Bachelor thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of control engineering

Device for hand guiding of an industrial robot

Voronov Serhii

Supervisor: Ing. Pavel Burget, Ph.D.
May 2022

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Voronov Serhii**	Personal ID number:	**484957**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Control Engineering**		
Study program:	**Cybernetics and Robotics**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Device for hand guiding of an industrial robot

Bachelor’s thesis title in Czech:

Zařízení pro ruční navádění průmyslového robota

Guidelines:

The purpose of this thesis is to design and implement a controller for the hand-guided industrial robot. The device will be mounted on a handle used as a hand-guiding tool for the robot. With an integrated camera, the controller will be able to position the robot's end-effector precisely.
The connection to the robot controller will be over the Profinet interface; there will also be an alternative connection over 5G wireless communication.
1) Using a prototype board with Raspberry Pi (RPi) module, integrate Profinet Device stack and test it with an industrial Profinet PLC controller.
2) Using a prototype board with RPi and a 5G modem module, connect the device to the existing 5G SA network in Testbed for Industry 4.0.
3) Create a HW design of a complete device containing RPi compute module, camera, display, 5G modem, and Ethernet interface. Consider the space constraints of the robot handle and its coexistence with the robot end effector.
4) Implement a prototype of the navigating algorithm that will help an operator guide the robot to the desired location of the objects detected below the end-effector.

Bibliography / sources:

[1] RT-LABS. (2021, May 4). P-net Profinet Device Stack. p-net Profinet device stack - p-net documentation. Retrieved January 28, 2022, from https://rt-labs.com/docs/p-net/_copied/README.html
[2] Pigan, R., & Metter, M. (2006). Automating with PROFINET: Industrial communication based on industrial ethernet. Publicis.
[3] Edric Szmynet (http://www.szmynet.com/). (2021, December 9). 5G. SIMCom. Retrieved January 28, 2022, from https://www.simcom.com/product/SIM8200EA_M2.html
[4] Raspberry Pi. (2021, October 12). RPi compute module 4. Raspberry Pi. Retrieved January 28, 2022, from https://www.raspberrypi.com/products/compute-module-4/?variant=raspberry-pi-cm4001000

Name and workplace of bachelor’s thesis supervisor:

Ing. Pavel Burget, Ph.D.   Testbed CIIRC

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **31.01.2022**   Deadline for bachelor thesis submission: **20.05.2022**

Assignment valid until:
by the end of summer semester 2022/2023

Ing. Pavel Burget, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Acknowledgements

I would like to say a big thank you to Pavel Burget, who proposed such an interesting topic and supported me in every possible way in its implementation. I would also like to say thanks to Vojtech Sustr, who is a software developer of the Delta robot and cooperated with me in every possible way in the implementation process. Also a great contribution was made by my colleague David S. Martinez who helped me a lot with AI and Computer vision. Also many thanks to my colleague Lukas Lilek who was the author of the 3D model for this project. Also thanks to my Ukranian friend Mykhailo Fursin who helped me and advised me on the part of creating my own PCB. And big thanks to Elizaveta Isianova for supporting me on the whole developing process.

Declaration

I declare that this work is all my own work and I have cited all sources I have used in the bibliography.

Prague, May 20, 2022

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 20. května 2022

Abstract

The topic of the bachelor thesis is to design and implement a controller for hand-guiding of an industrial robot with parallel kinematics. In addition to hardware design for the controller, the solution also implements a control algorithm to hand guide the robot that helps the operator position the robot's end effector to the desired position. A neural network with computer vision helps to detect objects and their positions on the conveyor. The developed controller is then connected to the PLC using Profinet and to the GPU server using 5G communication.

Keywords: industrial robot, computer vision, PCB development, Delta robot

Supervisor: Ing. Pavel Burget, Ph.D.
Testbed CIIRC

Abstrakt

Tématem bakalářské práce je navrhnout a implementovat kontrolér pro ruční navádění průmyslového robota s paralelní kinematikou. Kromě návrhu hardwaru pro kontrolér se v řešení také implementuje algoritmus řízení robota Delta, který pomáhá operátorovi navádět koncový efektor robota do požadované pozice. Implementována neuronová síť a počítačové vidění pomáhá v detekci objektů na dopravníku. Vyvinutý kontrolér ručního navádění je následně propojen s PLC pomocí komunikace Profinet a na GPU server pomocí 5G komunikace.

Klíčová slova: průmyslový robot, počítačové vidění, vývoj PCB, Delta robot

Překlad názvu: Zařízení pro ruční navádění průmyslového robota

Contents

Figures

Tables

Chapter 1
Introduction

Modern technologies, particularly robotics, have long begun to be introduced into all spheres of human activity, from everyday tasks to industrial production. The development of robotics expands the functionality of devices, which leads to a need for increased computing power. However, the desired computational capabilities of modern processors and batteries have already started to exceed the limits of the physical design of devices. One of the solutions to this problem is using 5G technologies that allow one to separate the computing tasks from the device and carry them out on a remote server. The 5G enables data transmission at high speed, so the functional processes of the device are not slowed down. The final product of my work represents an example of this concept.

This Bachelor thesis aims to design and implement a controller for the hand-guiding application using industrial robot with parallel kinematics. This robot is equipped with force and torque sensor which provides the necessary feedback for hand-guiding application. A camera which will be implemented to the controller will provide better positioning of the robot's tool center point and will be useful for teaching new pick and place positions. The developed controller must be integrated to the actual solution with a PLC controller which is using Profinet communication protocol. An optional 5G communication will be also integrated to this solution and connected to the existing 5G SA network in Testbed for Industry 4.0 at Czech Institute of Informatics, Robotics and Cybernetics (CIIRC). In addition to developing a custom hardware solution and implementing the RPi computing module, camera, display, 5G modem, and Ethernet interface, a navigation algorithm will also be designed and implemented to help the operator guide the robot to the desired location of objects detected under the robot's end effector.

Motivation

This project is focused on extending the existing handle of the Delta robot in Testbed for Industry 4.0 in CIIRC to achieve additional functionality of image processing and flexible data transfer among the tool and the hand-guiding controller. It can significantly improve the programming speed of the delta robot as well as eliminate the necessity of manual programming during the preparation of the robot trajectories.

This project combines the knowledge of hardware development, Linux administration, Python, and C coding, integrating communication protocols such as Profinet and 5G, neural networks, and computer vision.

Chapter 2

Hardware implementation

2.1 Project architecture

The controller works with many systems at different levels (2.1). The main controller RPi CM4 is connected to several communication interfaces: 5G for communication and outsourcing of neural network calculations and computer vision, Ethernet for communication via Profinet with a PLC controller, and optionally – WiFi for convenient SSH access (for debugging purposes) from a local network or via VPN. The controller is equipped with a LCD screen for a convenient visualisation of the image processing made by neural network. The image stream comes from the standard RPi camera V2, which is connected to the RPi.

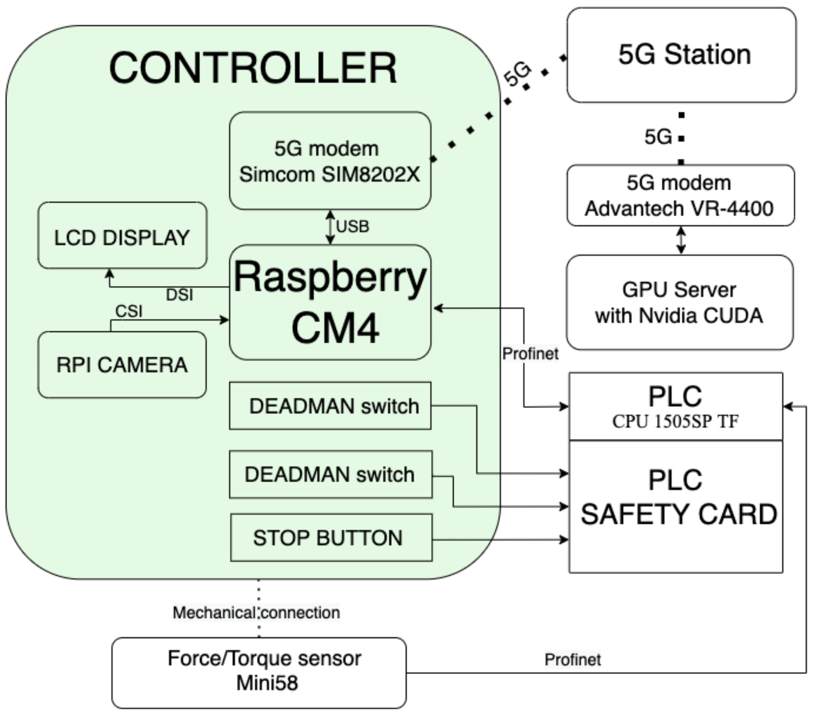


Figure 2.1: Block scheme of the project

Figure 2.2 shows the physical representation of the project. The 3D model is attached for a deeper understanding of the setup. The hand-guided controller is fixed on top of the 6-axis force/torque sensor. This way a controller is able to receive data about the external force applied to itself by the operator.

Under the gripper, there are two platforms on conveyor shuttles, which transport different types of objects placed in four possible predefined positions.

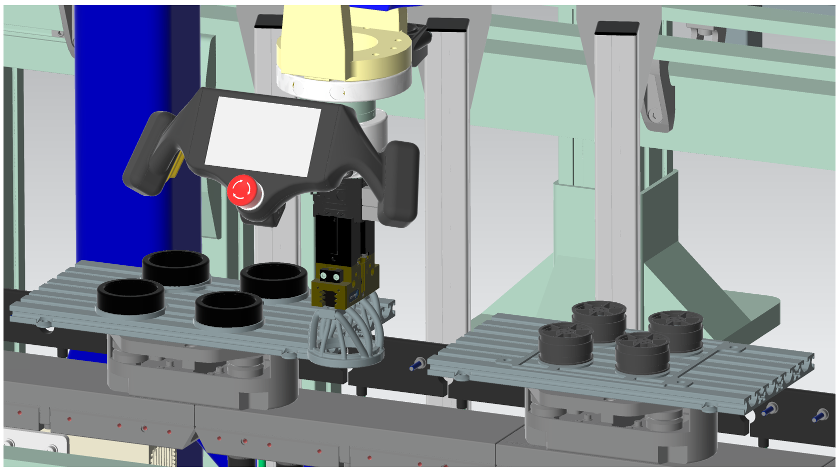


Figure 2.2: Physical representation of controller's placement on the robot's end-effector.

2.2 Linux microcomputer selection research

Today the most popular and affordable Linux microcomputers are the world famous Raspberry Pi version 4 and NVIDIA Jetson microcomputers. While researching the available options for the microcomputer, I came across the RPi Compute Module 4 (CM4), which was released in 2020. It was suitable for the purpose of this project, which is an embedded version of the RPi without interfaces connectors. Despite the fact, that it was almost impossible to buy it in Europe in 2022 due to the COVID-19 and outages in processor manufacturing, I managed to find an e-shop outside Europe where there was still the possibility to buy a CM4 module.

(a) : RPi 4 **(b) :** Jetson Nano **(c) :** RPi CM4

Figure 2.3: Main embedded platforms on the market.

	RPi 4	RPi CM 4	Jetson Nano
CPU	4-Core ARM Cortex-A72 64-bit, 1.5 Ghz	4-Core ARM Cortex-A72 64-bit, 1.5 Ghz	4-core ARM Cortex-A57 64-bit, 1.42 Ghz
GPU	Broadcom Video-Core VI (32-bit)	Broadcom VideoCore VI (32-bit)	128-core NVIDIA Maxwell, CUDA
RAM	8 GB	8 GB	4 GB
Net	Wifi+Gigabit Ethernet	optional: Wifi and Ethernet	Gigabit Ethernet
Display	2x miniHDMI, DSI	optional: HDMI, DSI	HDMI
IO	4xUSB, GPIO, CSI, 3.5 audio	optional PCIe, USB, CSI, GPIO	4xUSB, GPIO, CSI, PCIe
Video Decode	H.265(4Kp60), H.264(1080p60)	H.265(4Kp60), H.264(1080p60)	H.264/H.265 (4Kp60,2x4Kp30)
Video Encode	H264(1080p30)	H264(1080p30)	H.264/H.265 (4Kp30)
Extra Memory	up to 128GB on SD	32 GB eMMC	up to 128GB on SD

Table 2.1: Comparing RPI 4, RPI CM 4 and Jetson Nano

From the table 2.1 it is clear that both Raspberry and Nvidia Jetson have almost the same processor – ARM Cortex-A. The only difference is that the RPi 4 processor is slightly newer and has a smaller power consumption. [8] [6] However, there is a considerable difference in GPU cores. Jetson Nano is ideal for processing neural networks, but, unfortunately, such system requires massive passive cooling and optionally the addition of active cooling (i.e. a fan) to avoid throttling, which would greatly complicate the 3D modelling and the hand guiding controller would be too big to fit into the limited space of the hand-guiding controller handle. Moreover, in Jetson Nano, there is no DSI connector, which allows the screen to be compactly connected during debugging because HDMI wire and connector will take up quite a lot of space.

Therefore, the choice fell to the RPi computer. I decided to try to find a CM4 module so that it would be possible to make a neat PCB board and not use the mechanics of the usual RPi peripherals. In such a way, only those peripherals that are needed in the project can be used. The problem of a weak GPU core can be solved using 5G video streaming to the server with a GPU computational power.

2.3 PCB development

The EasyEDA program was chosen to develop the printed circuit board (PCB). This software developing tool contains many libraries, and enables a very intuitive developing process.

The DC/DC convertor from 24 V to 5 V

The DC standard in the industry sphere is 24 V [18], so the first element added to a PCB is the DC/DC converter from 24 V to 5 V, because interfaces like USB, HDMI, and RPi require 5 V to work.

The basis of the solution was generated by a web tool from Texas Instruments [21], which provided an available Step-Down DC/DC Switching Regulator solution.

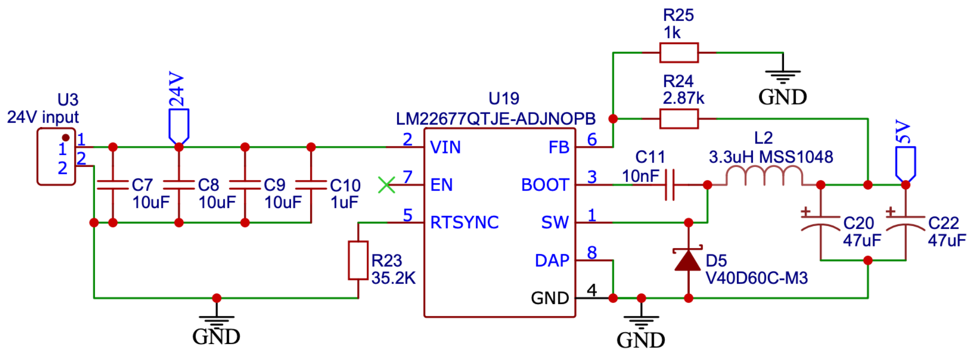


Figure 2.4: DC/DC converter from 24 V to 5 V.

The value of R_{25} resistor is predefined in the datasheet [20] as a 1 kΩ. So, R_{24} was calculated as follows:

$$R_{24} = (\frac{V_{out}}{1.285} - 1) \cdot R_{25} \tag{2.1}$$

$$R_{24} = (\frac{5}{1.285} - 1) \cdot 1000 = 2.87\,k\Omega$$

The induction L_2 was calculated with respect for the maximum frequency of the LM22677 chip to save space, as smaller capacitors and smaller inductors are used at high frequencies.

$$F_{max} = 1 \cdot 10^6 = 1\,MHz \tag{2.2}$$

$$L = \frac{(V_{in} - V_{out}) \cdot V_{out}}{0.3 \cdot I_{out} \cdot F_{max} \cdot V_{in}} = \frac{(24-5) \cdot 5}{0.3 \cdot 4.25 \cdot 10^6 \cdot 24} = 3.3\,\mu H \tag{2.3}$$

To set the frequency at 1 MHz the R_{23} resistor should be connected to RTSYNC pin. The R_{23} resistor's value could be calculated according to the following graph 2.5 taken from the datasheet [20].

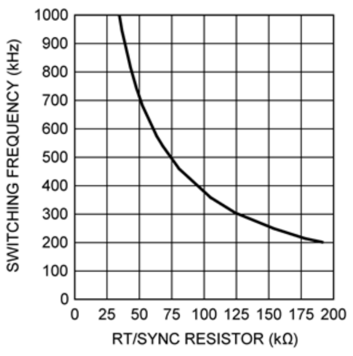


Figure 2.5: Resistor's value dependency on switching frequency for LM22677.

The capacitors were selected according to the datasheet [20], . For input capacitors (C_7, C_8, C_9, C_{10}), according to the Texas Instruments application report [10], peak to peak ripple amplitude of the input voltage to he LM22677 should be below 75 mV to keep the RMS currents in the bulk capacitors within acceptable limits. So this ripple amplitude was chosen as in the TI web tool equal to 40 mV. Formulas 2.4 were taken from the LM22677 datasheet [20].

$$V_{RI} = \frac{I_{OUT}}{4 \cdot F_{max} \cdot C_{IN}} \tag{2.4}$$

$$C_{IN} = \frac{I_{OUT}}{4 \cdot F_{MAX} \cdot V_{RI}} = \frac{5}{4 \cdot 10^6 \cdot 40 \cdot 10^{-3}} = 31\,\mu F$$

For the output capacitors the recommended capacitance is ≈ 100 µF according to the TI datasheet [20].

5G modem connection

The 5G modem must be powered with 4.2 V. Therefore, a DC/DC step-down converter was used, which will produce 4.2 V from 5 V.

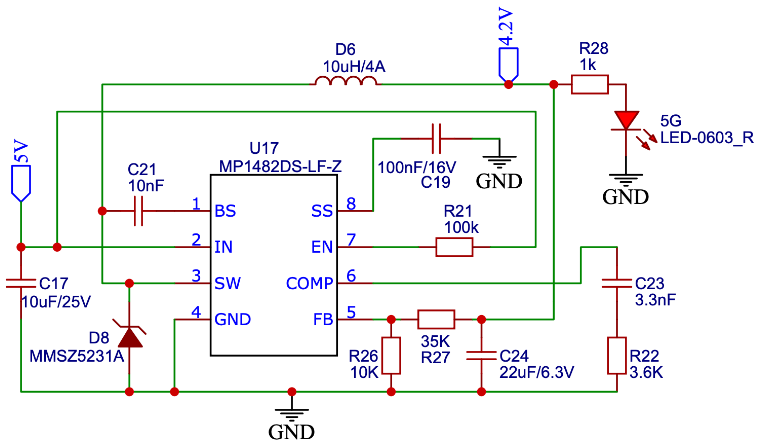


Figure 2.6: DC/DC convertor 5 V to 4.2 V

The MP1482-2A chip was chosen in a convenient SOIC8 package. The connection to the circuit itself is based on datasheet [11] recommendations. Additional calculations were carried out only for the adjustment of the output voltage. Since the circuit does not have an internal adjustment in the feedback voltage connection, it was necessary to calculate the voltage divider. R_{26}, according to the datasheet [11], should have a constant value of 10K and R_{27} should be calculated using the voltage divider formula where 0.923 V is the feedback threshold.

$$V_{out} = 0.923 \cdot \frac{R_{26} + R_{27}}{R_{26}} \tag{2.5}$$

$$R_{27} = 10.83 \cdot (-0.923 + V_{out}) \approx 35\,k\Omega$$

For the 5G communication, the SIM8202X-M2 modem was used due to its relatively low price and easy availability in Europe. Furthermore, an essential point in the choice was the availability of drivers for the Raspbian OS operating system.

To use the SIM8202X-M2 5G modem [19], it is necessary to connect only a few pins. To simplify the board in the current version, a USB type 2.0 bus was used despite the limitations in data transfer speed, although it would be better to use USB3.0. However, in such a case it would be necessary to connect a PCIe-to-USB 3.0 converter, which is now quite hard to find in convenient soldering cases.

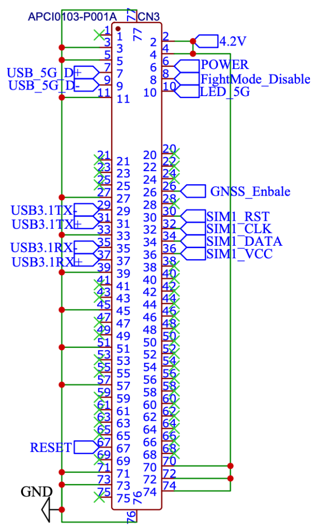


Figure 2.7: Scheme of the M2 5G connector

USB connection

To be able to use the debug mouse and keyboard connection, a USB connector was integrated into the PCB so it would be possible to control the operating system directly (in case of the SSH connection it is not available). Since the 5G modem also uses the USB interface, the task was to connect 3 USB devices to the RPi processor with one USB input. The solution was to integrate a USB hub to distribute three different USB interfaces.

Due to the big microelectronics crisis, the original chips are unfortunately unavailable for SMT soldering from the PCB manufacturer, i.e., JLCPCB. Therefore, the SL2.1A chip was used, which is an alternative of the USB-Hub chips from Texas Instruments or Microchip.

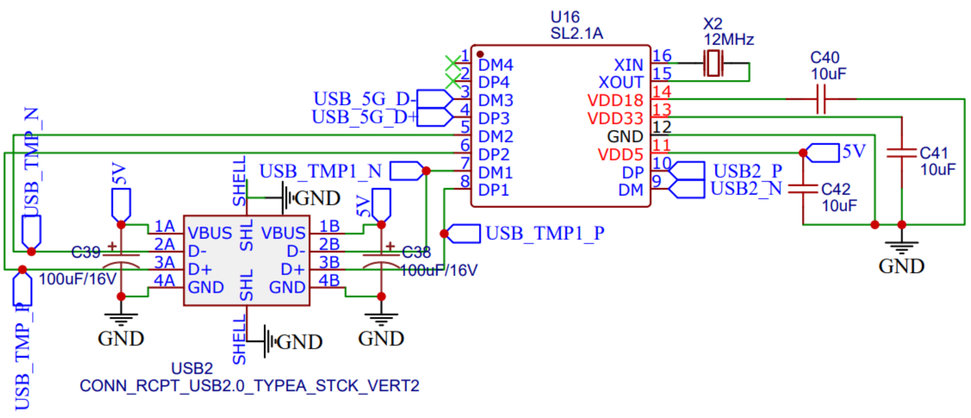


Figure 2.8: Scheme of the USB hub, and USB connectors

In accordance to the dashasheet [7], a 12 MHz quartz resonator and 10 μF capacitors are installed for the SL2.1 (Figure 2.8) to the input and outputs of the integrated SL2.1A lines of voltage stabilizers. Capacitors C_{36} and C_{39} are needed to smooth out the peak load when connecting the device to the connector.

RPI connection

RPi CM 4 is quite handy device to work with, as its pins 2.9 are very well described in its manual [14] and do not require much additional connections, because rPI CM 4 gives access directly to interfaces.

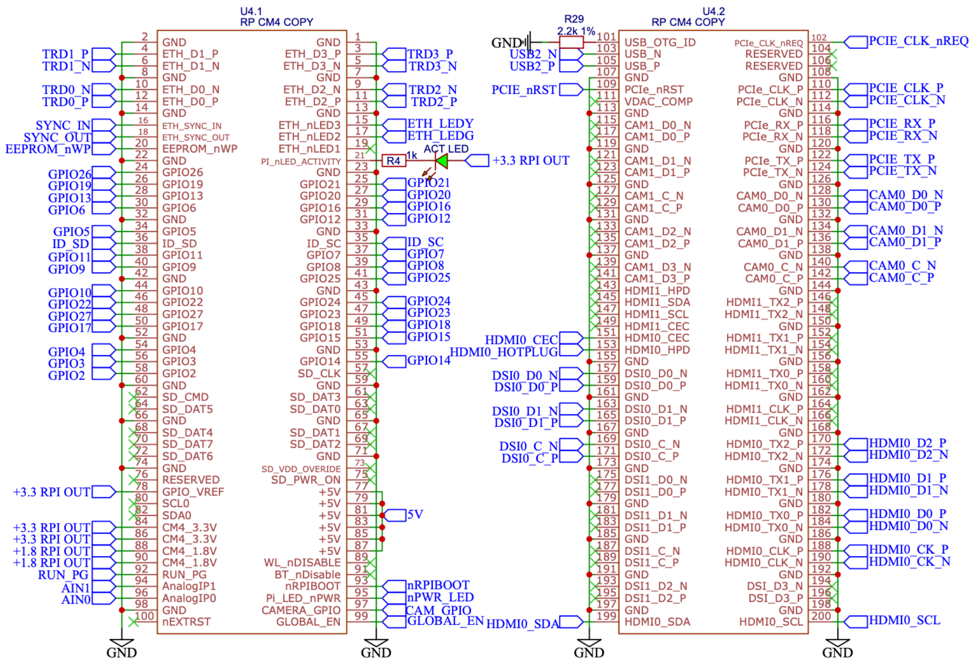


Figure 2.9: Scheme of the RPI connectors

The raspberry itself must be connected via two DF40HC(3.0)-100DS connectors [9], which unfortunately are almost impossible to buy, so the connectors themselves were soldered by hand from the RPI IO dev kit[13], which was used for the initial proof-of-concept development in this project. The rPI pinout allows quick connection to all the necessary peripherals, such as:

1. **HDMI** – in this version, it is needed only for debugging before assembly (not used in the final version), where four twisted pairs and an I2C line are used.
2. **DSI** – 3 twisted pairs to connect a Waveshare 4.3" LCD screen via DSI (HDMI was not used to save space in the hand guiding controller)
3. **CSI** – 3 twisted pairs to connect RPI V2.0 camera
4. **RJ45** – 4 twisted pairs using the PROFINET communication protocol with the PLC, which pass through the DFN2510 protection chips.
5. **12 I/O** connectors for buttons. In the current version, buttons were not used since the display has touch-pad capability and there is no need to use mechanical buttons .
6. **1 USB** connection to the USB hub, which will distribute 3 USB connectors for mouse and keyboard connection (for debug process, not currently used) and also for the 5G connection.

In addition, six 5 V pins are assigned to power up the micro-computer, which allows bringing a relatively high current (up to approximately 3A). This version of RPi has a built-in eMMS memory, so there is no need to connect an SD card.

2.4 PCB design

The main and most time-consuming part of the hardware design phase was the development of the PCB board that would contain all the necessary connectors for all connection interfaces, two DC/DC converters for different voltages required (5 V for most of interfaces and also for the rPI and HDMI and 4.2 V voltage for 5G).

This version is the so-called developer version, which contains several extra connectors and parts. In the next board, the size will be reduced to half the size compared to the current one using double-sided soldering and removing extra debug connectors such as USB and HDMI. The board is a 4-layer board with a thickness of 1.6 mm.

Due to the rather poor implementation of the automatic routing function in the easyEDA software, the entire board was routed manually with the maximum observance of the electrical regulations. In this PCB realisation it was planned to place:

1. Raspberry CM4 connector

2. Modem 5G SIMCOM SIM8202
3. CSI camera interface
4. DSI display interface
5. DC 24 V Input
6. 11x logic inputs for buttons
7. HDMI input
8. RJ45 for PROFINET connection

Most interfaces have differential pairs in them. Using the four-layer PCB (see figures 2.10 and 2.11) allowed keeping the PCB small.

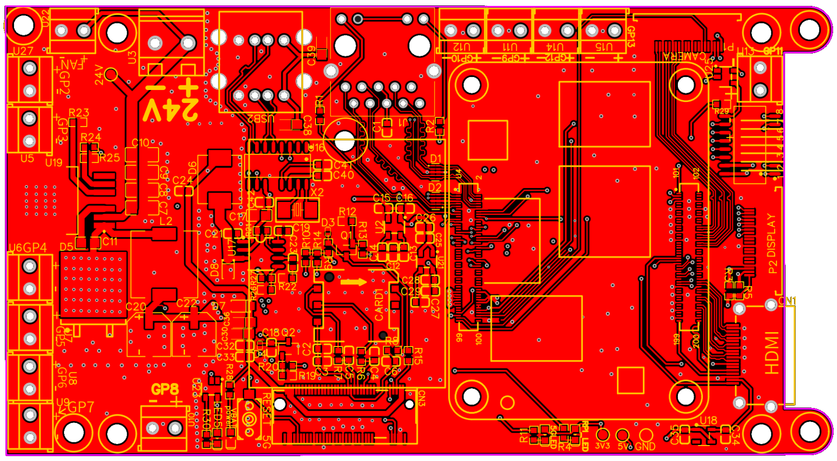


Figure 2.10: 2D view of the designed PCB

The following rules were followed during the PCB design:

1. Due to the relatively big current consumption component U7 and U19 (linear stabilizators) could be very hot. A system to remove heat from the those component through the vias was integrated to the PCB. So in fact, there are a lot of via holes, on the solder-paste place to let heat go through the board more quickly and decrease the temperature of the component.
2. Minimized current loops. The distance in DC/DC converters was minimized near to: input capacitors, MOSFETs, inductors and output capacitors. Also it was important to make thick traces for the 4A maximum current. As an reference the LM22677 datasheet [20] and the MP1482 [11] were taken.
3. The differential pairs should have a non-breaking reference layer.
4. All capacitors that belong to a chip in the board are also located near this chip.

5. All relating PCB components are grouped. For example a 5G sim card should be near the 5G connector.

6. The high-current blocks were separated with via holes to avoid noises in the signal tracks.

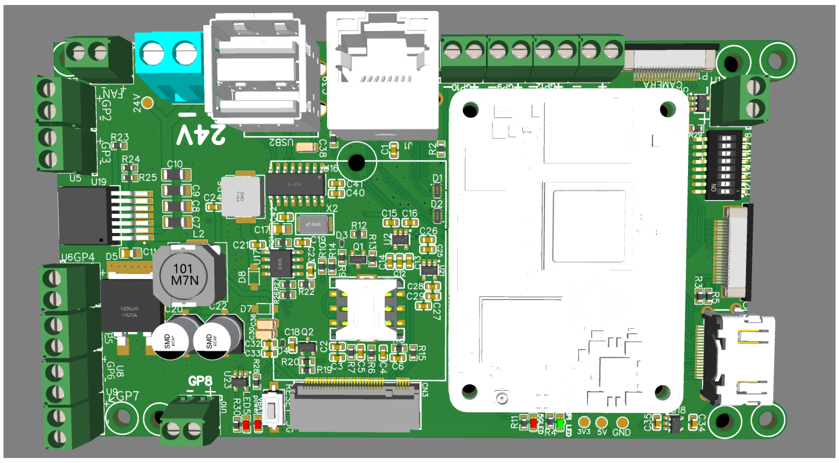


Figure 2.11: 3D view of the designed PCB

A temperature test was done after about 5 hours of running the board in full video streaming mode over 5G in an open case and 24 degrees in the room. It is also important to know that the temperature of the 5G modem on the figure 2.12 (a rectangle in the middle) is not accurate since the modem case is a silver crumb. To measure the modem temperature, the thermal camera was directed at the tape pasted on the module in a separate measurement that gave the result of about 60 degrees Celsius.

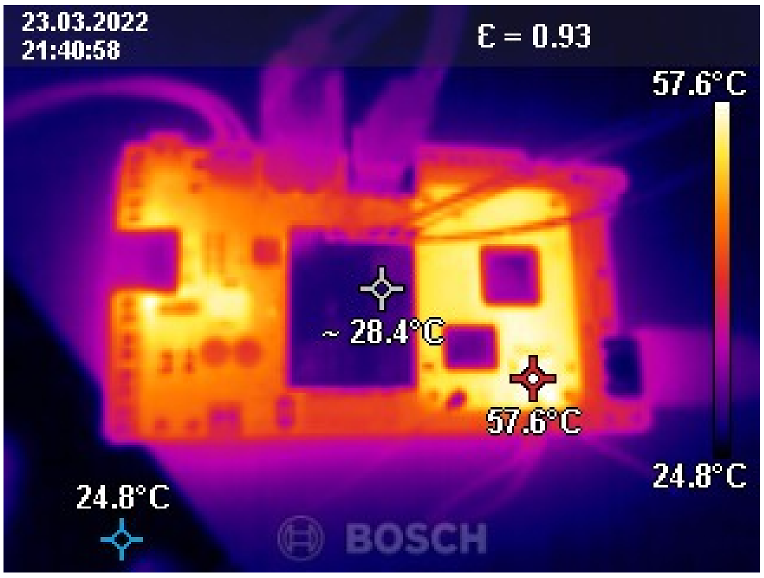
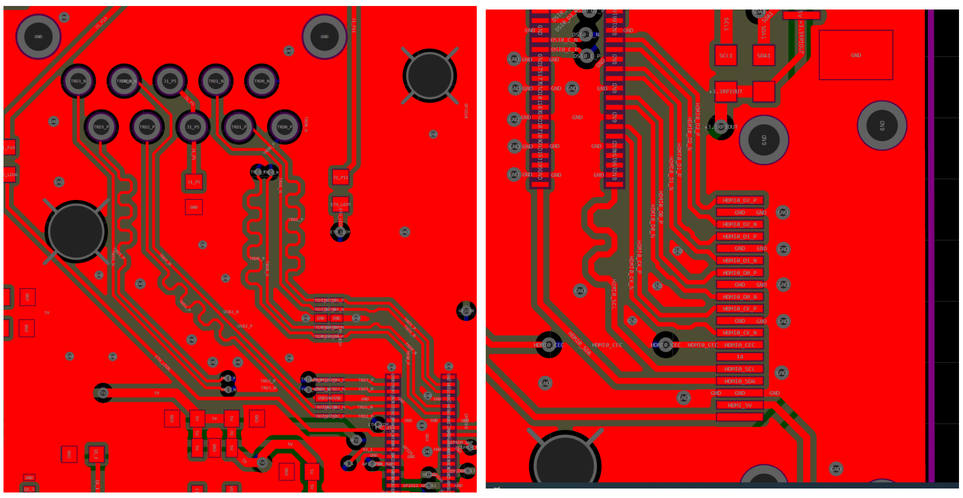


Figure 2.12: Thermal camera test

60 degrees Celsius is the normal operating temperature for all parts of this circuit, which means the design of the PCB was correct. However, in the following tests, there was a need to add active cooling to the board because when the board was in a closed case, after a few hours the temperature reached 80 degrees, which is already at the limit of RPi's stable operation.

The tricky part in the implementation of the board was pair routing, since in the EasyEDA editor, there is no compensation for the lengths of two contacts in one twisted pair, so it was done manually. Twisted pairs were made according to the following rules:

1. Matched lengths. Because of the usage of source-synchronous clocking, it is crucial to keep the net length in one parallel data pair the same, not to desynchronize the communication such as for the Ethernet and HDMI connections in the project scheme in figures 2.13a and 2.13b) [22] [23]. In each differential pair the length of routes should be matched up to 0.15 mm. And in the different differential pairs of the HDMI interface, matching could be up to 25 mm.



(a) : Pair routing in Ethernet connection **(b) :** Pair routing in HDMI connection

Figure 2.13: Example of pair routing

2. Limited amount of via transitions. When we use pair routing, it is recommended to use a limited amount of via transitions between layers. In the case of the hand-guiding controller, vias in one pair have been used maximum twice.

3. Clearances. There should be no components near or between the pair routing. In this way, unnecessary interference can be avoided and communication is more stable.

4. Different widths for different buses. This is the standard recommended by Texas Instruments.

Chapter 3

Profinet communication implementation

Profinet is one of the most popular industrial communication protocols [12]. This protocol exchanges data between the PLC controller and the devices, which in most cases are equipped with a 2 port switch to allow chain topology. Devices in the Profinet network can be completely different, for example, my hand-guiding controller, a pressure sensor, or another controller that will participate in the communication.

Profinet mainly uses the classic type of connector (though in a much stronger case than the classic one), RJ45. In rare cases, M12 4Pin is used if the working conditions are prone to mechanical damage. Since RJ45 is used, Profinet is fully compatible with a conventional Ethernet network. The only difference is that it is impossible to use hard-real-time data in an office Ethernet network. In addition, classic office Ethernet wires are not recommended due to their low mechanical durability. For the hand-guiding controller, the PROFINET RTC1 protocol is used with the cyclic update time being 4 ms.

3.1 Profinet core

The PROFINET RTC1 protocol utilizes the standard Ethernet frame as shown in figure 3.1. The type field of the Ethernet frame has the value of 0x8892, which says that PROFINET-specific information is send in the data field. The data field starts with the Frame ID, identifying the type of communication at the level of the PROFINET [3] protocol, and further contains the data specific for the given type of message. The status field contains the cycle counter and quality-related information. The frame ends with the standard Ethernet CRC field denoted as FCS.

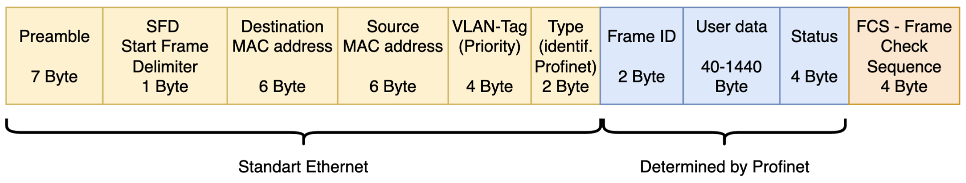

Preamble 7 Byte
SFD Start Frame Delimiter 1 Byte
Destination MAC address 6 Byte
Source MAC address 6 Byte
VLAN-Tag (Priority) 4 Byte
Type (identif. Profinet) 2 Byte
Frame ID 2 Byte
User data 40-1440 Byte
Status 4 Byte
FCS - Frame Check Sequence 4 Byte
Standart Ethernet
Determined by Profinet

Figure 3.1: PROFINET frame

At least 50 % of the traffic in a PROFINET communication is reserved for such data transmission, for which the exact time is not important, and which is transferred mainly using the TCP/IP protocol. The rest of the traffic is used to transmit information in a real-time data exchange with a delay of the order of ones or tens of ms.

Since PROFINET is a highly complex and voluminous communication protocol, an open source stack from rt-labs.com [15] was used, which is written in the C programming language and is relatively easy to use.

In the PROFINET stack from the RT-labs, there are almost all the protocol functions ready, such as

- Multiple Ethernet ports
- TCP/IP
- LLDP - Link Layer Discovery Protocol
- SNMP - Simple Network Management Protocol
- RT (real-time class 1)
- Address resolution
- Process IO data exchange
- Alarm handling
- Configurable number of modules and sub-modules
- Could work on Bare-metal or Linux OS
- Porting layer provided
- Supports I&M0 - I&M4. The I&M data is supported for the device, but not for individual modules.

3.2 Implementation

At the beginning of the program, there are two applications in `./bashrc` file (autorun) that is needed for communication through PROFINET. The first is the RT-labs stack [15] itself. This stack was redone into two threads. The first thread is responsible for ensuring that everything that comes from the PLC is immediately written to the `profinet_out` pipe file. The second thread is responsible for reading `profinet_in` pipe file and sends these data to the Profinet, that is, to the robot PLC.

The second application was written in Python; this application is, in fact, the user application built above the communication stack. In two threads, it reads and writes data to `profinet_in` and reads `profinet_out` pipes. Based on the received data, the program changes the target coordinates of the robot and, when they change, sends the data to the `profinet_in` pipe file.

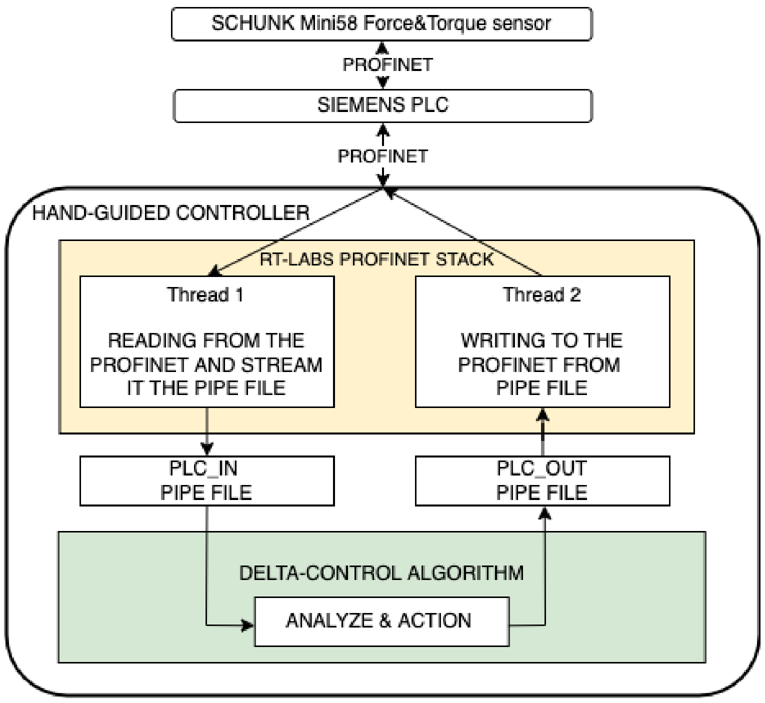


Figure 3.2: Logic scheme of the PROFINET communication

The stack also requires a GSDML file (General Station Description Markup Language, XML-based) that describes the operation of the PROFINET IO-Device; in other words, it plays the role of defining the sizes of communication data and define other device parameters.

The correctness of the GSDML file can be checked using the PROFINET GSD Checker application [4]. The file name is built as follows:

```
GSDML-V(#.##)-NAME-DATE.xml
```

For example:

```
GSDML-V7.77-My-Device-20220520.xml
```

GSDML of the hand guiding controller:

```
1. <DataItem DataType="Float64/>
2. <DataItem DataType="Float64/>
3. <DataItem DataType="Float64/>
4. <DataItem DataType="INT"/>
5. <DataItem DataType="INT"/>
```

where three `Float64` variables are for the Delta destination coordinates and two `INT` variables for pick & place task. First `INT` means that the object should be taken from this position on the first conveyor platform, and second `INT` means that the object should be placed in this position on the second conveyor platform.

Input of the hand guiding controller:

```
1. <DataItem DataType="Float64"/>
2. <DataItem DataType="Float64"/>
3. <DataItem DataType="Float64"/>
4. <DataItem DataType="Boolean"/>
5. <DataItem DataType="Boolean"/>
```

where:
1-3) Three Float64 variables are for the actual Delta destination coordinates.
4) Boolean means that the first conveyer platform is ready to be scanned
5) Boolean means that the second conveyer platform is ready to be scanned

A very important part is also the fact that the stack must run on a separate core. Because this communication protocol is industrial, communication disturbances or delays in the cyclic communication are detected as communication errors. Therefore, for a smooth and stable stack operation, it was necessary to completely dedicate one core and run only the communication protocol script on it. For the Profinet process, I chose the third core, which is dedicated by the command:

```
    sudo nano /proc/cpuinfo
```

And adding to the end of the file:

```
    isolcpus=2
```

After a reboot, locked kernels can be checked with the command:

```
    cat /sys/devices/system/cpu/isolated
```

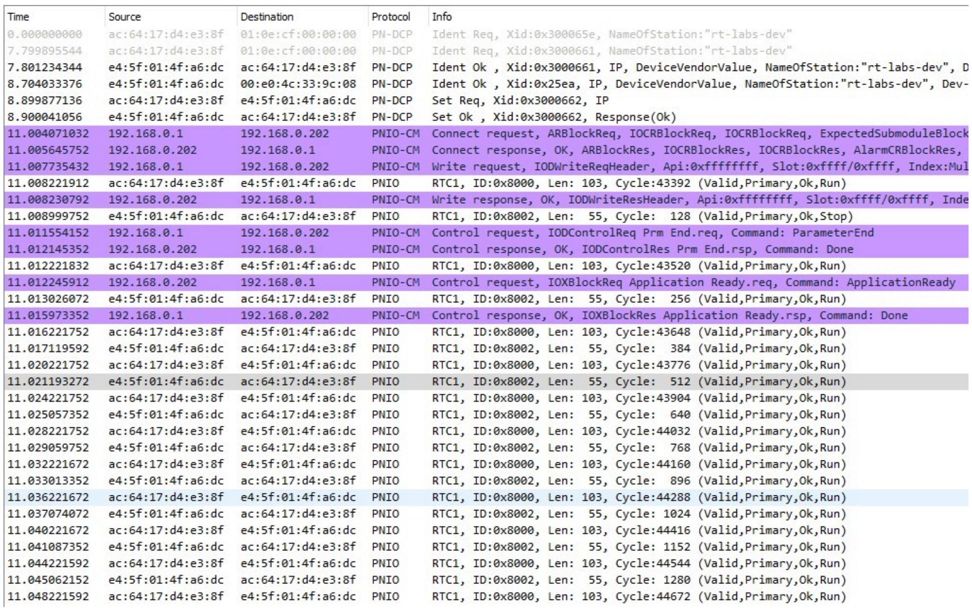
After blocking the kernel, the Profinet stack is started using this command:

```
   taskset -c 2 ./pn_dev
```

Figure 3.3 shows the identification of the PROFINET IO device using the DCP protocol and also the establishment of the application relationship (AR). After the device responds to DCP.Ident.Req with DCP.Ident.Ok, the IO controller sets its IP address. The Connect.req message represents the beginning of the startup phase – the AR creation. The correct configuration of the IO Device is signalled by the absence of the ModuleDiffBlock in the Connect.res message. After the AR is established, which is signalled by Control.req.ApplicationReady being sent by the IO Device, the cyclic IO data communication starts. Actually, the IO Controller starts sending the output data frames after the Write.req message, but the IO Device starts sending its input data frames after Application Ready is signalled.

The list in Figure 3.3 was displayed using a Wireshark filter to show only messages that are relevant to the communication of the IO controller ⇔ IO device, by defining the MAC address of the IO device and the communication protocol such as PN-DCP and others.

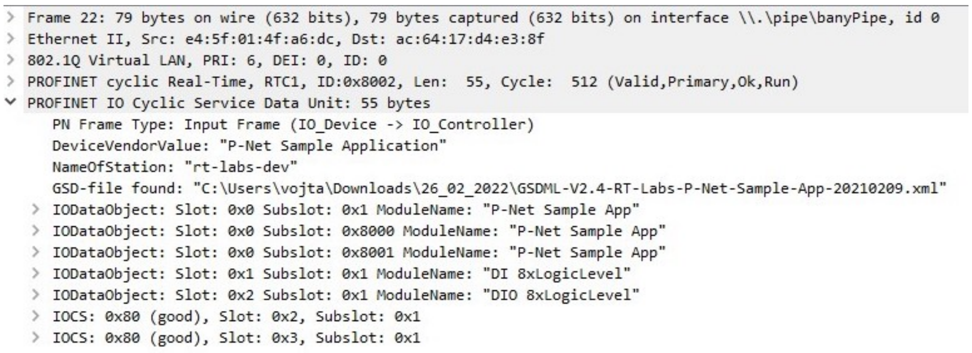
```
((eth.addr == E4:5F:01:4F:A6:DC) &&
 (pn_io || pn_dcp || dcerpc))  ||
  pn_dcp.suboption_device_nameofstation == rt-labs-dev
```



```
Time          Source             Destination        Protocol  Info
0.000000000   ac:64:17:d4:e3:8f  01:0e:cf:00:00:00  PN-DCP    Ident Req, Xid:0x300065e, NameOfStation:"rt-labs-dev"
7.799895544   ac:64:17:d4:e3:8f  01:0e:cf:00:00:00  PN-DCP    Ident Req, Xid:0x3000661, NameOfStation:"rt-labs-dev"
7.801234344   e4:5f:01:4f:a6:dc  ac:64:17:d4:e3:8f  PN-DCP    Ident Ok , Xid:0x3000661, IP, DeviceVendorValue, NameOfStation:"rt-labs-dev", D
8.704033376   e4:5f:01:4f:a6:dc  00:e0:4c:33:9c:08  PN-DCP    Ident Ok , Xid:0x25ea, IP, DeviceVendorValue, NameOfStation:"rt-labs-dev", Dev-
8.899877136   ac:64:17:d4:e3:8f  e4:5f:01:4f:a6:dc  PN-DCP    Set Req, Xid:0x3000662, IP
8.900041056   e4:5f:01:4f:a6:dc  ac:64:17:d4:e3:8f  PN-DCP    Set Ok , Xid:0x3000662, Response(Ok)
11.004071032  192.168.0.1        192.168.0.202      PNIO-CM   Connect request, ARBlockReq, IOCRBlockReq, IOCRBlockReq, ExpectedSubmoduleBlock
11.005645752  192.168.0.202      192.168.0.1        PNIO-CM   Connect response, OK, ARBlockRes, IOCRBlockRes, IOCRBlockRes, AlarmCRBlockRes,
11.007735432  192.168.0.1        192.168.0.202      PNIO-CM   Write request, IODWriteReqHeader, Api:0xffffffff, Slot:0xffff/0xffff, Index:Mul
11.008221912  ac:64:17:d4:e3:8f  e4:5f:01:4f:a6:dc  PNIO      RTC1, ID:0x8000, Len: 103, Cycle:43392 (Valid,Primary,Ok,Run)
11.008230792  192.168.0.202      192.168.0.1        PNIO-CM   Write response, OK, IODWriteResHeader, Api:0xffffffff, Slot:0xffff/0xffff, Inde
11.008999752  e4:5f:01:4f:a6:dc  ac:64:17:d4:e3:8f  PNIO      RTC1, ID:0x8002, Len:  55, Cycle:  128 (Valid,Primary,Ok,Stop)
11.011554152  192.168.0.1        192.168.0.202      PNIO-CM   Control request, IODControlReq Prm End.req, Command: ParameterEnd
11.012145352  192.168.0.202      192.168.0.1        PNIO-CM   Control response, OK, IODControlRes Prm End.rsp, Command: Done
11.012221832  ac:64:17:d4:e3:8f  e4:5f:01:4f:a6:dc  PNIO      RTC1, ID:0x8000, Len: 103, Cycle:43520 (Valid,Primary,Ok,Run)
11.012245912  192.168.0.202      192.168.0.1        PNIO-CM   Control request, IOXBlockReq Application Ready.req, Command: ApplicationReady
11.013026072  e4:5f:01:4f:a6:dc  ac:64:17:d4:e3:8f  PNIO      RTC1, ID:0x8002, Len:  55, Cycle:  256 (Valid,Primary,Ok,Run)
11.015973352  192.168.0.1        192.168.0.202      PNIO-CM   Control response, OK, IOXBlockRes Application Ready.rsp, Command: Done
11.016221752  ac:64:17:d4:e3:8f  e4:5f:01:4f:a6:dc  PNIO      RTC1, ID:0x8000, Len: 103, Cycle:43648 (Valid,Primary,Ok,Run)
11.017119592  e4:5f:01:4f:a6:dc  ac:64:17:d4:e3:8f  PNIO      RTC1, ID:0x8002, Len:  55, Cycle:  384 (Valid,Primary,Ok,Run)
11.020221752  ac:64:17:d4:e3:8f  e4:5f:01:4f:a6:dc  PNIO      RTC1, ID:0x8000, Len: 103, Cycle:43776 (Valid,Primary,Ok,Run)
11.021193272  e4:5f:01:4f:a6:dc  ac:64:17:d4:e3:8f  PNIO      RTC1, ID:0x8002, Len:  55, Cycle:  512 (Valid,Primary,Ok,Run)
11.024221752  ac:64:17:d4:e3:8f  e4:5f:01:4f:a6:dc  PNIO      RTC1, ID:0x8000, Len: 103, Cycle:43904 (Valid,Primary,Ok,Run)
11.025057352  e4:5f:01:4f:a6:dc  ac:64:17:d4:e3:8f  PNIO      RTC1, ID:0x8002, Len:  55, Cycle:  640 (Valid,Primary,Ok,Run)
11.028221752  ac:64:17:d4:e3:8f  e4:5f:01:4f:a6:dc  PNIO      RTC1, ID:0x8000, Len: 103, Cycle:44032 (Valid,Primary,Ok,Run)
11.029059752  e4:5f:01:4f:a6:dc  ac:64:17:d4:e3:8f  PNIO      RTC1, ID:0x8002, Len:  55, Cycle:  768 (Valid,Primary,Ok,Run)
11.032221672  ac:64:17:d4:e3:8f  e4:5f:01:4f:a6:dc  PNIO      RTC1, ID:0x8000, Len: 103, Cycle:44160 (Valid,Primary,Ok,Run)
11.033013352  e4:5f:01:4f:a6:dc  ac:64:17:d4:e3:8f  PNIO      RTC1, ID:0x8002, Len:  55, Cycle:  896 (Valid,Primary,Ok,Run)
11.036221672  ac:64:17:d4:e3:8f  e4:5f:01:4f:a6:dc  PNIO      RTC1, ID:0x8000, Len: 103, Cycle:44288 (Valid,Primary,Ok,Run)
11.037074072  e4:5f:01:4f:a6:dc  ac:64:17:d4:e3:8f  PNIO      RTC1, ID:0x8002, Len:  55, Cycle: 1024 (Valid,Primary,Ok,Run)
11.040221672  ac:64:17:d4:e3:8f  e4:5f:01:4f:a6:dc  PNIO      RTC1, ID:0x8000, Len: 103, Cycle:44416 (Valid,Primary,Ok,Run)
11.041087352  e4:5f:01:4f:a6:dc  ac:64:17:d4:e3:8f  PNIO      RTC1, ID:0x8002, Len:  55, Cycle: 1152 (Valid,Primary,Ok,Run)
11.044221592  ac:64:17:d4:e3:8f  e4:5f:01:4f:a6:dc  PNIO      RTC1, ID:0x8000, Len: 103, Cycle:44544 (Valid,Primary,Ok,Run)
11.045062152  e4:5f:01:4f:a6:dc  ac:64:17:d4:e3:8f  PNIO      RTC1, ID:0x8002, Len:  55, Cycle: 1280 (Valid,Primary,Ok,Run)
11.048221592  ac:64:17:d4:e3:8f  e4:5f:01:4f:a6:dc  PNIO      RTC1, ID:0x8000, Len: 103, Cycle:44672 (Valid,Primary,Ok,Run)
```

Figure 3.3: Establishing PROFINET communication

Figure 3.4 shows the details of an input frame, where the statuses (IOPS and IOCS) are already Good.



```
> Frame 22: 79 bytes on wire (632 bits), 79 bytes captured (632 bits) on interface \\.\pipe\banyPipe, id 0
> Ethernet II, Src: e4:5f:01:4f:a6:dc, Dst: ac:64:17:d4:e3:8f
> 802.1Q Virtual LAN, PRI: 6, DEI: 0, ID: 0
> PROFINET cyclic Real-Time, RTC1, ID:0x8002, Len:  55, Cycle:  512 (Valid,Primary,Ok,Run)
v PROFINET IO Cyclic Service Data Unit: 55 bytes
    PN Frame Type: Input Frame (IO_Device -> IO_Controller)
    DeviceVendorValue: "P-Net Sample Application"
    NameOfStation: "rt-labs-dev"
    GSD-file found: "C:\Users\vojta\Downloads\26_02_2022\GSDML-V2.4-RT-Labs-P-Net-Sample-App-20210209.xml"
  > IODataObject: Slot: 0x0 Subslot: 0x1 ModuleName: "P-Net Sample App"
  > IODataObject: Slot: 0x0 Subslot: 0x8000 ModuleName: "P-Net Sample App"
  > IODataObject: Slot: 0x0 Subslot: 0x8001 ModuleName: "P-Net Sample App"
  > IODataObject: Slot: 0x1 Subslot: 0x1 ModuleName: "DI 8xLogicLevel"
  > IODataObject: Slot: 0x2 Subslot: 0x1 ModuleName: "DIO 8xLogicLevel"
  > IOCS: 0x80 (good), Slot: 0x2, Subslot: 0x1
  > IOCS: 0x80 (good), Slot: 0x3, Subslot: 0x1
```

Figure 3.4: IO-cyclic message in Wireshark

Chapter 4

5G implementation

4.1 5G network communication in CIIRC

The RAN network provides the required coverage of 5G indoor spaces for CTU and its UE applications.

The radio access network can be characterized as a network composed of base stations NB (New Radio NodeB), which are characterized by support for OFDM modulation and advanced antenna techniques (MIMO), which form the basis of performance of the entire system. Each of NB - New Radio NodeB has its own IP address, which makes it part of the IP network.

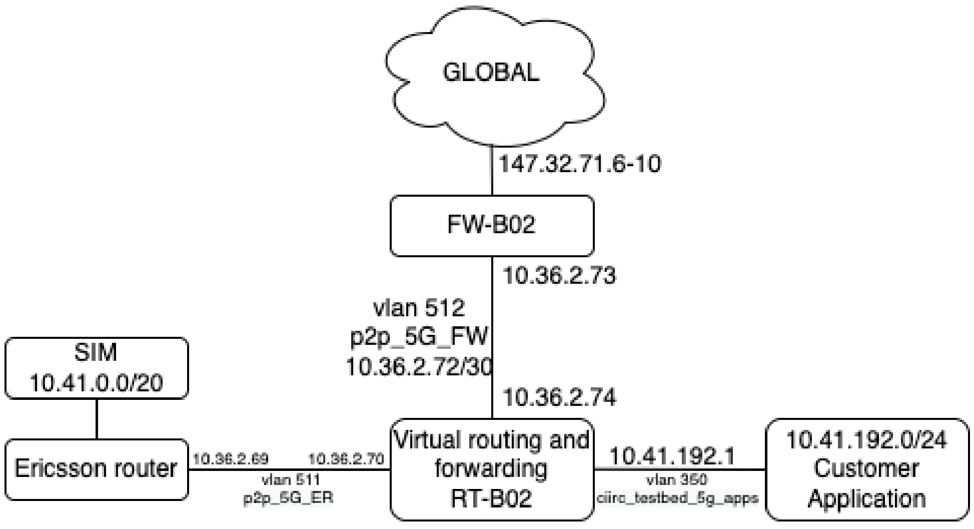


Figure 4.1: Network architecture in CIIRC

In terms of operating frequencies, the frequency spectrum bands in the band 3480 - 3540 MHz are used to cover the CN; their allocation is subject to the Frequency Spectrum Auction organized by the CTU institution, which took place in 2021. The allocated bandwidth is 60 MHz, and it is a band in TDD (Time Division Duplex), that is, with a time shift. This means that bandwidth is allocated for the downlink and uplink.

ENB (Electromagnetic Navigation Bronchoscopy) technology will be used to create the required coverage, which consists of the following components:

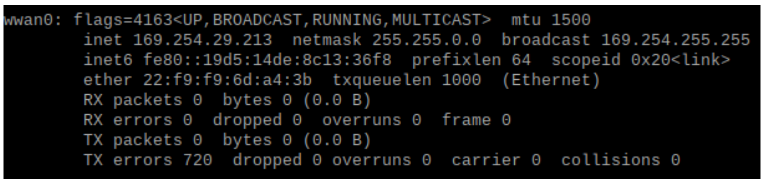
- DU (Digital Baseband Unit, sometimes referred to simply as BB - Baseband) provides aggregation of mobile traffic and connection to the 5G Core network.
- IRU (Indoor Radio Unit) ensures the convergence of the CPRI optical signal to Ethernet. The IRU is connected to the DU (baseband) via optical cabels.
- DOT (Pico Radio Unit) - radio transmitter with active antenna, which is connected to the IRU via Ethernet cable (Cat5e-7, CAT6, ideally shielded). It has a very low power consumption (less than one watt) and is optimal for indoor coverage, especially for office space. The active antenna is powered by Ethernet. The DOT units used are in 4x4 MIMO mode.

4.2 5G Implementation

Running 5G communication with the SIM8200EA-M2 5G modem [19] is required to carry out the installation process. First, it is extremely important to install the correct version of the Raspberry Pi OS from date 2020-08-20, since all driver settings work exclusively with the kernel version 5.4. Unfortunately, with other kernel versions(even older versions), the modem driver did not work. After installing the operating system, the first step is to download and install the driver from the Waveshare website with the following commands:

```
wget www.waveshare.com/w/upload/f/fb/SIM8200-M2_5G_HAT_code.7z
7z x SIM8200-M2_5G_HAT_code.7z
sudo chmod 777 -R SIM8200-M2_5G_HAT_code
cd SIM8200-M2_5G_HAT_code
sudo ./install.sh
```

After that, the wwan0 interface should have appeared in the network settings:



```
wwan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 169.254.29.213  netmask 255.255.0.0  broadcast 169.254.255.255
        inet6 fe80::19d5:14de:8c13:36f8  prefixlen 64  scopeid 0x20<link>
        ether 22:f9:f9:6d:a4:3b  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 720  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 4.2: wwan0 parameter before 5G activation

After that, it is possible to test the connection using AT commands, open the COM port of the monitor at the address: /dev/ttyUSB2.
With the standard AT command, communication can be checked, and the response should be OK.

After the connection to the modem has been confirmed, a script must run that turns the modem into a 5G modem mode and registers itself on the network:

```
sudo /Goonline/simcom-cm
```

After that, the modem needs to be configured using the AT commands as described in the modem's manual.

```
AT + CREG = 0
```

which means: disable the unsolicited result code for network registration.

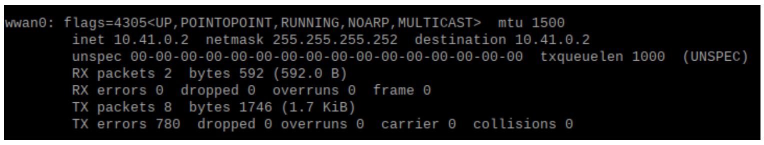
```
AT + COPS = 0, 0, 'CAMPUS', 11
```

which means:
AT+COPS=mode,format,operator,technology,
where:

- mode = 0 means automatic
- format = 0 means long format of operator name
- operator = "CAMPUS" – name of the operator of the 5G CIIRC
- technology = 11 means NR_5GCN (NR connected to the 5G core network)

```
AT + CNMP = 71
```

which means that the preferred communication mode is 'NR5G only'. After that, in the network parameters on the wwan0 IP interface, the address should change to 10.41.0.X (Figure 4.3), which will mean that 5G is successfully connected.



```
wwan0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.41.0.2  netmask 255.255.255.252  destination 10.41.0.2
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 1000  (UNSPEC)
        RX packets 2  bytes 592 (592.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 1746 (1.7 KiB)
        TX errors 780  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 4.3: wwan0 parameter after 5G activation

Chapter 5

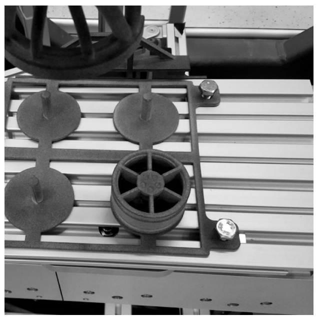
Computer vision implementation

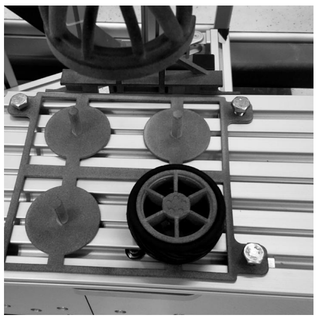
5.1 Data-set collection

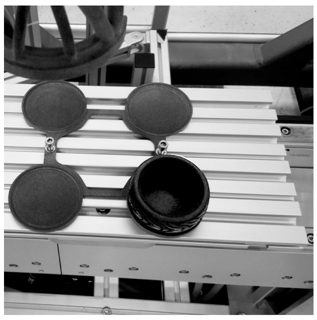
Data-set collection process was fully automated. For the purposes of the neural network, a data set of photos with each class was collected. In my AI there are 3 classes:

1. Wheel (plastic disk with rubber tire)
2. Plastic disk
3. Rubber tire

For each class, approximately 2500 photos were taken. The camera was located on the hand guiding controller and the delta robot was programmed to go through circles with different radii and vectors within those circles.


(a) : Data-set example for the DISC class


(b) : Data-set example for the WHEEL class


(c) : Data-set example for the TIRE class

The data collection process has been fully automated to save time. For the full operation of the neural network and good detection. An algorithm was executed that will allow the robot to go through different trajectories and angles. In addition, a Python script was launched on the hand guiding controller itself that took photos with a delay of 1-3 seconds (the delay was chosen randomly to avoid identical photos from the same places). Also, in each photo, the sharpness of the photo was checked, and if the photo was not sharp, it was not saved and photographed again.

To check the sharpness of the photo, the OpenCV library function was used.

```
cv2.Lapacian(img, cv2.CV\_64F)
```

The image is first converted into a grayscale and convolved with the following 3 x 3 Laplacian kernel.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{5.1}$$

Then the variance of the response was calculated [1]. Based on the obtained value, conclusions can be drawn regarding the sharpness of the image. With a high variance value, there is a wide spread of responses, both edge-like and non-edge-like, which means that the image is in focus. On the contrary, a low variance value indicates that there are very few edges in the picture. In other words, the sharper the image, the more edges there are. The next step is empirically setting a threshold value that will ensure that the photo is in focus. If the variance does not exceed the threshold, then the picture is blurry; otherwise it is sharp.

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{5.2}$$

The Laplacian kernel K (5.1) was obtained from the Laplacian operator using finite difference approximations(5.3), (5.4):

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x) \rightarrow \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} (x\ kernel) \tag{5.3}$$

$$\frac{\partial^2 f}{\partial y^2} = f(y+1) + f(y-1) - 2f(y) \rightarrow \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} (y\ kernel) \tag{5.4}$$

Applying (5.3) and (5.4) to the Laplacian operator definition (5.2), we get the Laplacian filter kernel:

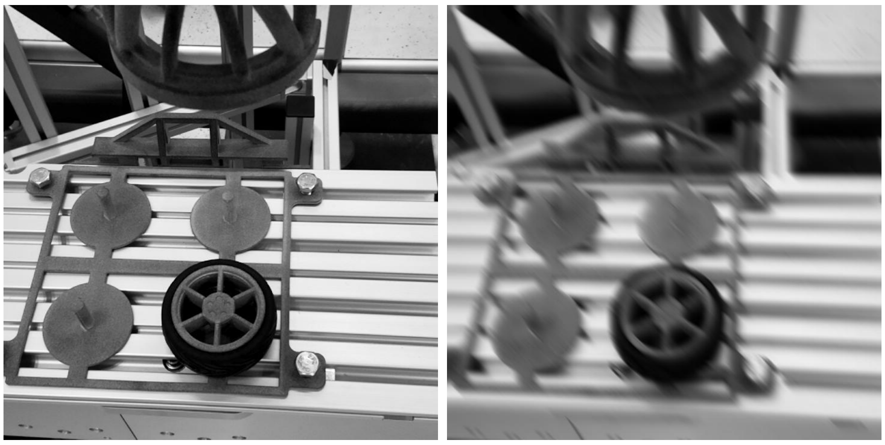
$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{5.5}$$

After finding the Laplacian, the variance will be calculated (in Python, just .var()) a If the image contains high variance, then there is a wide range of responses, both edge and non-edge, which means that the images are in focus. But if the variance is low, then this indicates that there are very few edges in the image.

Not sharp photos detection was made by the simple rule that the more blurred the image, the fewer edges it has. Therefore, the task is to empirically select a threshold that will ensure that the photo is in focus.

In practice, the usage of this detection is made by the cv2 function, in practice, the result is shown in the figures 5.2a and 5.2b

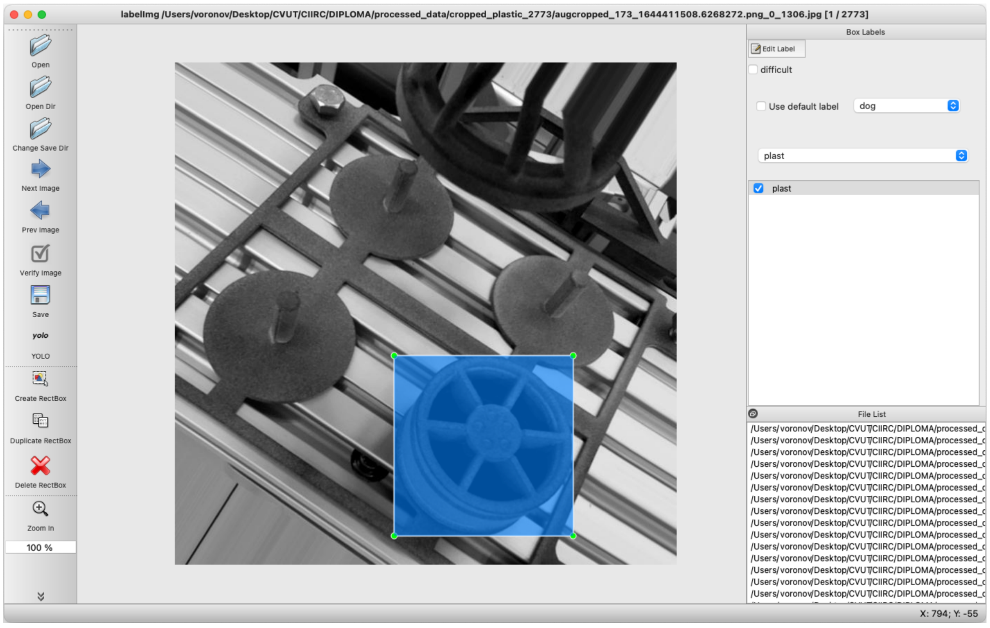
```
x = cv2.Laplacian(img, cv2.CV\_64F).var()
```



(a) : x = 313 **(b) :** x = 9

Figure 5.2: Example of sharp and not sharp photos

After the photos were collected, the bounding boxes with the defined class were added to each photo using the free labelImg software.



labelImg /Users/voronov/Desktop/CVUT/CIIRC/DIPLOMA/processed_data/cropped_plastic_2773/augcropped_173_1644411508.6268272.png_0_1306.jpg [1 / 2773]

difficult

Use default label dog

plast

✓ plast

```
/Users/voronov/Desktop/CVUT/CIIRC/DIPLOMA/processed_c
/Users/voronov/Desktop/CVUT/CIIRC/DIPLOMA/processed_c
/Users/voronov/Desktop/CVUT/CIIRC/DIPLOMA/processed_c
/Users/voronov/Desktop/CVUT/CIIRC/DIPLOMA/processed_c
/Users/voronov/Desktop/CVUT/CIIRC/DIPLOMA/processed_c
/Users/voronov/Desktop/CVUT/CIIRC/DIPLOMA/processed_c
/Users/voronov/Desktop/CVUT/CIIRC/DIPLOMA/processed_c
/Users/voronov/Desktop/CVUT/CIIRC/DIPLOMA/processed_c
/Users/voronov/Desktop/CVUT/CIIRC/DIPLOMA/processed_c
/Users/voronov/Desktop/CVUT/CIIRC/DIPLOMA/processed_c
/Users/voronov/Desktop/CVUT/CIIRC/DIPLOMA/processed_c
/Users/voronov/Desktop/CVUT/CIIRC/DIPLOMA/processed_c
/Users/voronov/Desktop/CVUT/CIIRC/DIPLOMA/processed_c
/Users/voronov/Desktop/CVUT/CIIRC/DIPLOMA/processed_c
/Users/voronov/Desktop/CVUT/CIIRC/DIPLOMA/processed_c
```

X: 794; Y: -55

Figure 5.3: Annotation process in LabelImg software

5.2 Convolutional neural network for the object detection

Initially it was planned to use only CV2 functions for object detection. But since the background image is an extremely unfavorable environment for detection (all details are gray with a reflection), it was not possible to provide a reliable algorithm that would detect objects under different lighting conditions.

Therefore, it was decided to use a simple neural network, which will be trained for 3 classes to detect all the necessary objects.

An example of using the Darknet neural network (YOLO V3) was taken [5]. To train this neural network, it is recommended to use convolutional weights, darknet53.

First, the data-set was divided into two parts - validation and training in a proportion of 30 to 70%. From each class, 30% photos were taken, which the neural network will further use to configure the hyperparameters of a classifier.

Before starting neural network training, it is necessary to configure its configuration for our case. To do this, the network parameters are added to the data/obj.data file:

```
classes = 3
train = data/train.txt
valid = data/test.txt
names = data/obj.names
```

Where:

- class - these are 3 classes of detected objects: wheel, disk, tire
- train is the path to the training files
- valid is the path to the files for the validation process
- names - the name of 3 classes for the neural network: plast, rub, wheel

It is also necessary to edit the cfg/yolo.cfg file: Where:

- Batch – the number of images and labels that are used in the forward pass to calculate the gradient and subsequently update the weights using backpropagation. In this implementation, this is 64 for the training process and 1 for the testing process.
- subdivisions - the number of mini_batches in the batch, because the GPU processes mini batch samples at the same time, and the weights will be updated for one batch. In this implementation, this is 16 for the training process and 1 for the testing process.
- max_batches - the training will be processed for this amount of iterations (batches), should be at least 2000 · number of classes

- Steps - at these amount of iterations the learning rate will be multiplied by the scale factor, should be $0.8 \cdot$ max_batches

Also, for a faster learning process in the Makefile of the Darknet neural network, the counting on CUDA graphics cores was activated:

```
GPU=1
CUDNN=1
```

For the neural network learning process, I chose the Google Collab online tool, where for a reasonable price it is possible to connect to extremely powerful video cards.
Starting the learning process:

```
./darknet detector train data/obj.data cfg/yolov3_training.cfg
                                         darknet53.conv.74
```

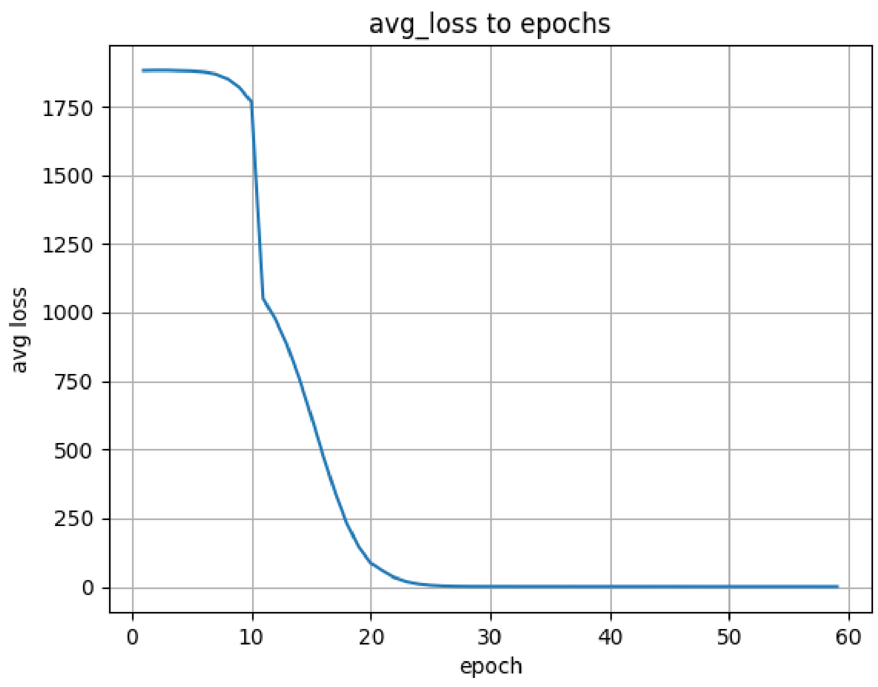


Figure 5.4: Loss value depend on epochs

Such a fast training 5.4 occurred because the neural network was already pre-trained and has only 3 classes.

After training, the neural network receives the weight file and we integrate it into the project on the server.

When the server receives a video stream, the detection is going to be used in photos from the stream. Commands to get the detections:

```
net.setInput(cv2.dnn.blobFromImage(frame, 0.00392, (640, 640),
                                  (0, 0, 0), False, crop=False))
```

```
out = net.forward(output_layers)
```

After that, the "out" array stores all detections of the percentage assessment for each class. After that, if the maximum grade of the class exceeds the threshold of 50%, then this class is displayed as the final one. For example, on figure 5.5.

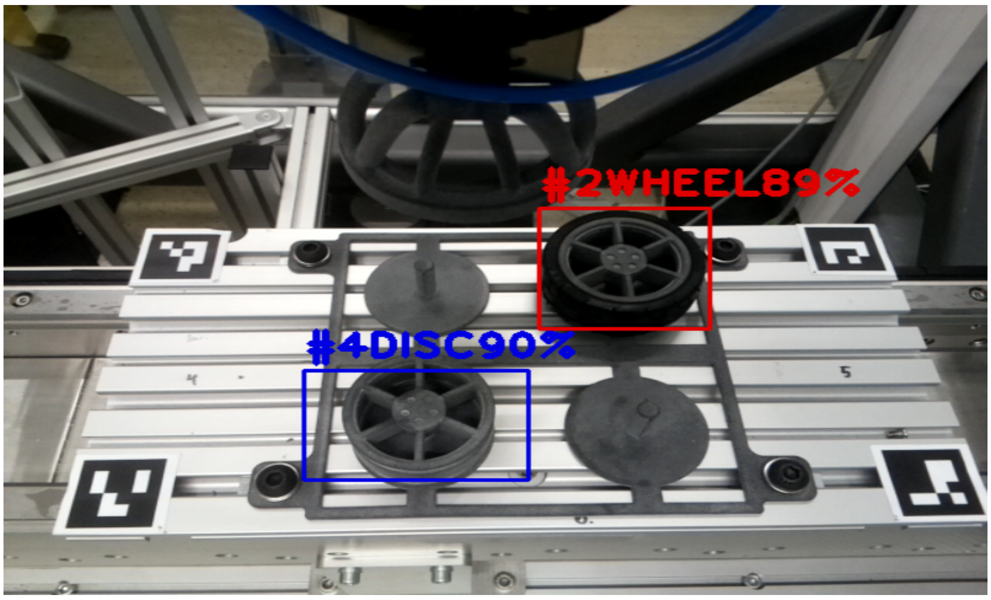


Figure 5.5: Screenshot of the RPI display after server detection

The number before the class name is the position where the object is located according to the server representation of position (5.1), and the number after the class name is the probability of detecting an object in percent.

1	2
4	3

Table 5.1: Representation of positions on the platform

5.3 Aruco tags

The ArUco tag [24]is a square marker in which an identifier is binary encoded. An important condition is a black frame outside the binary information matrix, which means the edge of the tag. Tags are used to transfer 3D planes to computer vision. Using the OpenCV - cv2.aruco library [17], it becomes possible to determine the rotational and translation matrix of the tag; in fact, using the 2D camera, we will determine the 3D position of the tag. In collaboration with several tags 5.7, it becomes possible to fully represent the 3D surface of an object.

In our case, one of the simplest matrix libraries was used, cv2.aruco.DICT_-4X4_50 (example on figure 5.6) which has 16 bits for binary storage of information and 1 layer of black pixels around - to define the frame.

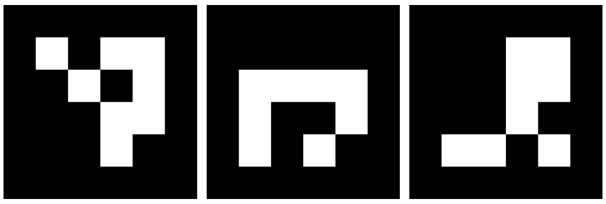

(a) : id = 0 **(b) :** id = 1 **(c) :** id = 2

Figure 5.6: Examples of the cv2.aruco.DICT_4X4_50 ArUco tags libruary

ArUco markers in this library have a minimum Hamming distance between any two codes - 4, therefore, incorrect detection is very unlikely. Of course, it would be better to use a tag library with a larger Hamming length, but the problem is that they have a lot more recognition bits and, therefore, a higher quality picture will be needed, which will reduce the data transfer rate over 5G (which is not so good so far).

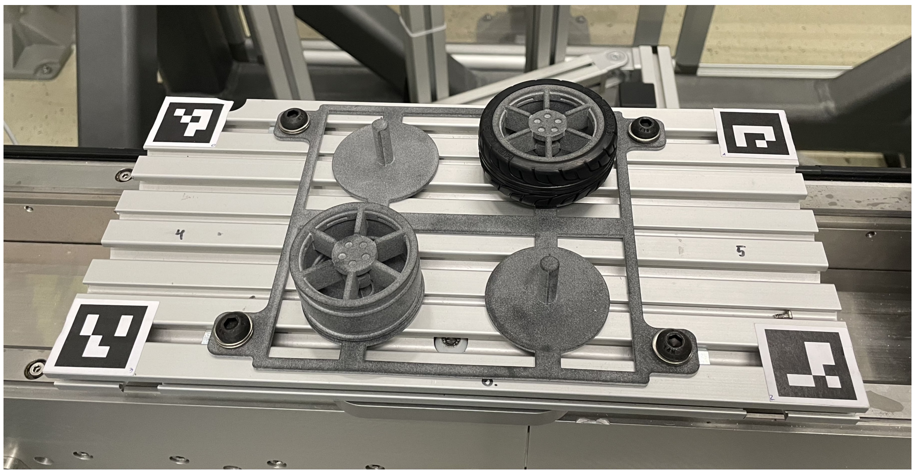


Figure 5.7: ArUco tags usage in DELTA robot

In addition to the fact that AI determines the class and coordinates of an object in a photo of the camera, it was also necessary to determine in which cell this object is located.

The problem is that the camera does not look at objects from above but from the side at different angles (since the camera is mounted on a moving gripper). Therefore, errors are possible if the algorithm determines an object in a 2D plane.

5.4 3D space representation

To represent the position of the plane with objects that the cameras can see, we placed four ArUco tags on each of the platforms. To detect ArUco tags, the CV2 open-source library function is used:

```
cv2.aruco.detectMarkers(frame, aruco_dict, parameters=param)
```

which returns 3 parameters:

- markerCorners - Nx4 dimention array, contains the coordinates of the corners of each tag
- markerIds – N dimention array, contains IDs of detected tags
- rejectedCandidates - the parameter is not used in our implementation, it contains the corners of Aruko tags that do not have the coding that is used in the implementation.

After detecting all available ArUco tags, the representation of space by one tag is predicted. With having a translational and rotational matrix and knowing the distances between the tags, representation of the space is made. The `define_world_pts` function in the `capture_stream.py` file (server side) is responsible for this - to which we send the ID of the first tag and based on the ID (that is, we know where each tag is located and based on this position we understand how to pre-build the cube) we build the cube (5.8).

For example, if the first tag to be detected was ID 0 (which is in the upper left corner), the cube would look like this:

```
world_points = np.array([
            0, 0, 0
            GRID_W, 0, 0,
            GRID_W, -GRID_H, 0,
            0, -GRID_H, 0,

            0 , 0, HEIGHT_CUBE,
            GRID_W, 0, HEIGHT_CUBE,
            GRID_W, -GRID_H, HEIGHT_CUBE,
            0 , -GRID_H, HEIGHT_CUBE

        ]).reshape(-1, 1, 3)
positions_3D=world_points * 0.5 * marker_size
```

Here the positions_3D array will mean a set of 3D points that are located relative to the start coordinate (0 0 0)

```
img_point, _ = cv2.projectPoints(postitions_3D, rvec, tvec,
                          camera_matrix, camera_distortion)
img_point = np.round(img_point).astype(int)
postitions_2D = [tuple(pt) for pt in img_point.reshape(-1, 2)]
```

Now we will get the points of the box in 2D with the starting point - the coordinates of the first detected tag. The algorithm then goes through the rest of the detected ArUco tags and corrects the predicted points. Thus, the corners of the box are bound to the ArUco tags.

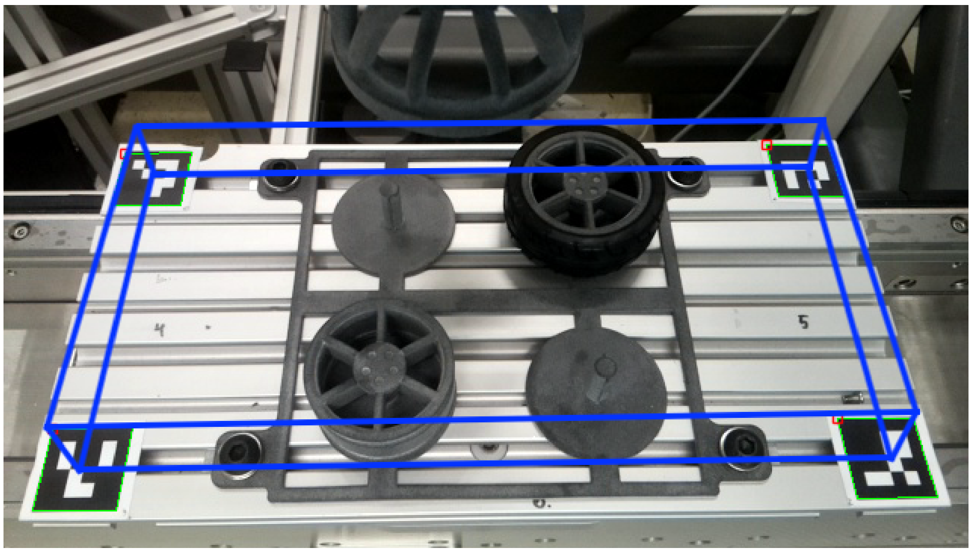
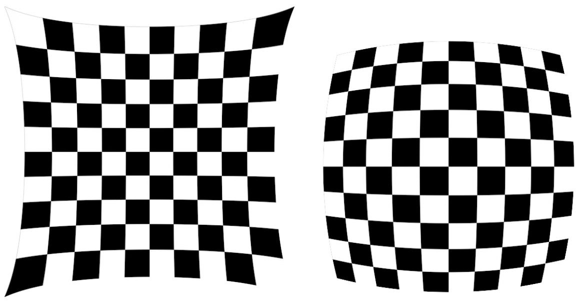


Figure 5.8: Representation of the surface in server part

An important part of using the cv2.projectPoints function is the calibrated camera. Since we use a translational and rotational matrix of ArUco tags, it is necessary that the camera does not distort the picture in any way; in other words, it is necessary to eliminate distortion(5.9). This is done using the cv2 library function - cv2.calibrateCamera [16] and an A4 sheet on which a special calibration grid will be printed. Also, for calibration, several dozen photographs of this sheet were taken on a camera that needed to be calibrated.



(a) : Pincushion distortion **(b) :** Barrel distortion

Figure 5.9: Examples of the camera distorsion

5.5 Outsource data computing

Since an RPI without CUDA cores is used and has a quite interesting feature - connectivity to a private 5G - it became possible to outsource computing capacities.

It works on the principle that Raspberry itself redirects the video stream to IP-streaming and then to the server part. Then RPI accepts the JSON parameters that the server sends. Thus, we get the opportunity to use any

GPU or CPU power, and it will begin to be possible to solve many more computationally difficult tasks.

The streaming process itself works via an open-source application, Mjpg-streamer. To use this application, only the standard libraries for working with JPEG files - libjpeg8-dev - should be installed and also the standard C compiler - G++.

Streaming gets started with the command:

```
./mjpg_streamer -o "./output_http.so -w ./www" -i "./input_ras
picam.so -x 640 -y 480 -fps 20 -ex auto -awb auto -vs -ISO 100"
```

As we can see from the team, this application will stream video in 640x480 resolution and 20 fps frame rate.

On the server side, to which the 5G modem is also connected, we can see the stream on the IP address in our case:

```
http://10.41.0.5:8080/?action=stream
```

Unfortunately, we do not have access to a 5G LAN distributor, so the IP address may change after the device is rebooted. In the Wi-Fi version, it was possible to perform an automatic search for an IP address using the MAC address with the command:

```
arp -a | grep "e4:5f:1:4f:a6:dd"
```

where the part with "arp -a" displays all connected devices in the local network, and the command:

```
grep "e4:5f:1:4f:a6:dd
```

- leave only a line with the IP address of the device with the MAC address of the RPI.

Unfortunately, there is no such possibility for 5G, so the IP address has to be entered manually after a reboot.

After processing the image (detection of objects by a neural network, ArUco tag detection, determination of the position of the object on the platform), the server part, using the Requests Python library, makes a POST request to the address:

```
http://10.41.0.5:5000/detections
```

with a JSON file that stores information about detections.

An example of the JSON file that Raspberry receives:

```
data = {'centroids': centroids,
        'rectangles': rectangles,
        'labels': labels,
        'confidences': confidences,
        'markers': index_marker }
```

where, for example, for the case of detection of two objects of different classes:

- centroids are a 2D array that contains the coordinates of the class detection centers, for example: [[x1,y1], [x2,y2]] → [[45,53], [305, 505]]
- rectangles are a 2D array that contains the sizes of detected objects, for example: [[width1, height1], [width2, height2]] → [[59,60], [61, 60]]
- labels are a 1D array that contains the class name of detected objects, for example: [class1, class2] → [wheel, tire]
- confidences are a 1D array that contains the percentage of the detection, for example: [percentage1, percentage2] → [98, 76]
- markers are a 1D array that contains the index of the object's place (based on the representation of the surface, using ArUco tags), for example: [place1, place2] → [1, 3]

5.6 Comparing CPU and GPU computing

For this project, the most important parameter is the speed of the detection process, which is taken as the detection frames per second. In other words, how many microseconds it takes to send an image, perform mathematical calculations, and receive data from the server.

Of course, my system is not a perfect product; some mistakes were made during development, for example, poor placement and poor antenna type. However, even taking into account all the shortcomings, the reaction speed has increased significantly (5.2) due to the 5G and out-of-the-box calculations.

RPI with:	Amount of CUDA cores	WiFi	5G	No outsource, RPI power
RPI without outsource	0	-	-	0.14 FPS
NVIDIA Quadro K2200	640	5 FPS	8 FPS	
NVIDIA 1080 TI	3584	5 FPS	14 FPS	
Macbook 6-Core Intel Core i7	0	4 FPS	4 FPS	

Table 5.2: FPS on different concepts

Chapter 6
DELTA robot

6.1 Principle's of the robot control

The Delta Robot is a parallel robot with three parallel arms. Delta robots are used in areas where speed and accuracy are important. The disadvantages are that the cost of such a kinematics is large - a suspended structure, and a rather small working area.

My work does not include the ability to program this robot; this is done by my colleague Vojta Sustr, who is responsible for operating this robotic workplace. In this work, I provide just general information on how the robot control works.

In general, PLC technological functions were used to control the robot's movement. Mainly, these functions are solving the inverse kinematics of the robot and also performing temporal and spatial interpolation of the end-effector motion along a line or a circle, observing dynamic constraints.

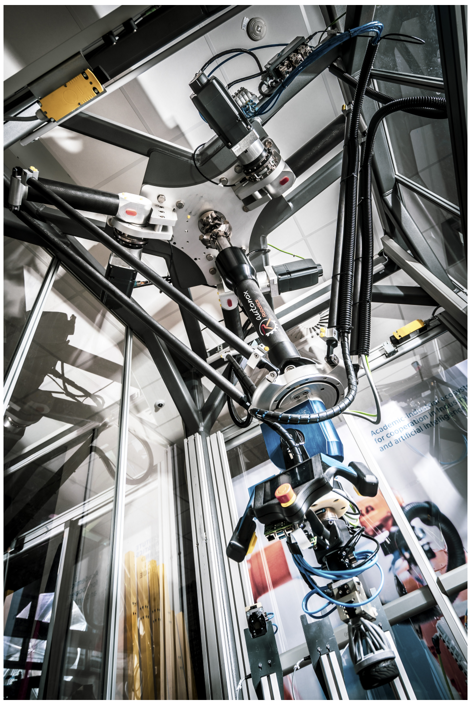
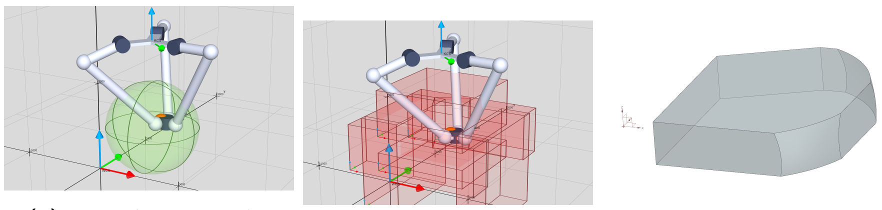


Figure 6.1: Delta robot in CIIRC

With respect to the goal of this thesis, the desired position is given by the hand guiding controller and is monitored in the PLC program. The PLC waits for the set point and moves in a straight line to the desired position. If a new set point is received during the movement, the program will append this new movement to the existing one and connect it (blend) with an arc, so the robot does not have to stop. A maximum of 3 linear sections are always connected in this way. The maximum Cartesian speed is limited, regardless of the set point of the changing position.

To improve the directional response in particular, with a large difference between the set point and the actual position, movement is not connected until the set point, but only to a certain intermediate point whose distance is determined based on the maximum robot speed and position set-point ground frequency.

The working space of the Delta robot is not as large as it might seem at first sight. Figure 6.2a shows the kinematic workspace of the Delta robot. Figure 6.2b already shows objects that can interfere with the delta robot. Unfortunately, since many optional robotic additions complicate our workspace, there are many zones, so the actual working area remains extremely small - Figure 6.2c.



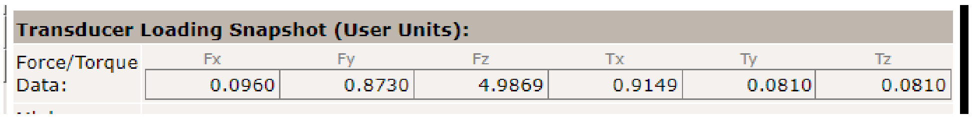
(a) : Working zone by kinematics **(b) :** Forbidden zones **(c) :** Real working zone

Figure 6.2: Delta zones

6.2 Force/torque sensor

The manual guidance mode is based on SCHUNK Mini58 force/torque sensor (6.4) data [2]. After the sensor was turned on, the data had to be set to zero to compensate for the weight of the tool and obtain a zero offset. Therefore, the sensor data were reset to zero. After attaching the console to it, the data showed a force of 4.98 N on the Z axis and 0.87 N on the Y axis. The sensor data perfectly represent that the hand guiding controller loads the Z and Y axes.

Figure 6.3: SCHUNK Mini58 force/-torque sensor



Transducer Loading Snapshot (User Units):

	Fx	Fy	Fz	Tx	Ty	Tz
Force/Torque Data:	0.0960	0.8730	4.9869	0.9149	0.0810	0.0810

Figure 6.4: Data from nullified sensor after attaching the hand guiding controller

To move the robot, the program checks the changes from constants in a cycle. If the force passes the threshold in the difference from the starting

values by ten units, the data normalization algorithm is activated. Empirically, human strength thresholds were found to be the loads that the remote control gives to the sensor.
The maximum value of the force sensor that a person can give by pressing on the hand guiding controller:

- 80 N in the X-axis
- 170 N in the Y-axis (large difference with the X-axis because the Y-axis is directed towards the person and the control is more convenient)
- 110 N in the Z-axis

Range of working values of the forces we have:

- -80 to 80 N in the X-axis
- -170 to 170 N in the Y-axis
- -110 to 110 N in the Z-axis

To allow the robot to respond to the given force, the program must normalize the force data to different robot steps. The range of robot steps was defined as 0.1 to 7 mm.
The normalization formula is:

$$step = \frac{(actual\_press - axis\_calm) * (new\_range)}{old\_range} + min\_new\_range \tag{6.1}$$

where:

- actual_press - the force we got from the sensor in this axis
- axis_calm - the force on this axis in stand-by mode
- new_range - the full range of steps, so it is from 0.1 mm to 7 mm
- old_range - the full range of forces, so in case of the X axis it is from -80 to 80 = 160 N
- min_new_range - minimum value from new range where we normalize values = 0.1 mm

For example in case of pushing the hand guiding controller in the left direction (in Y+ direction) with the force 130 N the formula going to be:

$$step = \frac{(actual\_press - axis\_calm) * (new\_range)}{old\_range} + min\_new\_range = \tag{6.2}$$

$$= \frac{(130 - 0) * (7 - 0.1)}{|-170| + |170|} + 0.1$$

6.3 Safety of the robot usage

In the project, two levels of safety are used. The first level of safety is the Safety Kinematics Toolbox in the PLC of the robot. This PLC program checks:

1. If the robot has not exceeded the maximum safe speed
2. If the robot is in the working area.
3. In case of hand guide mode, it also checks if both deadman switches are pressed (evaluated by the PLC safety card with communication to the main controlling PLC)
4. In case of auto mode, it checks if the robot doors are closed and locked.

If one of these conditions is not met, an emergency stop is triggered through the PLC safety card and the robot drives are brought to stop.

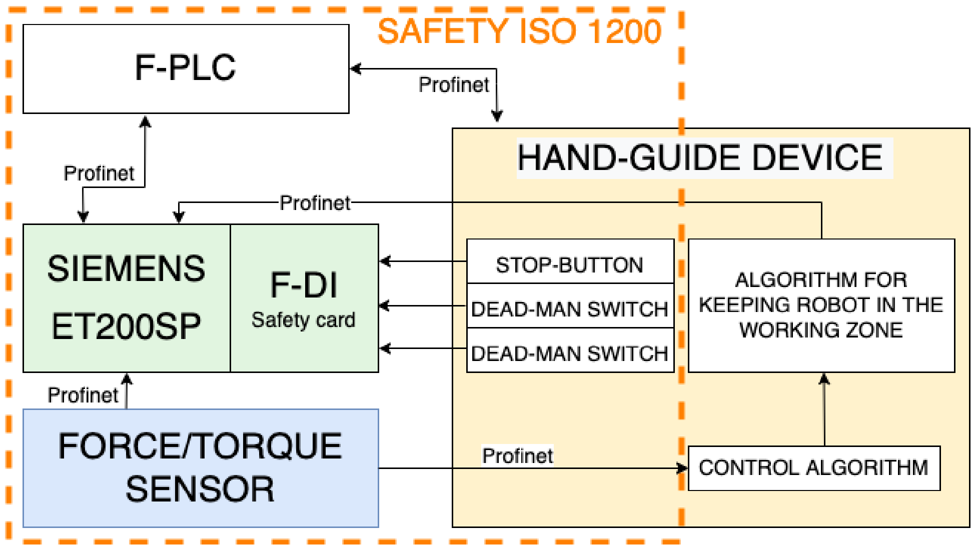


Figure 6.5: Safety architecture of the project

Figure 6.5 shows the standard ISO 1200 SAFETY for working with the robot in close proximity using hand-guide control technology. But an extremely important part of the project is the part in the hand guiding controller that does not allow the user to enter the safety zone, limiting the robot by 1 millimeter to the safety zone. Thus, it becomes possible to control the robot without constantly falling into a safety error when reaching the safety zone. In other words, the manual hand guiding controller additionally duplicates the safety system only by reducing it by 1 mm on each side.

Chapter 7

Conclusion

The work on this project started with the development of the first prototype on a regular RPi with a 5G development kit, which was placed in an FDM 3D printed case. Since then, a lot of progress has been made. The final version of the controller 7.1 consists of a real PCB board that was self-engineered and manufactured by a JLCPCB factory in China and a 3D powder-printed case that looks like a casting product.

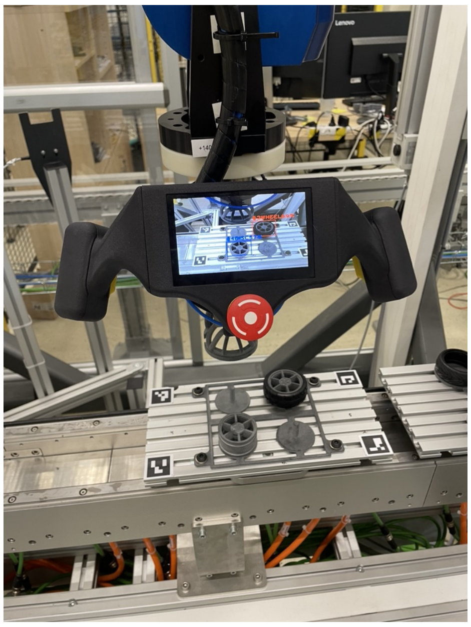


Figure 7.1: Final version of the hand-guided controller.

The development process of this project turned out to be more complex, than it was expected at the beginning. Despite my previous experience in PCB design and microcontrollers programming, the task of developing this controller in a way that the final product would meet the industrial standards, was challenging.

The complete solution included many stages. There was a need to develop hardware, computer vision programming, train and put into operation a neural network, develop software for outsourcing the calculation on the server, put into operation the Profinet stack and 5G communication, and develop software for processing forces of controlling the Delta robot. The other quite tricky part was developing a system with the help of which it became possible to represent space using ArUco tags.

Many compatibility problems were encountered and solved. These problems included the ability of the 5G modem drivers to only work with a particular version of the operating system kernel, or the fact that the process of the Profinet stack must be blocked on a separate core for a stable performance.

During the development process, management skills was gained, since my work was dependent on other jobs such as: programming a delta robot, making several versions of 3D printing.

The result is a device in which combine ready-made stacks and self-made software for the integrated operation of the camera, server, communications, and robot navigation.

As part of the CIIRC, I will continue to develop the project. There is a need to redesign PCB to replace antennas for 5G communication, as the current location of the antennas does not allow maximum communication speed. The final concept is also to create a hand-guiding system that, in collaboration with AI, will improve the precision of the hand-guiding process.

Working on the delta-robot hand guiding controller was extremely interesting because it combines multiple different knowledge and skills I had learned while studying the Cybernetics and Robotics study program to achieve the final result. Interesting part of this project was opportunity to work with RPi CM4, which allows us to make high-performace projects of very small sizes. Testbed for Industry 4.0, where this workplace is located, is part of the Czech Institute for Informatics, Robotics and Cybernetics in Prague.

Appendix A

Bibliography

[1] J. L. Pech-Pacheco, G. Cristobal, J. Chamorro-Martinez and J. Fernandez-Valdivia. *Diatom autofocusing in brightfield microscopy: a comparative study.* Institute of Electrical and Electronics Engineers, 2004.

[2] S. G. . C. KG. Product information force/torque sensor ft mini58. `https://schunk.com/fileadmin/pim/docs/IM0022168.PDF`. Accessed on 2022-03-17.

[3] Pigan, R., Metter, M. Fernandez-Valdivia. *Automating with PROFINET: Industrial communication based on industrial ethernet.* 2006.

[4] Profinet. Profinet gsd checker. `https://www.profibus.com/download/profinet-gsd-checker`, Aug 2021. Accessed on 2021-12-29.

[5] J. Redmon. Yolo: Real-time object detection. `https://pjreddie.com/darknet/yolo/`. Accessed on 2022-02-11.

[6] Chris Pietschmann. Raspberry pi 4 vs nvidia jetson nano developer kit. `https://build5nines.com/raspberry-pi-4-vs-nvidia-jetson-nano-developer-kit/`, June 2019. Accessed on 2021-09-08.

[7] CoreChips ShenZhen CO. Usb 2.0 high speed 4-port hub controller. `https://cdn-shop.adafruit.com/product-files/2991/1811151645_CoreChips-SL2-1A_C192893.pdf`. Accessed on 2021-10-19.

[8] Crazy Engineer. Nvidia jetson nano vs raspberry pi 4 benchmark. `https://www.arnabkumardas.com/topics/benchmark/nvidia-jetson-nano-vs-raspberry-pi-4-benchmark/`, August 2020. Accessed on 2021-09-14.

[9] Hiroce electric co. Board-to-board and board-to-fpc connectors. `https://cz.mouser.com/datasheet/2/185/DF40_Catalog_D31649_en-2301840.pdf`, Dec 2021. Accessed on 2021-10-19.

[10] Jason Arrigo. Input and output capacitor selection. https://www.ti.com/lit/an/slta055/slta055.pdf?ts=1652565332731&ref_url=https%253A%252F%252Fwww.google.com%252F, February 2006. Accessed on 2021-10-04.

[11] Monolithicpower. 2a, 18v synchronous rectified step-down converter. https://cz.mouser.com/datasheet/2/277/MP1482-1383982.pdf, September 2012. Accessed on 2021-10-04.

[12] PI North America. Profinet technology. https://us.profinet.com/technology/profinet/, Jan 2022. Accessed on 2022-01-11.

[13] Raspberry Pi Ltd. Raspberry pi compute module 4 io board. https://datasheets.raspberrypi.com/cm4io/cm4io-datasheet.pdf, Apr 2021. Accessed on 2021-10-19.

[14] Raspberry Pi Ltd (formerly Raspberry Pi (Trading) Ltd.). Raspberry pi compute module 4 - a raspberry pi for deeply embedded application. https://datasheets.raspberrypi.com/cm4/cm4-datasheet.pdf, April 2021. Accessed on 2021-11-14.

[15] rt labs. Profinet device stack rt-labs. https://rt-labs.com/docs/p-net/tutorial.html, Apr 2022. Accessed on 2021-12-29.

[16] W. G. I. Sergio Garrido, Intel Corporation. Calibration with aruco and charuco, opencv. https://docs.opencv.org/3.1.0/da/d13/tutorial_aruco_calibration.html, August 2013. Accessed on 2022-03-24.

[17] W. G. I. Sergio Garrido, Intel_Corporation. Detection of aruco markers, opencv. https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html, August 2013. Accessed on 2022-01-03.

[18] Siemens. Simatic s7 s7-1200 programmable controller - general technical specifications. https://support.industry.siemens.com/cs/mdm/91696622?c=59628332683&lc=en-CY, May 2014. Accessed on 2022-02-10.

[19] s. o. S. T. G. L. SIMCom Wireless Solutions. Sim8200ea-m2 5g hat. https://www.simcom.com/product/SIM8202X_M2.html, September 2020. Accessed on 2022-02-23.

[20] Texas Instuments. Lm22677/-q1 42-v, 5-a simple switcher®, step-down voltage regulator with features. https://www.ti.com/product/LM22677#design-development, April 2015. Accessed on 2022-10-03.

[21] WEBENCH Texas Instruments. Webench® designer lm22677. https://webench.ti.com/power-designer/switching-regulator/customize/4?VinMin=11&VinMax=13&O1V=5&O1I=3.5&base_pn=LM22677&AppType=FSW_ADJ&Flavor=5.0&op_TA=30&origin=pf_

panel&lang_chosen=en-US&optfactor=3&Topology=Buck&flavor=5.0&VoltageOption=5.0, May 2014. Accessed on 2021-10-19.

[22] Zachariah Peterson. Impedance controlled routing for boards made in altium designer. https://resources.altium.com/p/impedance-controlled-routing-for-boards-made-in-altium-designer, Feb 2019. Accessed on 2021-11-11.

[23] Zachariah Peterson. Pcb routing rules for single-ended and differential signals. https://resources.altium.com/p/pcb-routing-rules-single-ended-and-differential-signals, Oct 2020. Accessed on 2021-11-11.

[24] W. G. I. Sergio Garrido, Intel Corporation. Detection of aruco markers. https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html, August 2013. Accessed on 2022-02-13.