screw on the sensor must be adjusted to better focus the image.
With the image quality set we could then move on to trigger configuration and testing the trigger. We set the trigger type to positive edge and using an attached external trigger we could test the trigger using the [Test Trigger], and the device should perform a defined number of successful reads in a certain period of time after a trigger pulse.
We then perform a code recognition process in “Define Code”. When [Read Code] is clicked - if a QR code is present - it read one of the sixteen codes (01-16). All the information contained within the QR code was now available to be output and readable on the screen. No further recognition was needed as our codes were easily read so we could begin with the “Process interface”. In this mode we specify the operating mode and the output text for the process interface then we set the parameters of our string, setting a start, finish and fail string sequence.

Finally we performed an “Overall function test”. Clicking on [Test On] and using the previous settings, the device reads the QR code when the trigger is released. The result field shows:

- Number of codes found (figure)
- Number of codes searched (figure)
- Code string (contents)
- Read time (ms)
- Total time (ms)

Simple click [Next] and [Ok] to save the configuration. The program returns to the directory structure and the newly created configuration is active.

4.1.5 Test

We held the QR codes to the sensor and fitted an external trigger so we could test each QR codes. Each tile was scanned and the QR code was decoded, and if it passed the test, it would read a number from 01-16. It output the information contained within the QR codes.

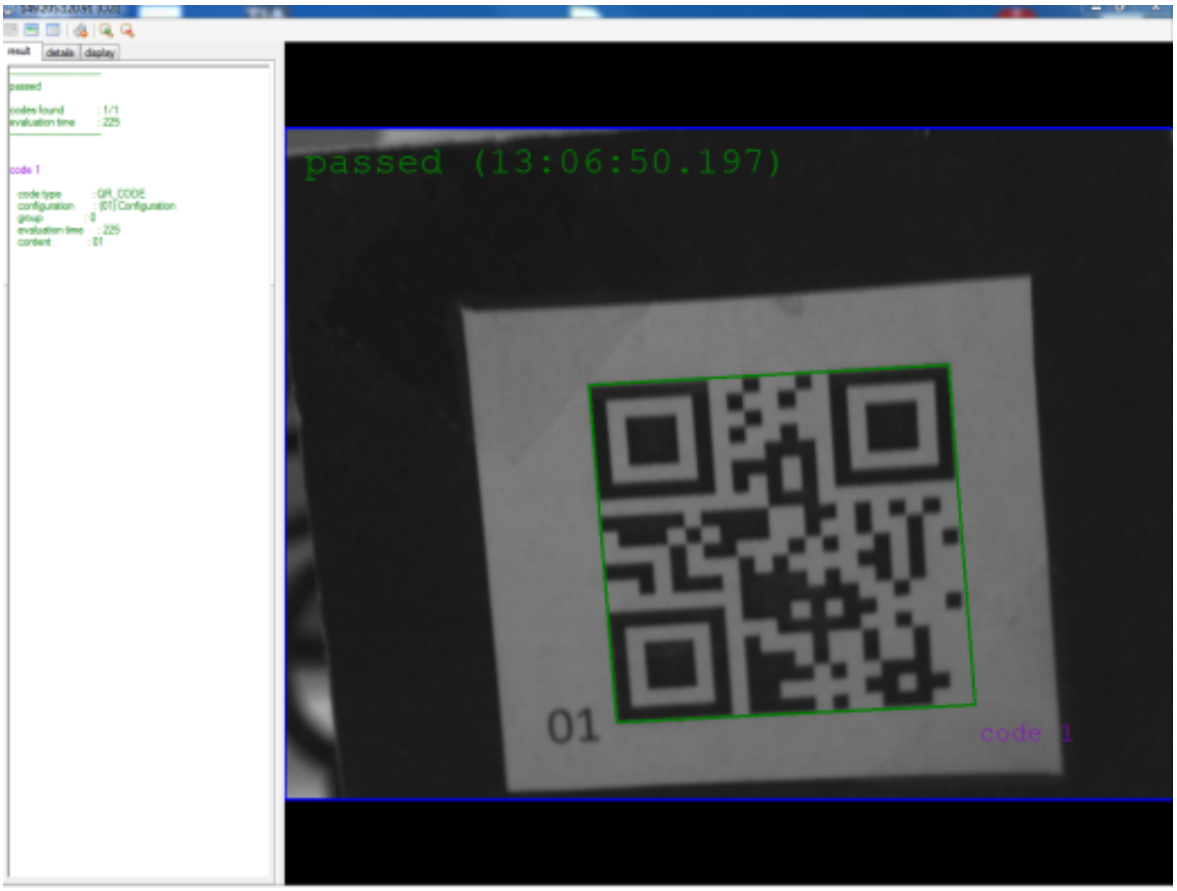


Figure 34 QR Code Passing A Test

4.2 IFM O2D220 Object Recognition Sensor

4.2.1 Software

In conjunction with the object recognition sensor the PC operating program, E2D200, provides the following options, the sensor uses incident light or backlight to detect the image of an object and compares it with the defined contours of one or several models in a reference image. Depending on the degree of conformity, orientation and tolerances the object is classified as being good or bad. It aids creation, administration and deletion of application-specific applications. It also provides real-time monitor mode for set-up and analysis of the application via the service report.[16]

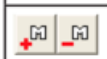
Symbol	Function
	Connect sensor.
	Disconnect sensor.
	Copy existing application to the sensor.
	Loads application from the sensor.
	Enlarge sensor image.
	Reduce sensor image.
	Generates original size of the sensor image.
	Mode for selecting contours.
	Add more contours to selection.
	Select contour with a drag frame.
	Select contour element.
	Edit segment selection.
	Adds all contours / removes all contours.
	Adds all selected contours / removes all selected contours.
	Adds new search zone / removes existing search zone.
	Tests the current model.

Table 3 E2D200 Tool Bar Items

[16] "Programming manual PC operating program for O2D ... - ifm." 2011. 5 May. 2015 <https://www.ifm.com/mounting/704420UK.pdf>

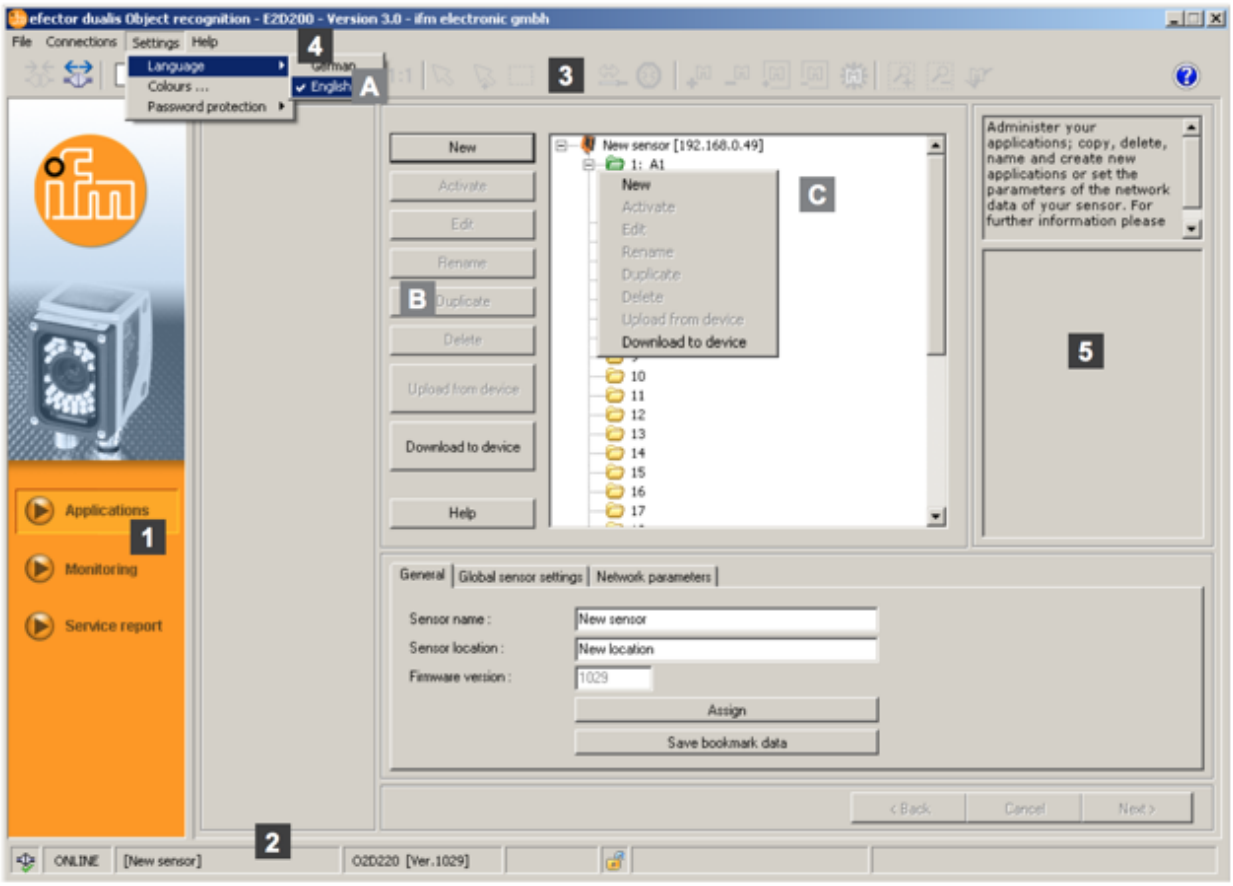


Figure 35 Basic Interface (E2D200)

Item	Operating elements	Contents
1	Mode	• Applications Create, edit, delete etc. applications • Monitor Display or visualisation – of the images (detected object) – of the contours – of the tolerances – of the search zones – of the results • Service report – Display evaluations – Save reports, images, etc.
2	Status bar	• Network status of the device (OFFLINE/ONLINE) • Device name • Article number/production status/firmware of the connected device • Device password protected or unprotected (lock symbol)
3	Tool bar	Buttons (e.g. "Save" or "Connect") Commands that cannot be selected are displayed in grey.
4	Menu bar	Pulldown menus with program functions.
5	Result field	• Display of the set parameters • Display of the results
A/B/C	Selection variants	Identical commands can be selected in different ways (depending on the program function). A = selection via pulldown menu in the menu bar B = selection via button C = selection via context menu (click with right mouse button)

Table 4 Interface Elements & Contents (E2D200)

4.2.2 Connecting Hardware and Software

Firstly as with the IFM O2I100, the O2D220's IP address must be set and verified, and this process is identical to that of the O2I100 and has been previously been explained.

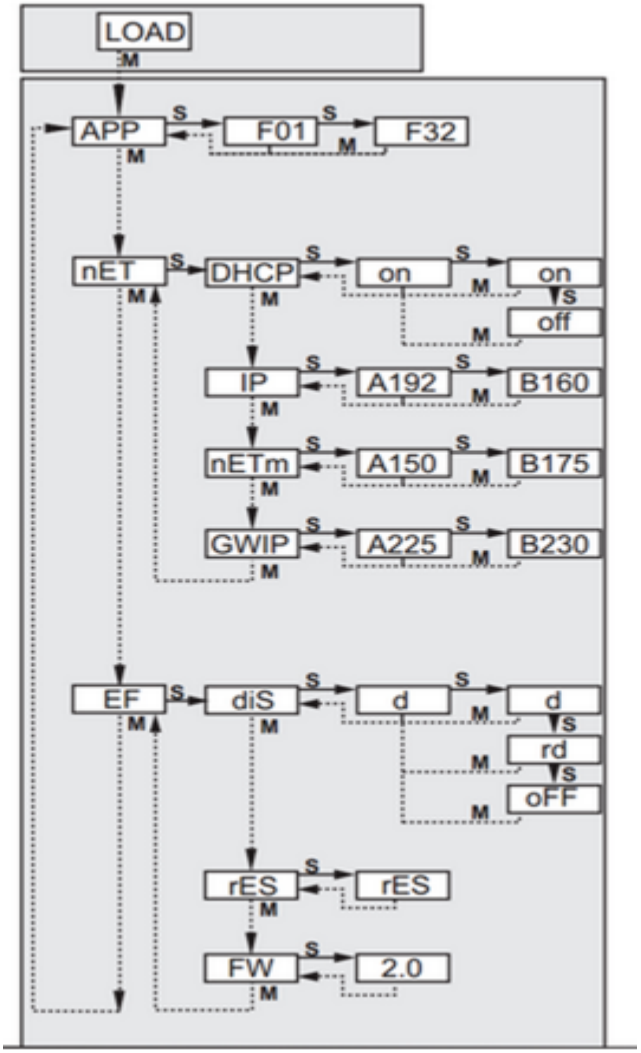


Figure 36 Operating Instructions (O2D220)

The next step is finding the verified IP address on the E2D200 software and this process is also identical to that of the O2I100 and the Dualis software. It can be done one of three ways, by using a bookmark of a previously connected device, a search for all devices between parameterizing IP addresses, or a direct attempt at connection to a known IP address.

4.2.3 Scanning The Contours Of Each Tile

The device can save up to 32 applications (parameter sets). When creating an application the user is guided via a predefined navigation. The following settings and indications are polled and defined step by step:

- Image quality
- Model definition
- Trigger configuration
- Overall function test[17]

In “image quality” the exposure time was set to get the best contrast of each tile. The focus screw on the sensor was also changed until the clearest focus of the image was found.
In “model definition” using the mouse to crop the image, we could highlight the contours of the images on the tiles. Clicking on [Extended options] the sensitivity and contour smoothing could be adjusted. Our settings were usually low sensitivity and smoothing degree at 1, 2 or 3. The reference point was set to center of gravity and click [Next] to move on.

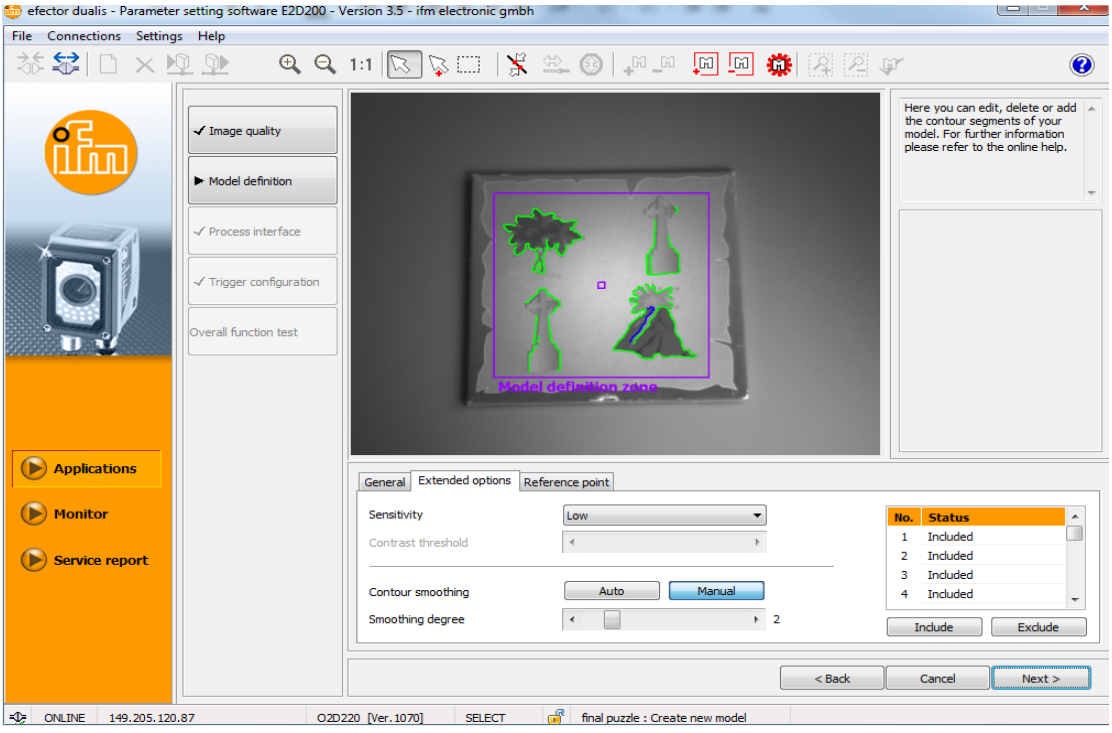


Figure 37 Model Definition Of A Tile

[17] "Programming manual PC operating program for O2D ... - ifm." 2011. 6 May. 2015 <https://www.ifm.com/mounting/704420UK.pdf>

The parameters of the tiles could now be set. First the contour tolerance width was set to 4 and the minimum match was set to 87%. This allowed for a high percentage of contour match usually in the nineties meaning each tile could not be matched with a different tile. The “orientation & symmetry” was set to +180 and -180 so not matter the angle of the tile the sensor could always get a match. The model could now be named, which for each tile was number 01-16.

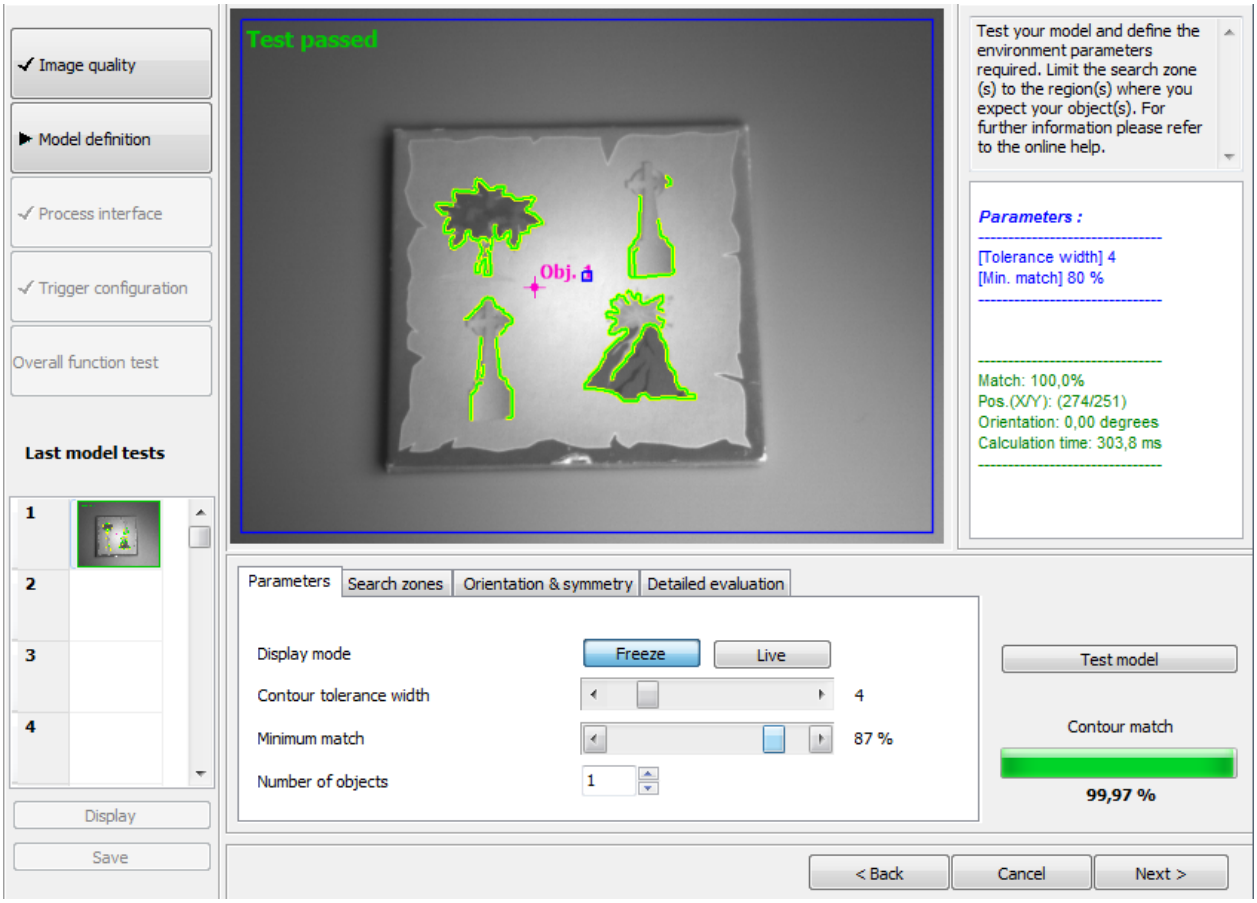


Figure 38 Test Model & Set Parameters

An external trigger was attached so the sensor could be triggered when a tile was ready to be tested. The “trigger configuration” allowed us to set the trigger to positive edge and allow so test to see if the trigger was operating correctly. When the trigger was working we could move on to “overall function test” and could be configured via TCP/IP. Here we could check the parameters we just set and test the tile to see if all was operating before saving the changes to the application.

4.2.4 Test

Once all the faces of each puzzle tile were scanned, a test was done by placing random cards under the O2D220 sensor and triggering the external trigger which was connected to the sensor. We also rotated the cards to make sure the sensor still read at any angle. If the test passed it would read the number of the tile back, the percentage match, and the angle of orientation. If it returned a fail it is because the tile was not read correctly. It was here where we had a lot of trouble as the sensor returned an error of a time-out occurring and all unsaved data would be lost.

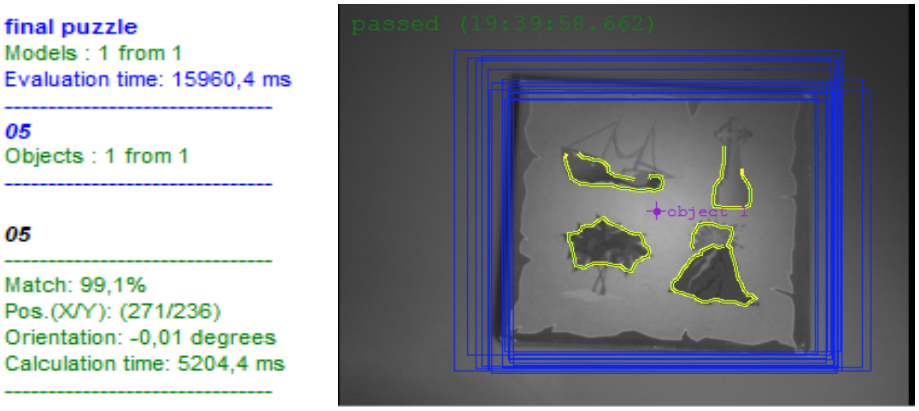


Figure 39 Final Test Match

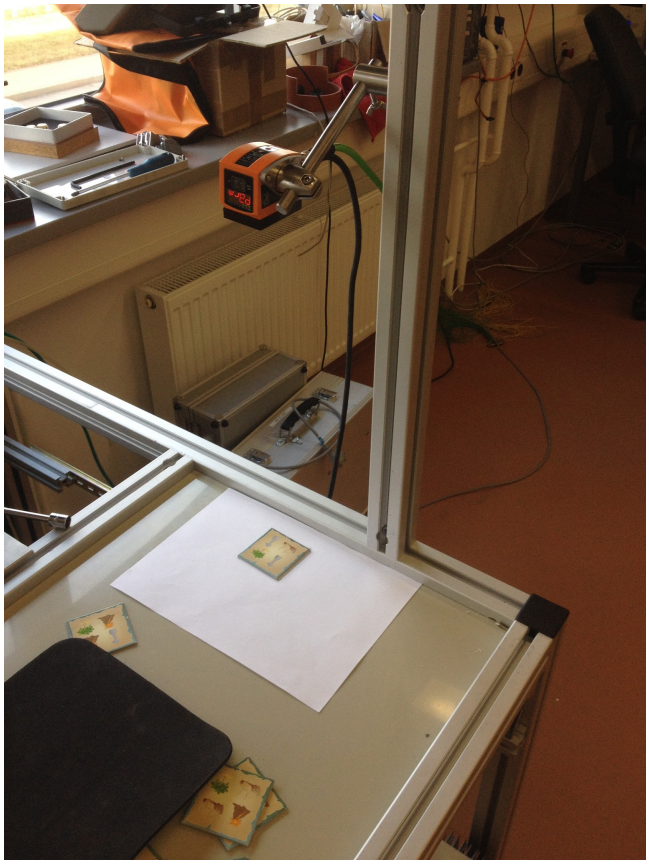


Figure 40 Test Set Up

Chapter 5 Interlinking All Components

5.1 IFM O2I100, IFM O2D220 & Siemens S7-1200

The O2I100 and O2D220 are both connected to the S7-1200 via the internal profinet interface of the CPU. In the S7-1200 program, TIA Portal, the program block O2Ix_PN and O2Dx_PN libraries allow an easy way to activate a parameter set saved in the sensors, and to receive the responsive data from the triggered sensors.

5.1.1 Installation

Both sensors are connected to the internal Profinet interface of the S7-1200 CPU using an Ethernet cable, M12/RJ45, 4 poles connector. If the sensors are directly connected to the interface then a crossover cable is recommended. The next step is to connect TIA Portal to the S7-1200 by changing to [Device view] register in Device configuration. Next open the properties of the Profinet interface and in the IP protocol section set the IP address and Subnet mask of the interface.

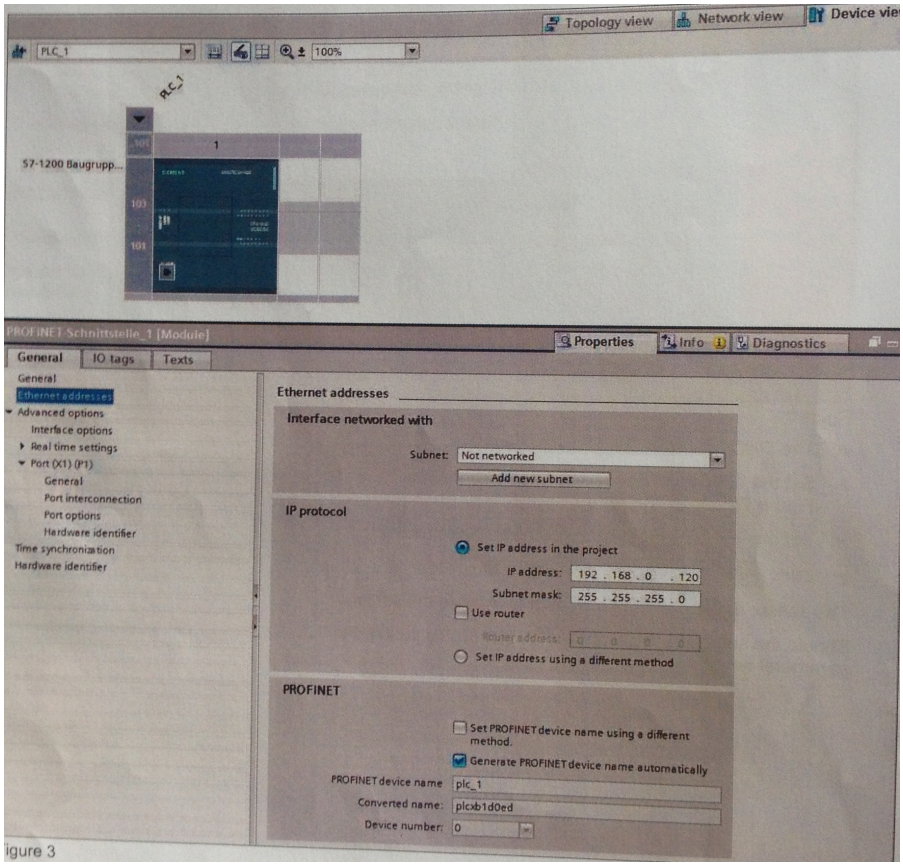


Figure 41 Profinet Interface

5.1.2 Programming

To be independent of existing TIA Portal software, the libraries are delivered as SCL source files, which we downloaded from an IFM support page http://www.ifm.com/ifmus/web/pdownloads020_010_050.htm, and from these two files the program blocks can be generated. In the external source files tab, click [Add new external file], browse to the source and add it to the project. Then right click on the added source file and click [Generate blocks from source]. The program blocks will be generated by TIA Portal. Each library contains a PLC data type, a function and a function block.

The function blocks are the core of the program blocks in the library. It contains the main application and provides the use of functions like riggers to the sensors. The function uses Siemens function blocks TSEND, TRCV and TCON.

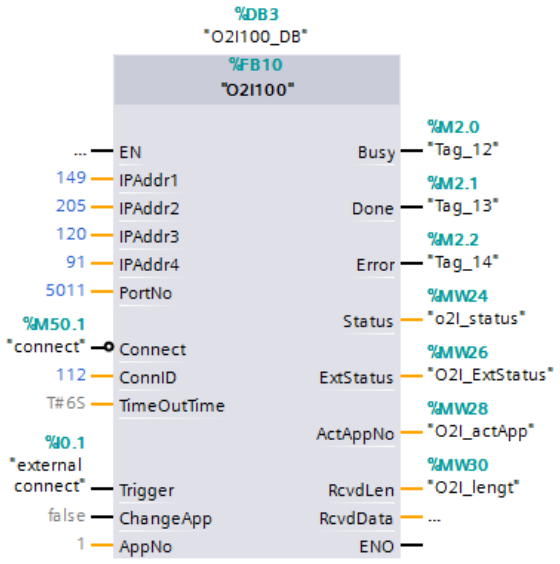


Figure 42 O2I100 Function Block

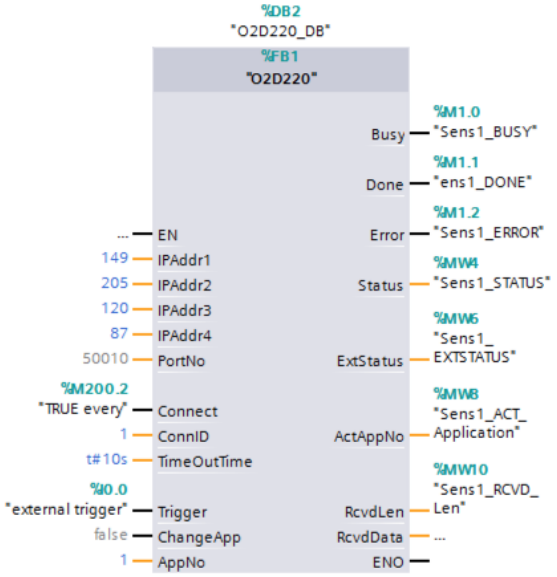


Figure 43 O2D220 Function Block

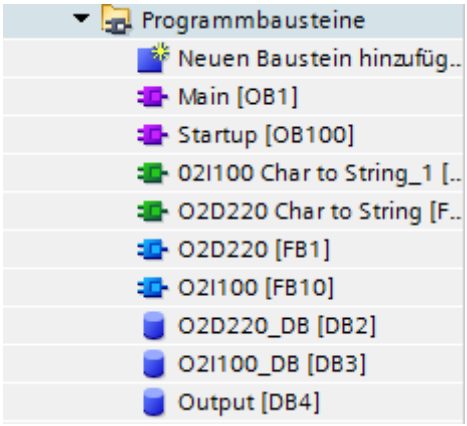


Figure 44 Function Blocks In TIA Portal

Once the output of a sensor is received it is held in a data block in char format. From here it must first be converted to a string and saved to a memory bit. Now it is snipped so we are left with the information we desire. For the O2I100, we only leave the contents of the QR codes which is the number of the tile which is then converted into an integer. And finally because we are left with a number between 01 and 16 we must subtract one from this final outcome each time the sensor has been triggered so that we can use a 4-bit binary output (0-15) to control the RV-E2 robotic arm.

```
1  // CHAR to STRING
2  Chars_TO_Strg(Chars := "O2I100_DB".RcvdData,pChars := 0,Cnt := 64,Strg => "Output"."O2I100 String");
3
4  // Find Model
5  "Output"."O2I100 Model" := MID(IN := "Output"."O2I100 String", L := 2, P := 6);
6
7  // String to Int
8  STRG_VAL(IN := "Output"."O2I100 Model",
9           FORMAT := 00,
10          P := 1,
11          OUT => "Output"."O2I100 Out");
12
13 #werd1:= "Output"."O2I100 Out";
14 STRG_VAL(IN := "Output"."O2I100 Model",
15          FORMAT := 00,
16          P := 1,
17          OUT => "Output"."O2I100 Out2");
18
```

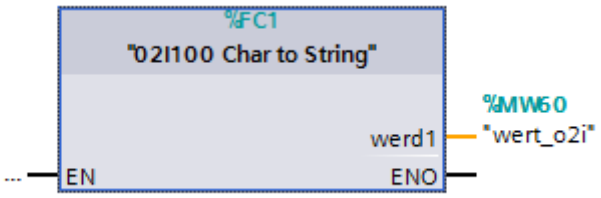


Figure 45 Char to String SCL & FB O2I100 Example

The same is done with the information in the O2D220 data block, first the tile number is converted into an integer, one subtracted to allow binary counting 0-15 instead of 1-16. Next the percentage match is converted to a float or a decimal integer as TIA Portal refers, and finally the full string is stored so that all the data can be displayed on a HMI later.

After both strings have been received the converted and trimmed down the two tile number integers are converted once more into a word and the two are compared as they should match.

```
1  // CHAR to STRING
2  Chars_TO_Strg(Chars := "O2D220_DB".RcvdData,
3                pChars := 0,
4                Cnt := 64,
5                Strg => "Output"."O2D220 String");
6
7  //Find Model
8  "Output"."O2D220 Model" := MID(IN := "Output"."O2D220 String", L := 2, P :=21);
9  // Percentage Error
10 "Output"."O2D220 Percentage error" := MID(IN := "Output"."O2D220 String", L := 4, P := 12);
11
12 //String to Integer
13 STRG_VAL(IN := "Output"."O2D220 Model",
14           FORMAT := 00,
15           P := 1,
16           OUT => "Output"."O2D220 Out2");
17 //output = output2 - 1, to fit binary
18 "Output"."O2D220 Out" := "Output"."O2D220 Out2" -1;
19
20 #werd2 := "Output"."O2D220 Out";
21
22 //String to Integer
23 STRG_VAL(IN := "Output"."O2D220 Percentage error",
24           FORMAT := 00,
25           P := 1,
26           OUT => "Output"."O2D220 Percentage error int");
27
```

Figure 46 Char to String SCL O2D220 Example

5.2 Siemens S7-1200 & Mitsubishi RV-E2 & CR-E116

From previously saving the output, tile number, of the sensor as a 4-bit word in a memory bit, after receiving a match from the comparison, we now set these as the outputs of the plc using 4-bit binary, using an internal conversion method, storing the words in MW%60 and MW%62. There are 4 inputs of the RV-E2 linked to 4 outputs of the S7-1200, and also 4 outputs of the RV-E2 which are are connected to 4 inputs of the the S7-1200. The outputs of the RV-E2 are switched internally by the program that the arm is running, these outputs of the RV-E2 have been set, one to trigger each sensor when in the correct position, internally within the PLC. A third output bit was used from the RV-E2 and was set to control the external LED used for the lighting of the IFM O2D220 object recognition sensor.

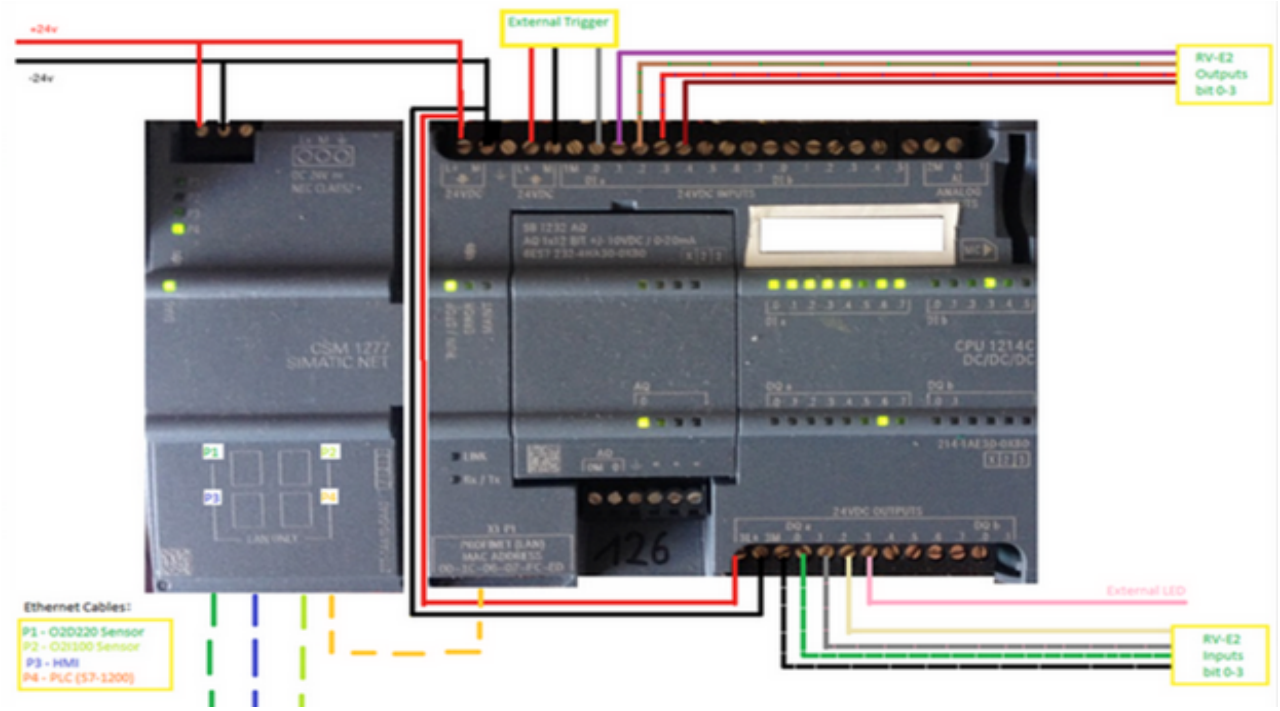


Figure 47 S7-1200 Wiring Diagram

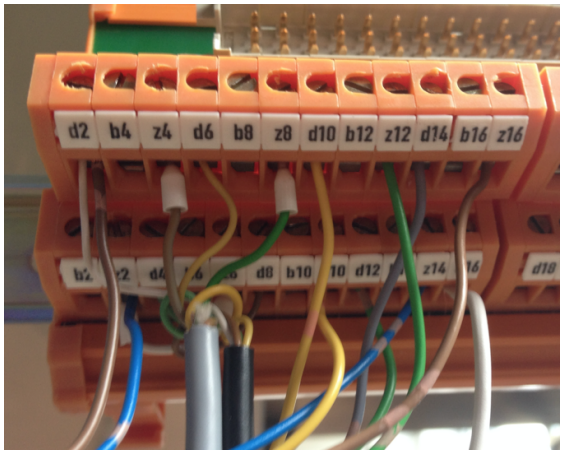


Figure 48 CR-E116 Controller Box Connection Block

For our final program for the RV-E2 Arm and CR-E116 Controller Box we decided, as the output of the S7-1200 PLC is a 4 bit binary number signalled by 4 output cables connected to the 4 input signal bits of the controller box, to use a series of loops to decode the binary input.

There are 16 cycles in total, one for each tile. Each cycle, while picking up each tile we drop the Z axis a further 3mm to reach further into the card holder. This is done with the use of a counter. Binary decoding worked as, first all inputs were read, then in order of 0-3, each bit was checked if it was positive, and if so the program was ordered to jump to a new line number which would have commands to move to a particular position restarting the cycle, upon the last cycle the arm returned to the next position.

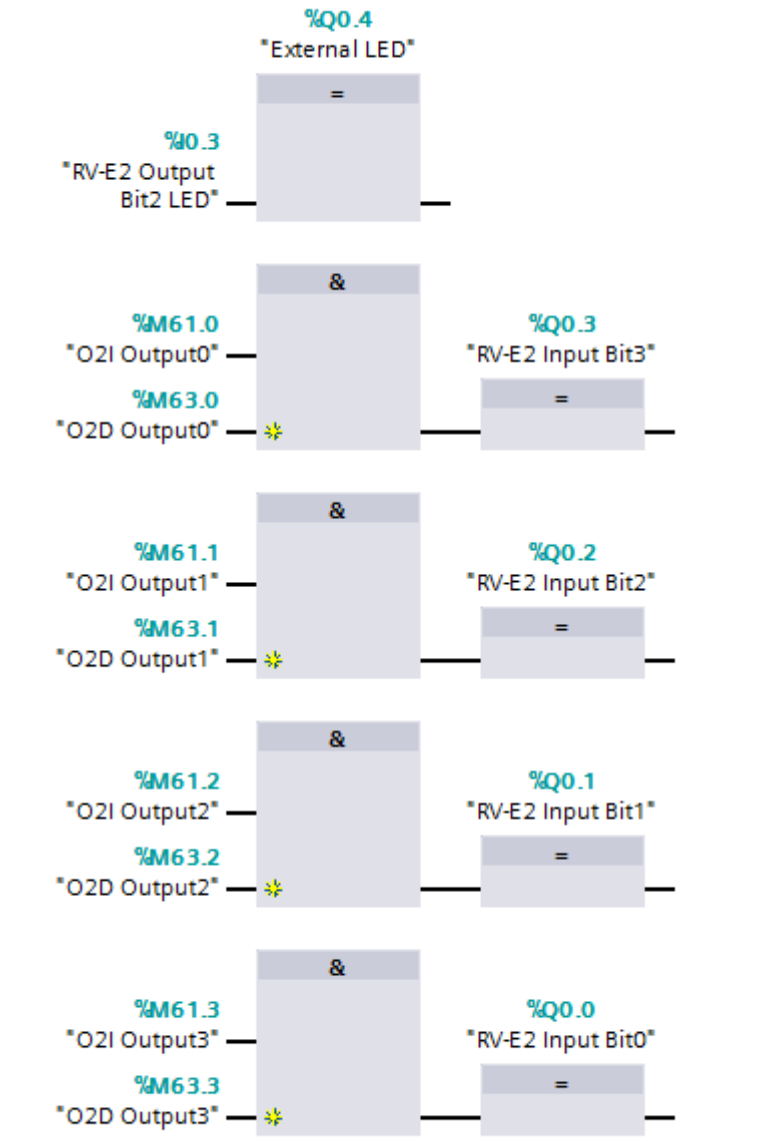


Figure 49 Matching The Outputs

And here is the code used in our program:

```
10 '
20 TL 115
21 SP 14
25 SC 1,16
27 RC 16  ** 16 cycles **
28 SP 14
30 MO 1,O
35 MO 2,O
36 TI 5
37 GT 800 **decide how far to sink**
38 GC
39 DC 1
40 MO 5,C
41 MO 11,C
42 OB +2
43 MO 10,C
47 TI 5
48 GO
49 DS 0,0,30
50 MO 11,O
51 OB +1
52 TI 5
53 OB -1
54 TI 20
55 MO 12,O
56 MO 10,O
57 TI 5
58 GC
59 DS 0,0,30
60 OB -2
65 MO 3,C
66 TI 5
67 OB +0
68 TI 5
70 OB -0
71 ID
72 TB +0,370
73 TB +1,215
74 TB +2,140
75 TB +3,115
76 MO 4,C
77 SP 10
78 MO 21,C
79 TI 5
80 GO
85 DS 0,0,30
86 NX
87 NT
88 ED
115 MO 4,C
116 SP 10
120 MO 22,C
125 TI 5
130 GO
131 DS 0,0,30
135 NX
136 NT
137 ED
140 TB +3,185
150 MO 4,C
151 SP 10
155 MO 23,C
160 TI 5
165 GO
170 DS 0,0,30
180 NX
181 NT
182 ED
185 MO 4,C
186 SP 10
190 MO 24,C
195 TI 5
200 GO
205 DS 0,0,30
210 NX
211 NT
212 ED
215 TB +2,290
216 TB +3,260
220 MO 4,C
221 SP 10
225 MO 25,C
230 TI 5
235 GO
240 DS 0,0,30
245 NX
246 NT
247 ED
260 MO 4,C
261 SP 10
265 MO 26,C
270 TI 5
275 GO
280 DS 0,0,30
285 NX
290 TB +3,340
300 MO 4,C
301 SP 10
305 MO 27,C
310 TI 5
315 GO
320 DS 0,0,30
325 NX
326 NT
327 ED
340 MO 4,C
341 SP 10
345 MO 28,C
350 TI 5
355 GO
360 DS 0,0,30
365 NX
366 NT
367 ED
370 TB +1,560
371 TB +2,460
373 TB +3,430
380 MO 4,C
381 SP 10
385 MO 29,C
390 TI 5
395 GO
400 DS 0,0,30
405 NX
406 NT
407 ED
430 MO 4,C
431 SP 10
435 MO 30,C
440 TI 5
445 GO
450 DS 0,0,30
455 NX
456 NT
457 ED
460 TB +3,530
480 MO 4,C
481 SP 10
485 MO 31,C
490 TI 5
495 GO
500 DS 0,0,30
505 NX
506 NT
507 ED
530 MO 4,C
531 SP 10
535 MO 32,C
540 TI 5
545 GO
550 DS 0,0,30
555 NX
556 NT
557 ED
560 TB +2,660
565 TB +3,630
580 MO 4,C
581 SP 10
585 MO 33,C
590 TI 5
595 GO
600 DS 0,0,30
605 NX
606 NT
607 ED
630 MO 4,C
631 SP 10
635 MO 34,C
640 TI 5
645 GO
650 DS 0,0,30
655 NX
656 NT
657 ED
660 TB +3,730
680 MO 4,C
681 SP 10
685 MO 35,C
690 TI 5
695 GO
700 DS 0,0,30
705 NX
706 NT
707 ED
730 MO 4,C
731 SP 10
735 MO 36,C
740 TI 5
745 GO
750 DS 0,0,30
755 MO 4,O
760 MO 1,O
765 NX
770 NT
800 CP 1
850 EQ 16,1100
860 EQ 15,1150
870 EQ 14,1200
880 EQ 13,1250
890 EQ 12,1300
900 EQ 11,1350
910 EQ 10,1400
920 EQ 9,1450
930 EQ 8,1500
940 EQ 7,1550
950 EQ 6,1600
960 EQ 5,1650
970 EQ 4,1700
980 EQ 3,1750
990 EQ 2,1800
1000 EQ 1,1850
1010 EQ 0,1900
1100 DS 0,0,-3
1110 GT 38
1150 DS 0,0,-6
1160 GT 38
1200 DS 0,0,-9
1210 GT 38
1250 DS 0,0,-12
1260 GT 38
1300 DS 0,0,-15
1310 GT 38
1350 DS 0,0,-18
1360 GT 38
1400 DS 0,0,-21
1410 GT 38
1450 DS 0,0,-24
1460 GT 38
1500 DS 0,0,-27
1510 GT 38
1550 DS 0,0,-30
1560 GT 38
1600 DS 0,0,-33
1610 GT 38
1650 DS 0,0,-36
1660 GT 38
1700 DS 0,0,-39
1710 GT 38
1750 DS 0,0,-42
1760 GT 38
1800 DS 0,0,-45
1810 GT 38
1850 DS 0,0,-48
1860 GT 38
1900 NT
1910 ED
```

Figure 50 Final COSIPROG Program

5.3 Siemens S7-1200 & Siemens Touch HMI KTP-600

Once the IFM O2D220 has scanned the tile, and the information has been trimmed and converted into the correct data-type within the PLC. The PLC then sends the information required to the HMI. Both the HMI and PLC are Siemens products and both are programmed with TIA Portal, this allowed for easy interactions between them. On the HMI we created a welcoming page with two choices, an "About Us" page and a "Puzzle Information" page. The "Puzzle Information" page is where we sent the information from the PLC. On this page the tile number, percentage match and output string could all be viewed.

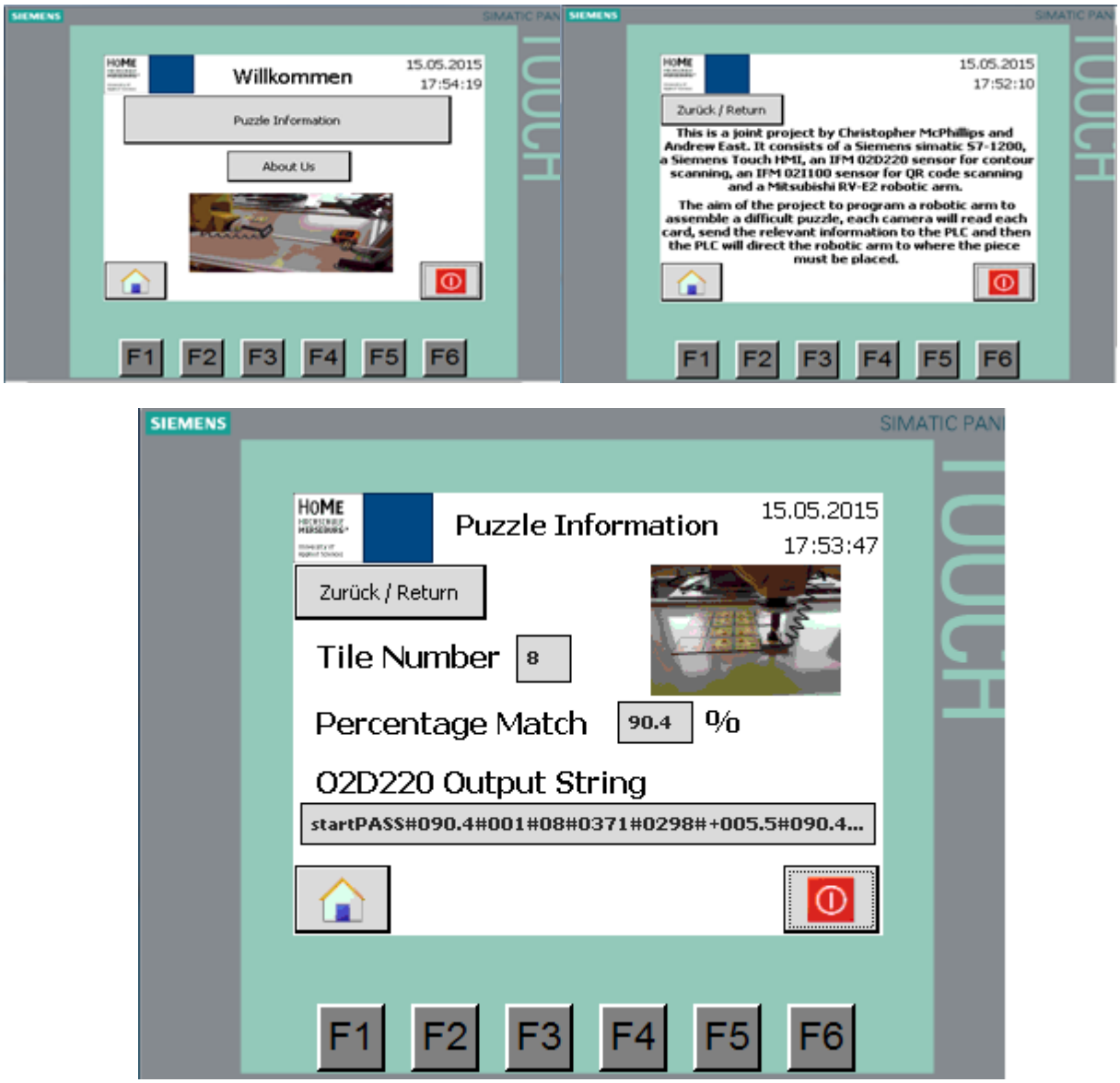


Figure 51 HMI Pages

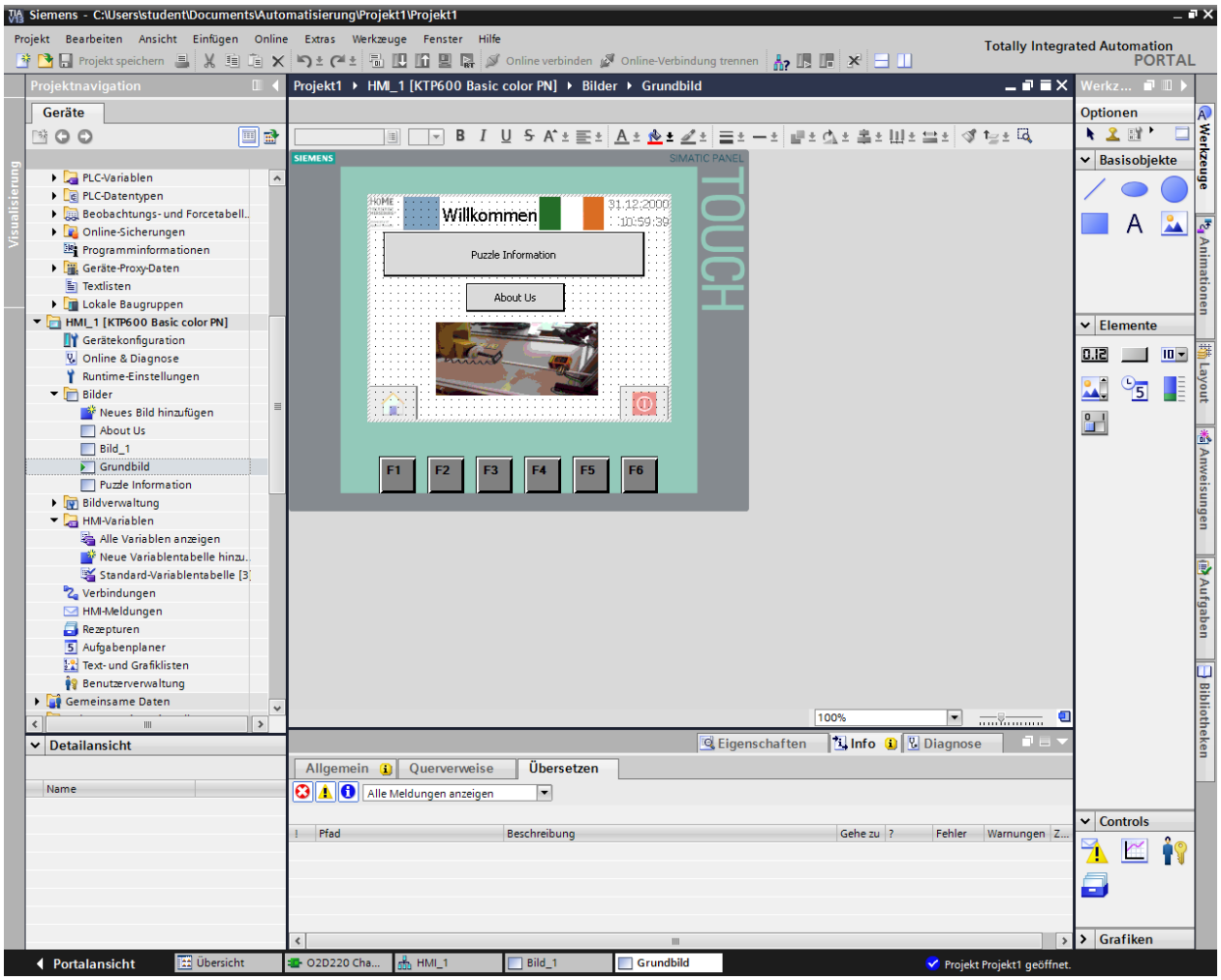


Figure 52 HMI KTP-600 in TIA Portal

The HMI was connected via Ethernet cable and configured by setting the IP address. After which we created the pages and also set variable tags to each the string, percentage match and tile number so that they could be fetched by the HMI within the data block of the PLC program files and displayed to the user.

5.4 Conclusion

The puzzle tiles were placed inside the tile holder and the robot was turned on. One by one the tiles were taken out of the holder and brought to each of the IFM vision sensors. These sensors then communicated with the PLC and if both sensors matched and identified the same tile, the robotic arm (RV-E2) would move the tile to its position inside the 4x4 puzzle. Here is a youtube link to a video of our final project, https://www.youtube.com/watch?v=yjkZb0mFjC0.

This was the final outcome of our project, the machine was able to determine the final correct positions of all the tiles. Each sensor was used to check and verify the correct tile before it was used in the puzzle. The robot can carry out the same task, no matter the order of the tiles, each time it is used.

The communications of the project were quite difficult due to one sensor (IFM O2D220), it was timing out each time we tried to insert the applications. By changing firmware, ethernet cables and eventually computers we managed to get it fully operational. The other sensor (IFM O2I100) worked perfectly from the beginning and with the two sensors working, it wasn't long until we managed to get both communicating and integrating with the PLC.

To get the correct information from each of the sensors, we had to write code to the PLC that would only extract the information that we would need, by conversions and snipping. This SCL was similar to code we have used before in previous projects but different to the code used in COSIPORG for programming the robotic arm.

Being in Germany for our final year project was great, but also challenging at times. Communication was hard at times, although everyone was a great help. Some of the programs we used were in German which made us feel we could have used it better and more accurately if we understood it fully. It was also a great opportunity to work and learn outside of Ireland and also use new programmes that we wouldn't get the opportunity to use back home.

Please check out our blog which followed us through this project here, https://projectmerseburg.wordpress.com/.

Bibliography / References

1. "What is Transmission Control Protocol (TCP)? Webopedia." 2002. 28 April 2014 <http://www.webopedia.com/TERM/T/TCP.html>

2. "TIA Portal - Totally Integrated Automation Portal - The Future ..." 2010. 28 April 2014 <http://www.industry.siemens.com/topics/global/en/tia-portal/pages/default.aspx>

3. "Operating instructions Multicode Reader O2I10x - ifm." 2011. 28 April 2014 <https://www.ifm.com/mounting/704247UK.pdf>

4. "http://www.simatec.ir/ 2015-01-14T04:54:21+00:00 monthly ..." 2014. 28 April 2014 <http://www.simatec.ir/sitemap.xml>

5. "1st Generation Device Overview - Operator devices - Siemens." 2014. 28 April 2014 <http://w3.siemens.com/mcms/human-machine-interface/en/operator-interfaces/basic-panel/devices-first-generation/pages/default.aspx>

6. "Mitsubishi Robot RV-E2 Sale - Mitsubishi Robots Sale." 2010. 28 April 2014 <http://www.mitsubishirobots.com/RV-E2.html>

7. "Intro to Ladder Logic - Transwiki." 2015. 29 Apr. 2015 <https://wiki.xtronics.com/index.php/Intro_to_Ladder_Logic>

8. Zhang, Wei, Wolfgang A Halang, and Christian Dietrich. "Specification and verification of applications based on function blocks." *Component-Based Software Development for Embedded Systems* (2005): 8-34.

9. "Structured Control Language (SCL) for S7-300 ... - Siemens." 2015. 29 Apr. 2015 <https://cache.industry.siemens.com/dl/files/188/1137188/att_27471/v1/SCLV4_e.pdf>

10. "MOVEMASTER." 2011. 30 Apr. 2015 <http://dim.usal.es/eps/im/roberto/pesados/rob/RV-E2_especifications_manual.pdf>

11. "mitsubishi robot movemaster manuals - Mitsubishi RV-3AL." 2008. 30 Apr. 2015 <http://www.mitsubishirobot.com/ta112.html>

12. "What is QR code (quick response code)? - Definition from ..." 2012. 4 May. 2015 <http://whatis.techtarget.com/definition/QR-code-quick-response-code>

13. "E2I200 / V1.3 - ifm." 2010. 4 May. 2015 <http://www.ifm.com/mounting/704743UK.pdf>

14. "E2I200 / V1.3 - ifm." 2010. 4 May. 2015 <http://www.ifm.com/mounting/704743UK.pdf>

15. "E2I200 / V1.3 - ifm." 2010. 4 May. 2015 <http://www.ifm.com/mounting/704743UK.pdf>

16. "Programming manual PC operating program for O2D ... - ifm." 2011. 5 May. 2015 <https://www.ifm.com/mounting/704420UK.pdf>

17. "Programming manual PC operating program for O2D ... - ifm." 2011. 6 May. 2015 <https://www.ifm.com/mounting/704420UK.pdf>