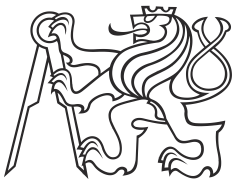
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Measurement

Automatic tester of device control elements

Bc. Hana Nováková

Supervisor: prof. Ing. Jan Holub, Ph.D.
Field of study: Cybernetics and robotics
January 2021

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Nováková Hana** Personal ID number: **434933**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Measurement**

Study program: **Cybernetics and Robotics**

Branch of study: **Cybernetics and Robotics**

II. Master's thesis details

Master's thesis title in English:

Automatic tester of device control elements

Master's thesis title in Czech:

Automatický tester ovládacích prvků strojů

Guidelines:

Design, develop and test the automated hardware component tester for human-machine interface (command and control devices – switches, ligths, buttons). The tester should consist of CNC for x, y, z and rotating axes including its control motion SW and recognition camers including the SW for image recognition of devices under test (DUT). Also test interface blocks to check the functionality of DUT (switches, lights, buttons) should be developed. Demonstrate the device functionality in a suitable manner on number of attempts and discuss the advantages and disadvantages of the developed solution (including deployment and operation times, costs etc.).

Bibliography / sources:

[1] https://www.bogotobogo.com/cplusplus/files/OReilly%20Learning%20OpenCV.pdf
[2] https://www.gadgeon.com/iot-services/robot-arm-based-testing/
[3] https://www.optofidelity.com/news/touch-panel-testing-6-axis-robot

Name and workplace of master's thesis supervisor:

prof. Ing. Jan Holub, Ph.D., Department of Measurement, FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **10.06.2020** Deadline for master's thesis submission: **05.01.2021**

Assignment valid until: **20.02.2022**

prof. Ing. Jan Holub, Ph.D. Supervisor's signature	Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

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

Date of assignment receipt	Student's signature

Acknowledgements

Firstly, I would like to thank the Czech Technical University in Prague for being my *alma mater* and providing a lot of knowledge and experience.

Secondly, I want to thank the professor Ing. Jan Holub, Ph.D., for his advice and willingness to help.

Thirdly, the thanks go to Eaton Corporation for letting me create something useful for my thesis. To Ing. Tomáš Vítek for the support, for all the cutting and drilling in the laboratory, I had no access to, and for countless pieces of advice. To Ing. Ondřej Ficner and Ing. Pavel Dědourek for the task specification and help understanding it correctly. I would also like to thank Ing. Jiří Čečrle for the mechanical advice, even though most of it were not applied due to budget limits.

And last, but not least, I would like to thank my loved ones for their support and understanding as well as proof reading.

Declaration

I declare that I have prepared the submitted work independently and that I have listed all the used literature.

In Prague, 5th January 2021

Abstract

Product testing is a critical factor in the industrial manufacturing process. Automation of the testing can bring clear and stable results and can keep running long term. This thesis discusses design and development of an automated tester for mechanical parts of device control components.

The robot was designed as an orthogonal CNC in order to test the components. A custom control board for motor timing was created, and its firmware was developed. Communication with the computer using a virtual COM port was designed. A computer program was implemented to do the path calculation as well as the image recognition of the component states. The user interaction is possible either via terminal or the graphical interface.

The thesis shows the design, development and validation of the testing robot. It evaluates the final error rate of image recognition and discusses the problematic parts.

The assembled robot can push the smart buttons and manipulate switches. It can recognise a light's state and its colour and can determine the position of switches.

Keywords: automatic testing, CNC, motor control, PCB, image processing, STM32

Supervisor: prof. Ing. Jan Holub, Ph.D.
Czech Technical University in Prague,Faculty of Electricity,
Technická 2,
Prague 4

Abstrakt

Testování produktů je důležitou součástí průmyslové výroby. Automatizace testování může přinést jasnější a spolehlivější výsledky a přinést možnost testovat dlouhodobě. Tato práce pojednává o návrhu a vývoji automatického testeru hardwarových částí ovládacích prvků pro průmyslové stroje.

Robot byl navržen jako ortogonální CNC stroj pro testování takových komponent. Byla navržena řídící deska pro ovládání motorů a k ní vyvinut firmware. Dále byla vytvořena komunikace s počítačem pomocí virtuálního COM portu. Počítačová část programu byla vytvořena pro plánování trajektorií robota a rozpoznávání obrazu pro zjištění stavu komponent. Bylo vytvořeno uživatelské rozhraní, jak v terminálu, tak i jako grafické uživatelské rozhraní.

Práce ukazuje návrh, vývoj a testování robota. Testuje také výslednou chybovost rozpoznání obrazu a popisuje problémy s ním spojené.

Sestavený robot dokáže mačkat tlačítka, otáčet spínači a rozpoznat jejich pozici. Dále dokáže rozpoznat, jestli komponenty svítí a případnou barvu tohoto světla.

Klíčová slova: automatické testování, CNC, ovládání motorů, plošný spoj, zpracování obrazu, STM32

Překlad názvu: Automatický tester ovládacích prvků strojů

Contents

Figures

Tables

Chapter 1
Introduction

This thesis describes the design, construction and programming of an automatic machine control tester focusing on its build, bench testing, firmware development, image processing algorithms, communication protocol with a computer and motion planning.

The paper presents a plan and construction details for building a CNC machine for device control element testing. In the paper a printed circuit board is designed, installed, set up and firmware for the microcontroller is developed. Interface with a PC is designed. An image processing algorithm using a camera input for detection of components' position and state of their lights is prepared. The system can plan the trajectories for the CNC to perform the motions that are to be tested.

1.1 Chapter overview

Chapter 2 introduces the tested components and lists their available variants. There is also a section analyzing other third party automated testers which are already built. Then an orthogonal CNC design is discussed, focusing on construction options, from which the designer can choose.

In chapter 3, the final design of the robot is discussed. A skeleton design built from aluminium profiles and the interconnecting parts are shown. There is also an electronic selection and computer connection described, as well as the controller board design. The final assembly is also shown there.

Chapter 4 then shows the software portion of the thesis. A program for an STM32 microcontroller, which is part of the controller board PCB, is described. This program controls the motor movements and communicates with the personal computer. The main PC program used is then described in greater detail. The subsections focus on the protocol between both parts of software, camera interface and image processing as well as trajectory calculation.

Image recognition consists of algorithms for recognising light component states, colours, and the positions of both illuminated and dark switches.

1.2 The desired use case

The contracting authority commissioned a robot for automatically testing its device control elements described in chapter 2.1. Those components are smart buttons, switches and light modules, intended to control some kind of large industrial device, with which they interface using the SmartWire-DT bus.

They requested a CNC robot which would simulate human interaction with the components. It would receive orders via a computer terminal and return data the same way. The test engineer using it would run a script containing the requested actions such as pressing a button on specific coordinates or reading the light state of some component. Simultaneously, another test program would be simulating the large industrial device on the SmartWire-DT bus, changing the light states, verifying that the smart button registered being pressed by the robot, and so on. By simulating and continuously evaluating both ends of the components' interaction with their environment, their correct functionality would be verified.

Chapter 2

Research

2.1 Devices under test

The contracting authority has commissioned the robot to test their M22 product line, which features control elements mountable to a slot of such dimension. The catalogue can be found at [6].

There are several main types of components. There are pushbuttons, double and four-way buttons as well as switches and lights. Some components are operated by a key, some joystick components and potentiometers, although those types are not subject to this testing.

The components are composed of two main parts. The front physically interacts with the user and provides mechanical input and allows the light output to pass back to the end-user. The rear portion consists of a mechanism that reads which position of the component is currently set, and a light to serve as an indicator. Both are optional.

This chapter consists of an overview of the buttons, switches and light components, which were the tested subject supplied by the contracting authority. The component's part number, used as an ordering number, encodes the components' features.

All codes start with the "M22-" prefix, which indicates the product line. Then there is a code of the component type followed by a colour and a labelling variant. The code looks like:

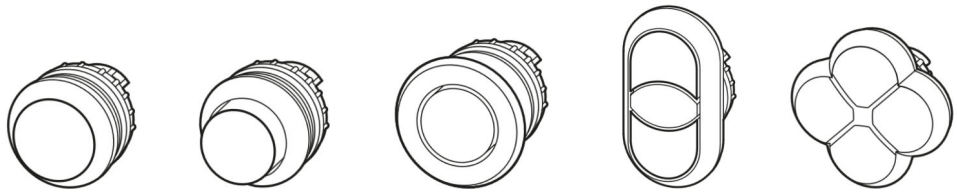
```
M22-<code>-<color>-<label>
```

2.1.1 Buttons

There are several types of button components available. The most common types can be seen in figure 2.1.

For every button, a whole palette of the colours is available. The flush and extended types can be in black, white, red, green, yellow, blue, grey colours. The component can also be labelled with a circle or a vertical line. For the mushroom type and four-way buttons, the colours are the same, excluding grey.

Double buttons can use a combination of green and red or white with black; both can be manufactured with glyphs.



(a) : Flush **(b) :** Extended **(c) :** Mushroom **(d) :** Double **(e) :** Four-way

Figure 2.1: Button component types [6].

Code of the button component

Figure 2.2 shows frequently used codes. For example, the code can look like:

```
M22-DRH-W-X1
```

It indicates that the component is an extended button, with a maintained arrest mechanism. The colour of the press area is white with a vertical line.

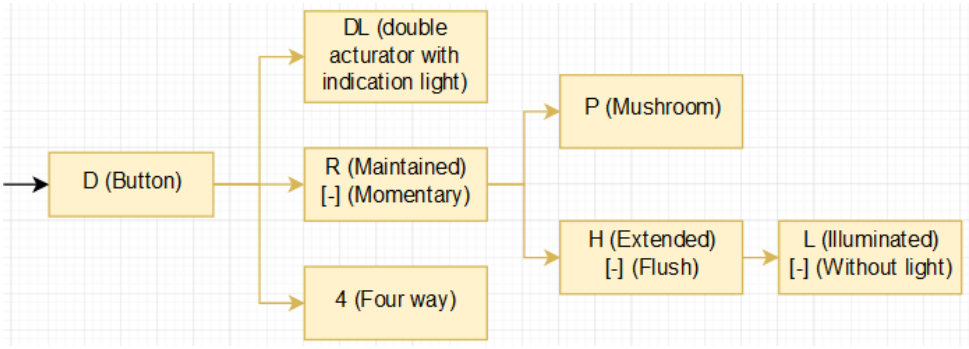


Figure 2.2: The diagram of possible codes of button components.

There are more codes with specific meanings, for example, the double button component can feature a combination of flush and extended push area, for which the F and M letters are used, but these are beyond the scope of this thesis.

The light of the button components

Some of the button components come with the option of having a coloured translucent cover so that there could be a light installed behind them. The double button always has a white translucent space between the push areas. The mushroom and the four-button component do not support lights at all, whereas all the remaining button component's versions can have the translucent plastic cup with a white, red, green, yellow, blue or orange tint.

2.1.2 Switches

There are only two visible types of the switch component; they are shown in figure 2.3. The mechanism behind the different head is the same, and

the contracting authority does not mind testing only one of them, so the switching tool does not have to be changed mid-testing. The thumb-grip variant was selected to be the tested one since the grip would be easier to achieve, and the thumb-grip supports light.

(a) : Rotary head

(b) : Thumb-grip

Figure 2.3: Switch component types [6].

Code of the switch component

The code diagram is shown in figure 2.4. For the rotary head, the only colour is black, with some labels.

However, the thumb-grip has space for the colour cup. The cup can be translucent or opaque. The supported colours are white, red, green, yellow and blue.

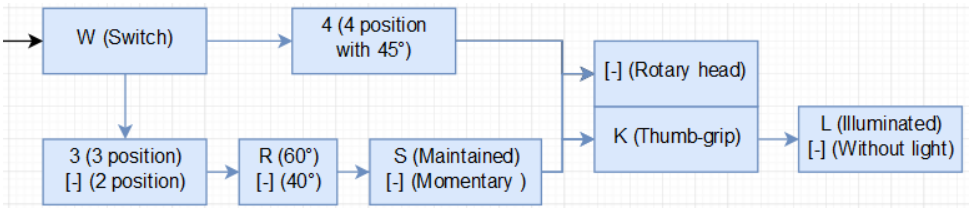
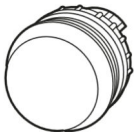


Figure 2.4: The diagram of possible codes of switch components.

For the three-way switches, the momentary and maintained mechanisms can vary as well as the 40 ° and 60 ° rotation angle.

2.1.3 Lights

The last tested component type is a light. The component has two versions, shown in figure 2.5. Both exist in a classical and a compact version; the difference is only in size.



(a) : Flush

(b) : Extended, conical

Figure 2.5: Light component types [6].

Code of the light component

The diagram for the light component code can be seen in figure 2.6. The lights always have a translucent plastic cup. The supported colours are white, red, green, yellow and blue. For the classical version, the orange colour is supported too.

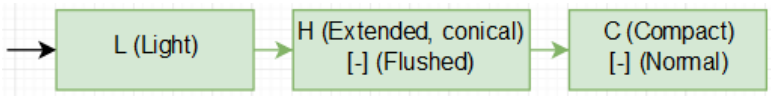


Figure 2.6: The diagram of possible codes of light components.

2.1.4 Rear parts

The rear part of the component is used for communication with some equipment. It can read the button or switch position as well as show the light to the user.

The front of the component mostly dictates sensors present in the back part as the part consists of the sensors that read the front component's position. It can have one or two inputs. In the case of the four-way input, two rear parts need to be mounted.

For now, only a white, red, green or blue LED can be mounted into the light component. Soon an RGB LED shall be added to the list.

The light module can be separate or combined with the inputs for the button and switch position reading.

The rear components have their own set of codes. Since the automated tester does not have any access to this part of the component, the codes are irrelevant. However, they can be found in [6].

2.2 Automated testing

In the industrial development process, there are monotonous tasks which require a long-term focus and continuous operation for extended periods of time. In those scenarios, the use of human resources can be impractical and better replaced by automated solutions. An automated tester is able to work without rest. It does not make mistakes typical for humans, which are easy to do under stress or fatigue.

Some testers use a robotic arm, for example [11]. These testers are non-invasive, and typically the tested subject does not have to be modified to interface with the tester. Such approach is well suited for touch screens, mobile applications and many others. A tester for touch screens can be seen, for example, in [29].

These testers offer considerable freedom of movement, making them suitable for demanding applications such as stylus angle simulation. However, such arm's cost is high, which makes it suitable only for projects with a large

budget, high demands for safety and reliability (medical applications), and so on.

In contrast, the cheapest application testing can use software API to simulate user input. However, it does not directly test any hardware, only the software reaction to the inputs.

When hardware testing is needed, and the budget is limited, orthogonal CNC robots can be fully sufficient. In those designs, the working area is somewhat limited, and the tool inclination is usually fixed. However, lots of applications do not need such features. The axes are independent of each other; the coordinates are directly linked to the axis in which the robot does the movement; contrary to the arm movement, which needs to compute inverse transformations to navigate.

2.2.1 Existing automatic hardware testers

Automatic hardware testers are getting more common in the industry. There are lots of different kinds, testing different things. In this chapter, some of those focusing on hardware testing are listed, but there are even more out there.

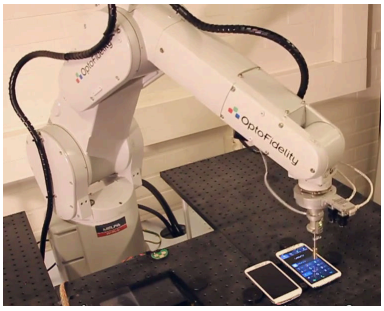
Touch screen testers

Numerous machines are specialized for tapping on touch screens, some are using a robotic arm, but other solutions exist too. The tool of such machines is a capacitive touch pen simulating a human finger, or a dedicated stylus, which interacts with the screen's sensors. The test scenario can be programmed to touch specific locations, or measure "finger" proximity or pressure response. Multi-touch capable screens also exist, which can require the machine to have more tools to perform hand gestures.

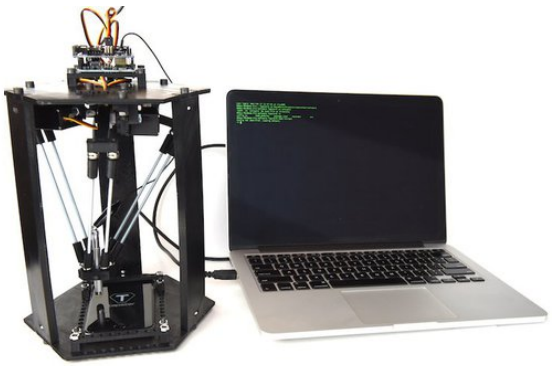
OptoFidelity. The touch panel tester from this company uses a 6-axis robotic arm [29]. The solution can test touch screens up to 27 inches, with a customizable ability to adapt it to bigger ones. The robot is shown in figure 2.7a.

The robot can also test the force applied to the styluses and its response on the writing angle. The result comes in the shape of numerous statistics with graphs.

Tapster 2. Different kind of touch screen automatic tester. This robot is built as a delta robot, shown in figure 2.7b. The robot is designed to simulate a human touch on the touch-screen. The testing area is significantly smaller than the previous one. The maximum size is 140 mm×80 mm, which is only suitable for mobile phones. However, the robot is compact and does not need any dedicated room space. It only needs a connection to the computer and can run anywhere. The robot can be found at [32].



(a) : The OptoFidelity touch panel testing robot[29].

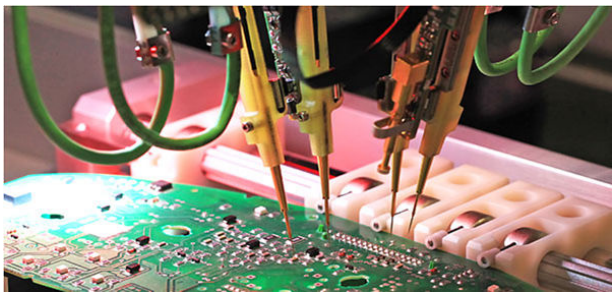


(b) : The Tapper 2 [32].

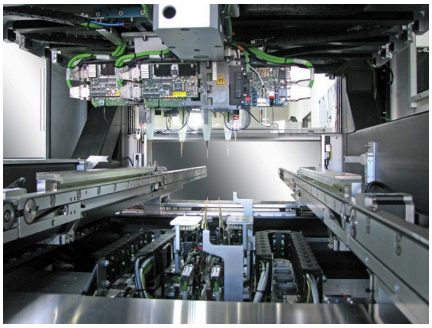
Figure 2.7: The touch screen tester robots.

Flying probe testers

Flying probe test is a method of testing printed circuit boards (PCBs) during the manufacturing process. These testers exist in many variants since they are used in the electronic industry. Some information about the flying probe testers can be found at [7], [24], [17] or [1]. The system has several spring-loaded conductive probes, which connect to the PCB pads check whether the connection is where it should be, and that no neighbouring signals are shorted together. The flying probes testers can be found in figure 2.8.



(a) : Detail of flying probe test[24].



(b) : Inside of the flying probe tester[7].

(c) : The flying probe tester [1].

Figure 2.8: The flying probe testers.

The testers are usually universal and do not require any additional nor manual configuration; the user just provides the CAD data, and the machine tests all the pads quickly. It is mostly used to check the state of an unpopulated PCB. It can also be used to check if the right resistors, capacitors or Zener diodes are populated.

Automatic Headlight Tester

This testing is used on a large scale in the automotive industry. For example, at [4], an automatic headlight tester can be found. The tester measures light intensity coming from a car at different angles. The intensity is measured, and it is determined whether it meets the legislative standards for road transportation.

The tester processes the images, automatically finds the headlight beam even with the low beam setting. The positioning system is doubled to prevent interference of the ambient light. The system also has an online adjustment mode, which makes it easy to operate. The tester can be seen in figure 2.9.

(a) : The headlight tester in use.

(b) : Detail of headlight tester.

Figure 2.9: Automatic Headlight Tester [4].

Other testers

There are lots of other testers in the world. Some of them can be seen in figure 2.10. For example, the automotive usually at least partially test the car electronic systems such as ignition, door locks and audio equipment automatically.

There is an RFID technology tester [2] for a card and passport reading. It tests the working distance and reliability of the readers.

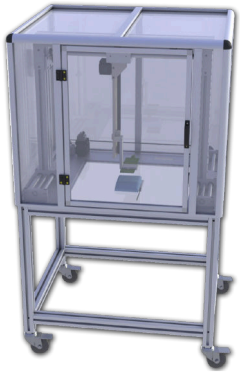
Another tester verifies the Samsung smartphone's ability to survive in the back pocket of the trousers [21] while the human sits on it.

For package delivering services, an Automated Drop Tester was created [30]. The tester takes the package and drops it to determine if the package's

wrapping is satisfactory to survive the delivery process. The robot consists of a robotic arm with a vacuum gripper, camera and a 3D vision system.

There are also machines for testing material properties, such as plastic or metal. They test the deformation after pulling or heating it. These testers exist in more variants; some can be found at [46] and [15].

There are lots of other automatic testers. Most of them are purposefully built for one type of testing and need to be changed to adapt to anything else. The machines are usually dimensioned for one manufacturing line of an industrial product, making it useful only for big companies.



(a) : The RFID tester [2].

(b) : The automated drop tester [30].

(c) : The plastic material tester [46].

Figure 2.10: Automatic Testers.

2.3 The orthogonal CNC design

The CNC (Computer Numerical Control) machines are used in the industry as well as hobby spheres. The most common usages nowadays are milling cutters and 3D printers.

The internet provides some creditable or less creditable articles about the CNC design. Some of the articles can be found, for example, at [28], [33], [26] and [43].

Frame

The design should start with the robot frame. It needs to be firm enough to hold the robot and not allow even the slightest movement. The most important thing about the frame is the material. Some handymen use wood since it is cheap and easily sliced in various sizes and shapes. However, wood strength is debatable. It can be satisfactory for home applications; for industrial use, the wood is not a long term reliable support. Most solutions can be found using aluminium, which is a good compromise between the price and the strength. For most applications, the aluminium is firm enough.

There are some solutions with steel instead, the robot is then heavy to move, but firm to be able to operate with almost anything.

Another part of the frame design is the shape. The frame needs to be able to hold the working space as well as support the robot's movement parts without any problem. Still, the size needs to be reasonable enough to fit in the given room or a part of it.

It would be useful to make the frame as simple as possible to avoid complicated pieces which could be expensive and hard to get.

The guides

There are several options available on the market. Every one of them has its advantages and disadvantages which need to be taken into account. Figure 2.11 shows a commonly used types of guides.

Precision rods. The rods are cheap and easily obtained. However, they are not firm enough for designs bigger than 500 mm in length. The rods themselves are also not able to transfer the motor movement into the robot movement. They are used to support the transfer methods, which can be suitable for long designs if the transfer is firm enough.

Supported rods. The enforced variant of the precision rods. They are more expensive than the rods themselves; however, the price is low enough compared to the rails.

Rails. Another solution could be the rails. They are expensive, but in the case of precision critical applications, they are commonly used. They are also sensitive to dirt, which forces the solution to be housed in a somewhat sterile environment.

Other methods. Numerous custom solutions can be cheaper but not precise enough or not easy to manufacture.

(a) : The rods [34].

(b) : The supported rods [34].

(c) : The rails [34].

Figure 2.11: The guides.

Transfer methods

There are several methods to transfer the movement from the motor to the robot. The commonly used types are shown in figure 2.12.

Belts. They are cheap and easy to procure. This method is used mostly by model makers and is not useful in the CNC. The precision is low, and the belts can change their parameters in long term use. The longer the belt is, the lower is the precision which makes it suitable only for small applications.

Gears. The gears are significantly more expensive than the belts. They use the same principle. This method, however, is more stable. It is solid without the chance of the parameters changing.

Trapezoidal screws. One of the cheaper methods which can be used in all directions. The trapezoidal screws are commonly used as the compromise between the cost and precision. However, they have backslash which needs to be compensated.

Ball screws. An expensive version of a transfer. However, industrial applications are common, and it is possible to get them cheaper in big companies. There is no backslash to be compensated. The position is exact without any drifting away even when the robot is in a vertical position with motors powered off. However, the ball screws are sensitive to dirt.

Transfer methods mounting. All the mentioned methods need to be mounted with bearings. There are lots of mounting types and shapes available.

(a) : The belt [34].

(b) : The gears [34].

(c) : The trapezoidal screw [34].

(d) : The ball screw [34].

(e) : Bearings[34].

Figure 2.12: The transfer methods and their mounting.

Electronics

For the operation and control of the CNC, several electronic parts are needed.

Motors. To be able to move the robot, at least one motor for each axis is required. There are more types of motors, three of which are the most commonly used; the stepper motors, BLDC motors and the servo motors. The

detailed description of advantages, disadvantages and the operation principle can be found in [13].

The **stepper motors** are more precise. The control is easy with the PWM timers, which can be achieved even with cheap microcontrollers. However, they can be slower than the BLDC version. They are designed to have high holding torque and do not need a feedback mechanism if such a motor's reliability is high enough. These motors are on the cheaper side.

Servo motors are precise, and the motors have speed control integrated. They are slower than the DC motors, with higher torque. Servo contains an encoder or another type of position sensor. The control loop with the sensor causes the motor always to try to correct the speed to the desired one, which can cause some twitching in the steady position. This type of motor is usually more expensive than the other ones.

BLDC (Brushless DC) motors are designed for the motion's smoothness, which could be desirable. However, the precision could be lower than in the stepper case, and the control of this motor could be challenging. The motor needs to have some feedback mechanism for a user to tell its position, for example, Hall effect sensors for measuring the magnet position or measuring the electromagnetic field. The BLDC motors are usually more expensive.

Motor drivers are an essential part of the system. They are transferring the power from the power supply to the motor's windings. They usually provide optical separation between the power supply and the control electronics.

Limit switches. The limit switches can be done in two ways; proximity ones, mostly the inductive types, and the touch ones. Both types can come in the normally open (NO) and normally closed (NC) versions. Usually, the NC sensors are used so that a wiring problem can be detected.

Power supply. The supply selection is vital to run the motors with the desired torque and speed. It is recommended to have more power than needed so that the system does not fail if consuming more.

Control

The CNC needs to be controlled. Usually, the computer is used to create the routes which need to be followed by the robot. For this, the commonly used protocol is G-code. The computer then communicates with a breakout board which controls the motors.

However, the available breakout board solutions usually use parallel port (LPT) cables that are not standard in modern computers. Either a different breakout board needs to be obtained, or an older dedicated computer must be used.

The breakout board is the connection between the motor and the computer. The board takes care of motor timing and speed as well as reading the switch signals. It usually receives the G-code from the computer and transforms it into driver signals.

Chapter 3

Hardware

3.1 CNC design

The CNC design consisted of several smaller tasks which needed to be done before the components could be manufactured and assembled together.

3.1.1 Robot shape

There are multiple basic CNC shapes which suit the given requirements equally well; some of them were described previously in chapter 2.3. This chapter focuses on designing a cube-shaped machine which was built as part of this thesis.

Several factors were affecting the final shape of the testing robot. The first requirement was the size of the working area. In the beginning, the working area was intended to be a 2D area of 500×500 mm. The second condition was the robot's size itself. Since a considerable amount of space was needed around the working area, the final size of the robot would be too large. In respect to this, the final dimensions of the working area were changed to 400×700 mm.

However, the space around the working area was not the only parameter in the shape of the robot. Since there was a camera for image recognition needed, there was the condition for it to see and be able to focus the image of the entire tested place. The camera needed to be firmly connected to the robot at a certain distance from the tested components. The camera attachment had to be secure enough to avoid frequent manual camera repositioning or lens refocusing. It would also help with image processing of the scanned area.

In respect of the conditions above, the size of the robot ended as a 1120×890×910 mm (h×w×d) block. Since it is relatively easy to connect the "X" aluminium profiles rectangularly, this material had been chosen for the building of the outer block. The designed robot can be seen in figure 3.1, which was created in FreeCAD [10].

There is no working area visible in the figure. The area is placed in the front of a functioning prototype, here marked as 1. There is also no camera visualized in the figure, in prototype, it was placed far back, on the spot marked as 2.

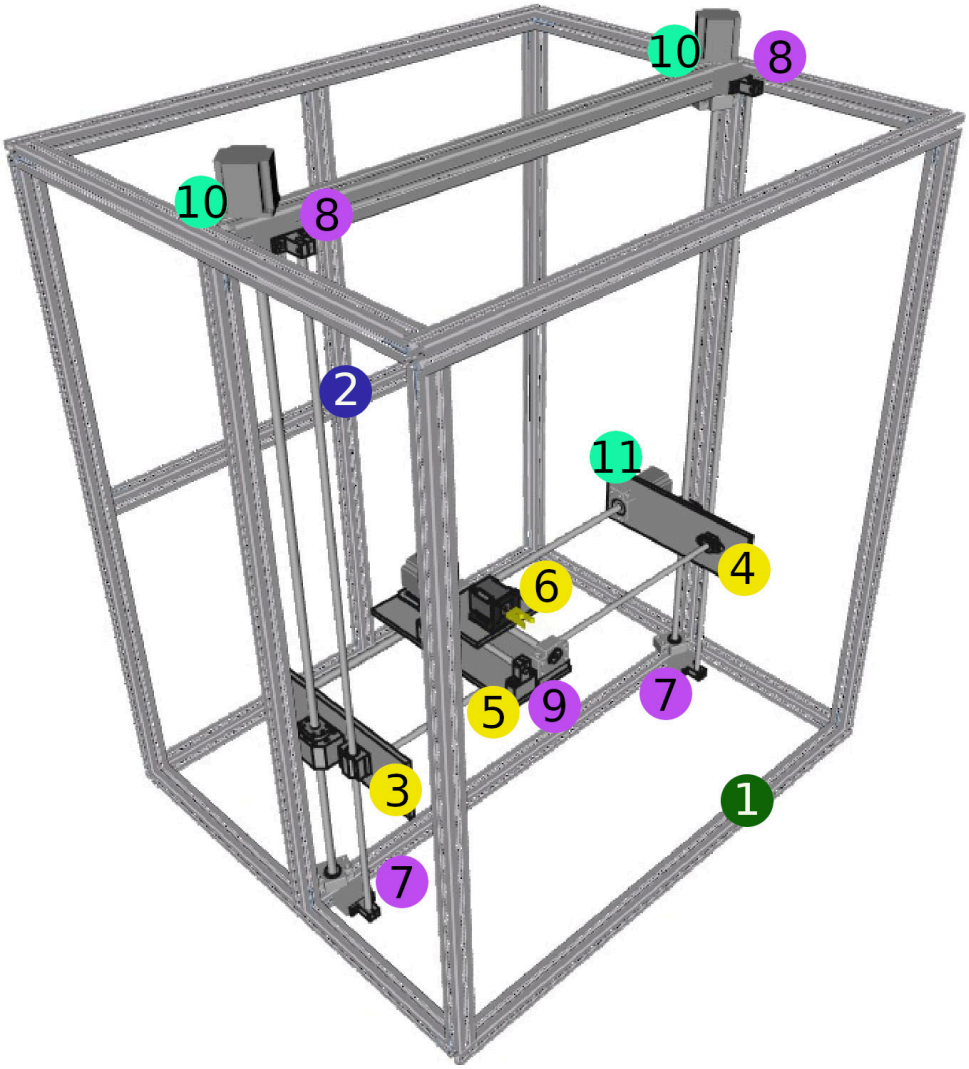


Figure 3.1: The robot layout

3.1.2 Lead type selection

From the available linear lead options, a ball screw with linear slide was selected for its precision and the ability to retain its position even in the power-down state. This choice is also favourable due to cost, should this machine be mass-produced.

A common type of the ball screw has a 12 mm diameter screw, which is entirely sufficient in this case. There is also linear lead with a diameter of 10 mm next to it, to help bear the mass and to provide linear motion against undesired rotation.

As shown in figure 3.1, there are four pairs of ball screws and linear leads. Two pairs make the Y (vertical) axis, to help bear the mass of the tray in the middle of the robot. It also ensures the motion will be possible regardless of the motion in the other axis. In the Y-axis, the ball screws have 1000 mm

length each. In the X-axis (the horizontal one), there is only one pair, and ball screw length is 600 mm, creating a 400 mm effective operating space. The ball screw in the Z-axis (forward one) is only 250 mm long.

It is crucial not to forget that the length of the ball screw also includes ends for mounting the bearing and connection to the motor. So this length is not fully used for the motion. There is also some space around the ball screw nut, which is occupied by the attachment, so this is why, for example, X-axis has only 400 mm effective motion space although a 600 mm ball screw is used.

After the skeleton was assembled, the linear lead in the X-axis had such a loose grip strength, that the robot needed one more linear carriage in series, to eliminate the vibrations while in motion.

3.1.3 Motor selection

There are two types of stepper motors in the robot and five units in total. Four motors are of a higher torque type, driving the linear motion and one is of a smaller type performing the rotation movement for manipulating the tested switches.

The needed motor torque can be calculated from the screw gear and the expected weight which is to be moved. The calculation of minimal needed torque for Y-axis can be done by equation (3.1). The $m = 10\,\mathrm{kg}$ is the expected mass of the moving parts, $g \doteq 9.814\,\mathrm{m{\cdot}s^{-2}}$ is gravity acceleration on Earth, $r = 12\,\mathrm{mm}$ is the radius of the ball screw, and the gear ratio is 25 %.

$$\begin{aligned} T_{min} = m \cdot g \cdot r \cdot gear = 10 \cdot 9.814 \cdot 12 \cdot 10^{-3} \cdot 0.25 \\ \doteq 0.294[\mathrm{N\,m^{-1}}] \end{aligned} \tag{3.1}$$

The calculated minimal torque is split between two motors according to the robot shape, so the minimum limit for motor torque used in the Y-axis is $0.147\,\mathrm{N{\cdot}m^{-1}}$. The selected motors needed to be way more potent than that, to accelerate the mass of the robot reliably. The selection was done for Y-axis and then used for X and Z axis as well, so the system of motion is unified.

The linear motion motors are 57HS09 hybrid stepping motors (datasheet available at [22]). The motors can connect as bipolar parallel or series connection, or as unipolar connection. In this case, the bipolar series connection was used, which results in 1.3 Nm holding torque per motor, phase current 2.0 A and phase resistance $1.6 \pm 10\,\%\,\Omega$. Each motor weighs 0.6 kg. The size of the motor is NEMA 23.

The motor cannot be used without an appropriate driver controlling it. The M542 High-Performance Microstepping driver was used. This driver's output current is adjustable between 1.0 to 4.2 A using supply voltage from 20 to 50 VDC, which makes it suitable for the motors used in the linear motion.

A different type of motor was selected for the rotation axis because there was no need for that much power. It is also advantageous to have less mass at the end of the Z-axis since it will be less to support, and the force lever will be smaller with less weight. The stepper motor selected is 17HS4417P1-X4

which is NEMA 17 size (datasheet available at [42]). It is a two-phase stepper motor with phase current 1.7 A and phase resistance 1.5±10% Ω. Holding torque of this motor is minimally 0.4 Nm.

There is a DM420 2H Microstep driver used for driving the 17HS motor. This driver works from 12 to 36 VDC input voltage and draws less than 2 A input current. The output current is between 0.44 to 2.83 A, which makes it a suitable one for the selected motor.

3.1.4 Camera selection

The camera requirements were primarily its ability to withstand vibration without losing contact with the computer. For the communication, a USB connector was found acceptable, with the possibility of upgrading to a screw-locked variant if needed. The selection of cameras was narrowed down to this connection option.

The customer expressed the wish that the camera would be one of the industrial solutions and suggested the FLIR company from its references, whose Chameleon3 USB product line was selected (datasheet of the product line can be found in [8]). The product line offers further options depending on the sensor, colour type and FPS.

The choice was ruled by the low need for FPS since the image would be read from a mostly static image, wherever a readout is required. The sensor needed to have colour input since the tested components light colour had to be detected. Another deciding factor was cost, which is why only a 1/3 inch sensor was chosen.

The selected model **CM3-U3-13S2C-CS** has 30 FPS, 1288×964 resolution with 3.75 µm pixel size, and a CCD type sensor.

Lenses

With the selected camera, there are lenses needed. For the right selection, the equations for optical imaging are needed. A detailed explanation can be found in [14]. The equation (3.2) was published by the same source, where β' is transverse magnification, y is the size of the object, y′ is the size of the image, f is focal length and z is object distance.

$$-\beta' = \frac{y'}{y} = \frac{f}{z} \tag{3.2}$$

The f and z needed to be calculated for the right selection of the lenses. The calculation can be started with fixed information which is the image and object size. In X-axis the object size is y = 400 mm as a minimum what the camera needs to see. The Image size in that axis is given by the size of the CCD sensor, and the size is y′ = 3.4 mm. In the Y-axis, the dimensions are y = 700 mm and y′ = 4.8 mm. Calculation of transverse magnification is done in equations (3.3) and (3.4). For the following calculation, β'_y is used since it is the one with a smaller value.

$$\beta'_x = -\frac{y'}{y} = -\frac{3.4 \cdot 10^{-3}}{400 \cdot 10^{-3}} = 8.5 \cdot 10^{-3}\,\mathrm{m} = 8.5\,\mathrm{mm} \tag{3.3}$$

$$\beta'_y = -\frac{y'}{y} = -\frac{4.8 \cdot 10^{-3}}{700 \cdot 10^{-3}} = 6.86 \cdot 10^{-3}\,\mathrm{m} = 6.86\,\mathrm{mm} \tag{3.4}$$

Calculation of focal length is dependent on object distance and vice versa. Therefore the research about possible CS lenses for 1/3 inch sensor type took place. The most common version had a stable focal length which starts at 16 mm. The problem with this solution is that the object would have to be placed more than 2.3 m from the end of the lenses, which is not acceptable for the system.

Therefore the other way around was taken, the system should be less than 1 m deep so that physical manipulation with the robot would be possible. For the calculation, the length used was z = 800 mm. Therefore the focus length is calculated in equation (3.5).

$$f = -\beta' \cdot z = 6.86 \cdot 10^{-3} \cdot 800 \cdot 10^{-3} = 5.44\,\mathrm{mm} \tag{3.5}$$

With the calculated focal length, the lenses which are on the market were rapidly reduced. The calculated value is the maximum focal length which will be able to see the monitored space so that any lower one would be satisfactory. Finally, concerning price, the **Computar T0412FICS** lenses were selected (the datasheet can be found in [3]). The lenses have a fixed focal length of 4 mm with manual focus and iris.

With this selection, the minimum distance of the object from the end of lenses could be 583 mm, and with the 33 mm lens length, the placement of the camera would be at least 613 mm from the testing area. However, at this distance, the image would have to be fixed without any possibility to move it even slightly, and the orthogonal position of the camera relative to the testing area would have to be ensured. However, it would be useful for the image recognition to be able to detect the edges of the plexiglass holding the tested components. The Y-axis object size with these edges is 885 mm, and the distance concerning lens length would have to be at least 770.5 mm, with extra space for the freedom in the placement of the camera. The resulting space between camera location and the testing space is 850 mm, which provides roughly 980 mm object space to be recorded by the camera in the Y-axis and 694 mm for X-axis, where a minimum 590 mm is needed to see the edges.

3.1.5 Sensors

The system needed to track its position. No expensive encoders were used, the position is only known relative to the reference point by counting steps - a much cheaper solution. The reference position is detected by using end switches **LJ18A3-8-Z/BX**, the datasheet can be found in [45]. These sensors are proximity (inductive) ones, but testing showed that the precision

is sufficient. In case of future improvement, precise touch switches can be added to the home position side.

3.1.6 Physical connection of parts

The selected components needed to be connected; thus, multiple connecting parts were ordered. The "X" aluminium profiles come with fastenings of several types. The angle ones were selected so that the strength of the skeleton was reinforced. For holding the fastenings, the T-slots were also needed. The profile size was 30×30 mm with upper rung 40×40 mm for motor holding.

With the skeleton assembled, the active part of the robot was built. Some parts could be fixed directly to the skeleton, but most of the parts required some intermediate element.

Most of the needed elements can be seen in the previously mentioned figure 3.1. The parts numbered 3, 4, 5 and 6 were the most significant connection parts transforming the force from one axis to another. Parts 7, 8 and 9 were mostly the underlay for the leads holders and bearings to achieve the required position. The parts 10 and 11 were motor holders, which are not visible in the figure. All parts were made from aluminium.

Models for manufacturing the pieces were designed in OpenSCAD program [23].

Axis transformation parts

The axis transformation parts were all cut from 8 mm thick aluminium sheet. The parts were used to connect the linear rod slide, and ball screw nut housing with bearings for the ball screw and the linear rod holders as well as create space for motors and sensors if needed.

For the Y-axis, there were two different sides created. In the figure 3.2, the side without space for the motor was shown, which was numbered 3 in figure 3.1. The size of the part was 220×80 mm.

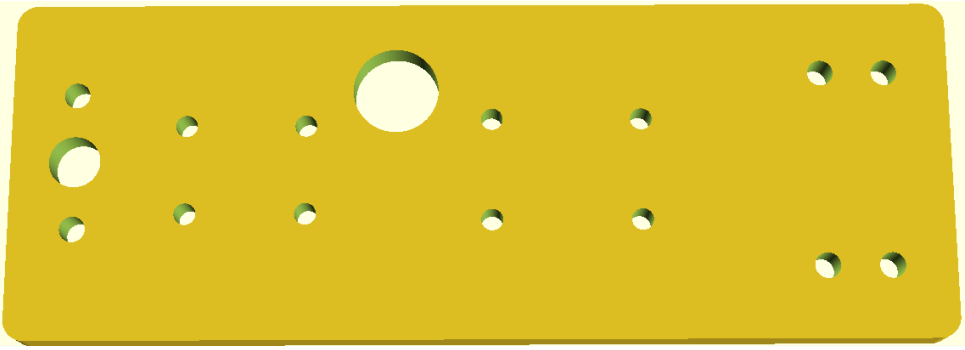


Figure 3.2: The intermediate part connecting Y-axis to X-axis.

From the left, the piece contains a hole for the linear rod holder with space for the linear rod, in case the length of the rod was longer than needed,

followed by space to connect the linear slide. The bigger opening is a spot for proximity sensor attachment. Next to it is a space for fixing the ball screw nut housing and on the right side, the bearing holder space is left.

The second part for the Y-axis, which was numbered 4 in figure 3.1, can be seen in figure 3.3. This part is similar to the previous one with two differences. The size of the part is slightly larger to be able to fit motor holder space to it. The dimensions are 220×90 mm. The motor is located on the right side, where there is space for fixing the motor and shaft.

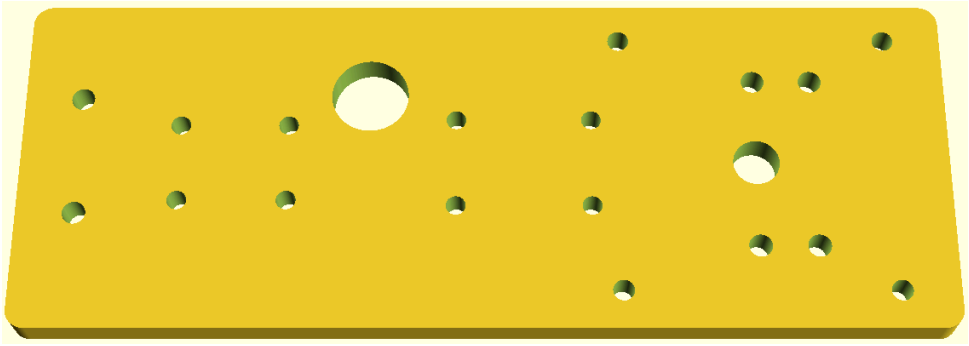


Figure 3.3: The intermediate part connecting Y-axis to X-axis on the motor side.

The figure 3.4 shows the piece with non-standard size 364×103 mm. The size had to be minimalized so that the X-axis would not lose any more effective travel space. This part was used to transfer the motion from X-axis to Z-axis and has number 5 in the figure 3.1.

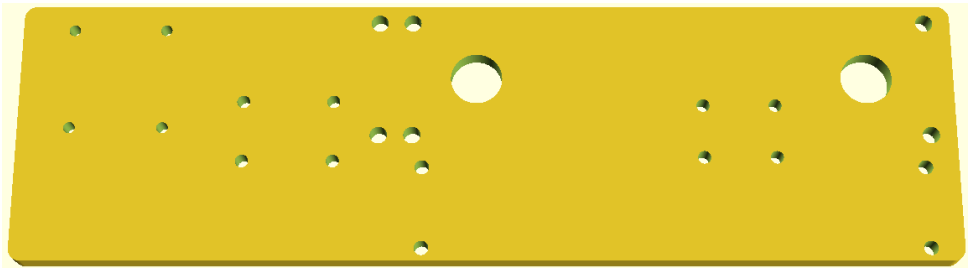


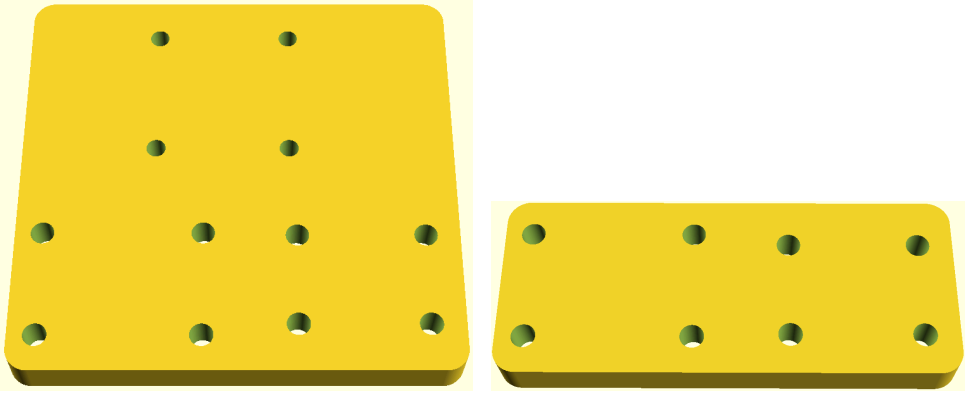
Figure 3.4: The intermediate part connecting X-axis to Z-axis.

The model contains a place to fix a motor holder on the left side, followed by ball screw housing securing space. The bearing and linear rod holders were placed next to it. The opposite end was on the right side of the piece. The sensor space was placed at each inside end of the ball screw area. In between the linear rod, the slider was attached.

The remaining transformation was Z-axis to Phi-axis. This part can be found under number 6 in the figure 3.1, and it consists of two parts. These parts can be found in 3.5.

First part 3.5a creates the main plate with dimension 97×93.5 mm. Holes at the bottom were prepared for connection of ball screw and linear leads. The upper part then supports motor holder screws. The second underlay

part 3.5b is just a way to lift the motor above its bearings so that it can be pushed in front of the Z-axis assembly.



(a) : Main part. **(b) :** Underlay part.

Figure 3.5: The intermediate part connecting Z-axis to Fi-axis.

Motor holders

Other parts cut from the 8 mm thick aluminium material were holders for the motors. There were two types of them, both containing a big centre hole for placing the motor with four holes to hold it. These pieces were different in the way they connect to the system.

Motor holder for the Y-axis can be seen in the figure 3.6. This piece was placed on the position marked as 10 shown in the figure 3.1. The motor was fixed to the profile from above, and a hole was drilled into it. That created a space for the shaft and the flexible coupling. The fixation of the motor was impossible in the perpendicular position because the screws holding the motor would have to go through the profile. Therefore, the holder was rotated by 90 degrees to move the outside screws far enough from the profile and the one in line with the gripping holes would fit to the profile groove. This piece was manufactured two times to hold both Y-axis motors.

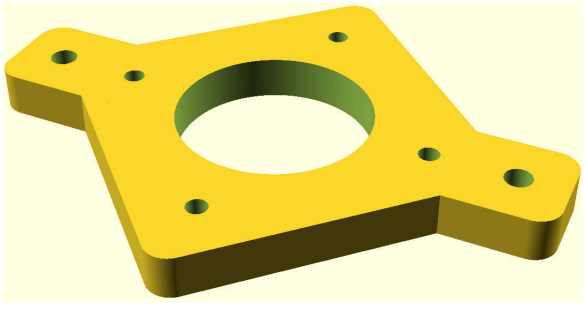


Figure 3.6: Motor holder for Y-axis.

The second type of motor holder was for the X-axis. The position was mentioned as number 11 in the figure 3.1, and its model can be seen in 3.7.

This part was designed to use spacers to make space for the flexible coupling and connecting the motor shaft to ball screw the similar way as was used profile in the first version.

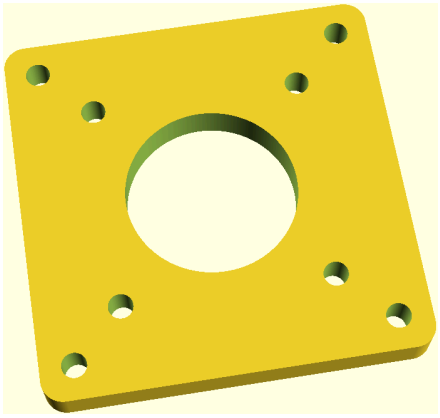


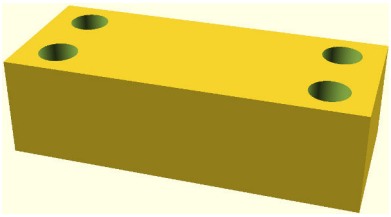
Figure 3.7: Motor holder for motion translation from the Y to X axis.

Underlay for bearings and linear lead holders

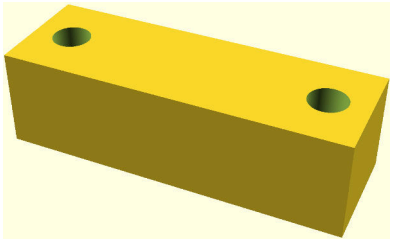
It was essential to ensure the guide was aligned to the right plane, for the movement to work. Therefore, the underlay was needed in more than one case. It was made from leftover pieces of aluminium located in the laboratory. The pieces were just rectangular objects with the necessary size and holes to tighten the screw to attach bearings or linear rod holders.

To lift the bearings, two types of underlay were needed. A high precision one was needed to ensure trouble-free movement. The height of the underlays was 17.4 mm. The models can be found in 3.8. The place to use this was marked 9 in the figure 3.1.

Two types were necessary to support both sides of the ball screw. The ending side uses a smaller bearing with only two screws, so the 3.8b underlay was used. The opposite side needed to support a bigger bearing holder with pass-through bearing to push through the ball screw and attach it to the flexible coupling. The model for that underlay can be seen in the figure 3.8a.



(a) : Motor side.



(b) : Ending side.

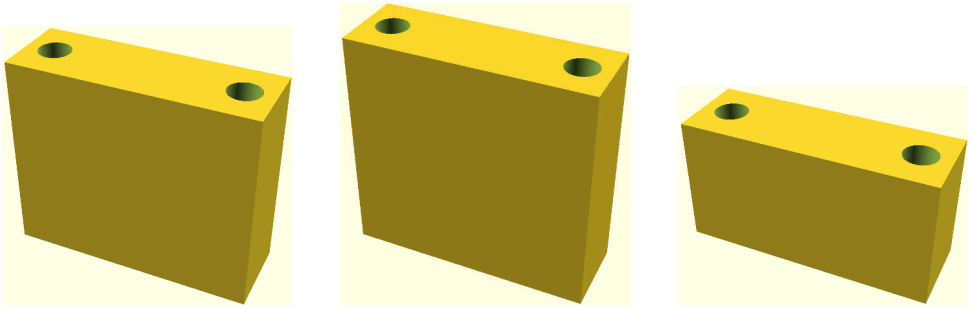
Figure 3.8: Underlay for lifting bearings to the motor axis.

For the linear rod holders, more underlays were needed to support the parallel running of the linear rods next to the ball screw in the Y-axis. All

types of linear rod underlays are shown in 3.9. There were three types needed, and every one of them was used in two instances.

In figure 3.1, the violet numbers correspond to these models. The model for number 7 can be found in the figure 3.9b with a height of 40 mm. The number 8 is in 3.9a with a height of 35 mm and number 9 in 3.9c with a height of 21.4 mm.

The height difference in the upper and bottom side of the Y-axis is due to the different profile size. The bottom profile was 30×30 mm and the upper one 40×40 mm to be able to support motors.



(a) : Y-axis upper side. **(b) :** Y-axis bottom side. **(c) :** Z-axis.

Figure 3.9: Underlay for lifting linear rod holders to the motor axis.

3.1.7 3D printed parts

The switching tool of the robot was not cut from aluminium. The same applies to several other parts added to the design later, after the manufacturing order for the aluminium was placed. In case of future reconstruction, most of these can be cut from aluminium as well, instead of being 3D printed. However, the print was tested strong enough to use.

The only part too complicated to be cut from plate metal was the tool for switching. With dimensions 36×19.4×8 mm and holes for the motor shaft and screws with nuts to hold it, the tool was easier to 3D print. The model can be seen in figure 3.10. The size of the tool was designed so that it is possible to push even the flush buttons. The cut-out in the front is slightly wider than the switch thumb-grip to easily slide it on the switch even if the position is a few steps off.

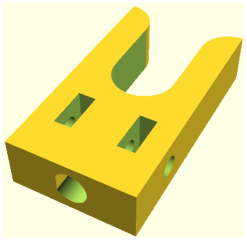


Figure 3.10: Switching tool.

The following parts were designed later than the order of the aluminium cut out was done, and therefore the 3D print was used. These parts did not need to be so tough as the parts in previous chapters, so their use was not a problem.

The cut parts support proximity sensors for the X and Z axes; however, there was not enough space for them in the Y-axis. So, unique upper and bottom holders were designed.

The upper one is attached using angle connectors designed for joining profiles so that the sensor would be perpendicular to the system. The model with dimensions 22×102×40 mm can be seen in the figure 3.11.

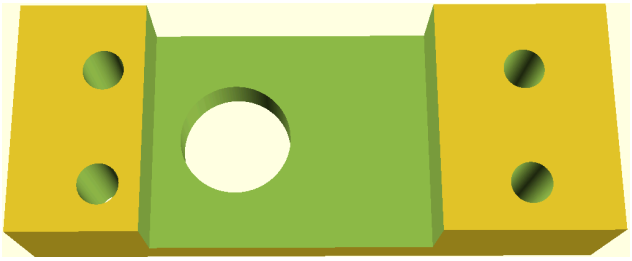


Figure 3.11: Holder for the proximity sensor in the Y-axis, upper side.

The bottom sensor holder was designed to be inserted into the groove of the profile. There are also holes for attaching cables that run around it. Figure 3.12 shows the designed model with the outer dimensions 100×100×8 mm.

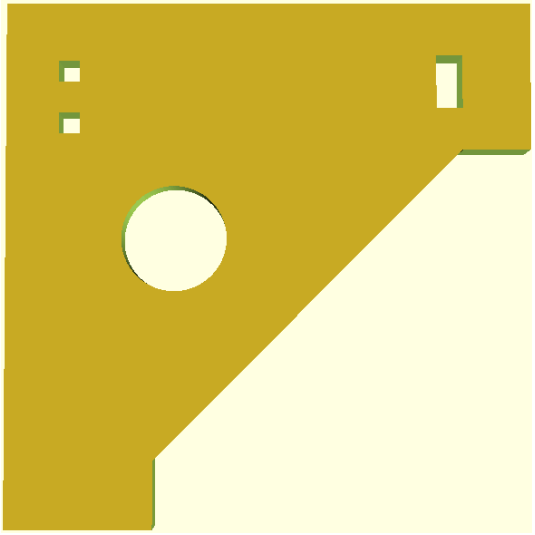


Figure 3.12: Holder for proximity sensor in Y-axis bottom side.

The last printed part was needed to secure the camera on its mounting point on the profile cross. The holder size is 25×60×9 mm so it could hold the camera and add one hole to the securing screw. Figure 3.13 shows the model. The thickness of the holder is due to the need of providing a flat surface. Whereas the screw heads holding the camera are sunken inside the holder.

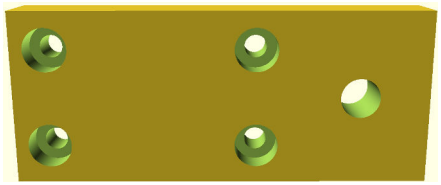


Figure 3.13: Holder for camera

3.1.8 Working area

The working area is the space where the tested components are placed. The components are of M22 size with a security protrusion which prevents them from rotating in place. The active space is 400×700 mm large, and the components need to fully fit inside this area, including their edges. Especially the four and the two-way buttons need more free space around them. With this in mind, an 8×10 matrix of mounting holes was designed. The visualization can be found in figure 3.14. The outer board of the model is 530×840 mm. The mounting plate was cut from plexiglass and secured to the front of the robot.

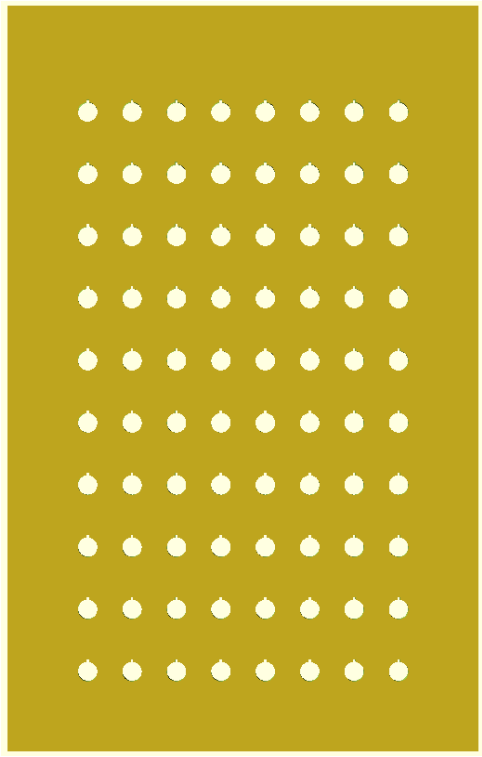


Figure 3.14: Working area layout.

3.2 Electric part connection

The system consists of several electronic subsystems. Their interconnection is shown in figure 3.15.

The green parts of the block scheme are motion ones. They consist of motors and sensors as well as its power supply. The yellow parts show a control board described in detail in chapter 3.2.1. These parts are separated by optocouplers on both sides. On the sensor side, the optocoupler is placed on the PCB. For the motors, the optocoupler is part of the motor drivers.

The blue part consists of the main computer and a connected camera. This part communicates with the control board via USB and provides power for it.

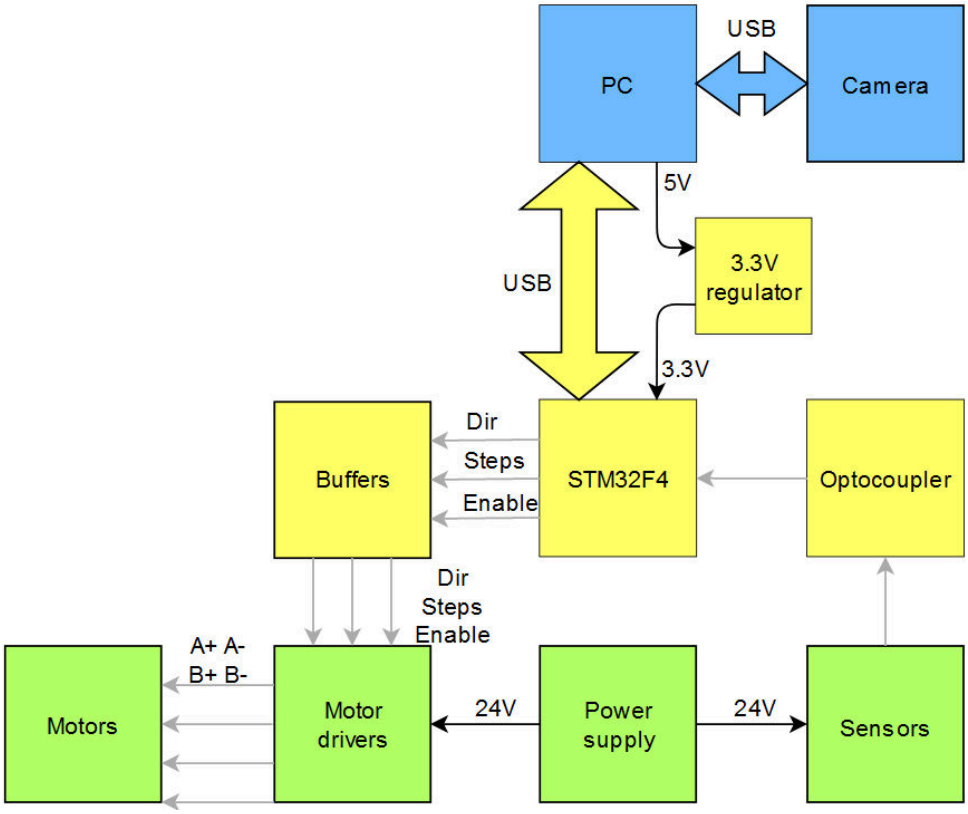


Figure 3.15: The block scheme of electrical parts connection.

3.2.1 Control board

This section consists of the circuit schematic and printed circuit board design. The control board is used for timing the steps for motor drivers and reading the sensor inputs. The computer with an operating system was not suitable to do the timing since the system has more priority tasks which could be problematic for the smoothness of the robot movement.

To assure that the movement is not interrupted, the Real-Time Operating System can be used. One of the standard micro-controller series supporting

such system is the STM32. The STM32F4 family was selected from the series. A printed circuit board had to be designed so that the micro-controller could be used.

The circuit visualization, circuit schematic and gerber files can be found in appendix A. KiCad tool was used for the design. Information about it can be found in [19].

Circuit scheme

The circuit is built around the STM32F401RCT6 micro-controller. Its datasheet can be found at [41] and reference manual at [40]. The schematic was designed using reference manual and hardware development application note which can be found at [39].

The power is drawn from a USB connection to a computer, which is also used for communication. Figure 3.16 shows the protection of this communication. The 5 V power is protected by a resettable fuse. The data transfer is then protected by STM USBLC6-2SC6 (datasheet at [37]), a very low capacitance ESD protection.

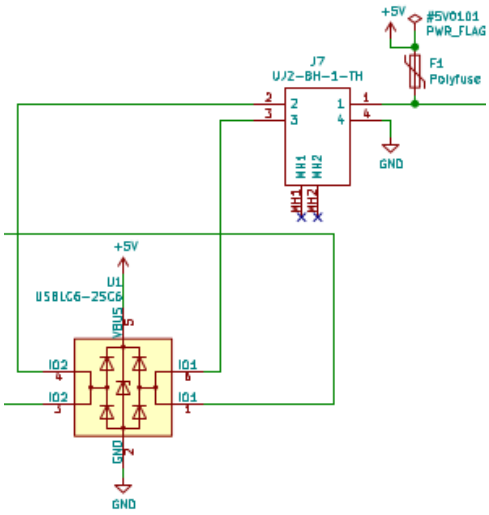


Figure 3.16: USB connection and protection.

In the figure 3.17, the linear regulator to 3.3 V connection is shown. The linear regulator LF33CDT-TRY is connected with filtering capacitors according to the datasheet, which can be found at [38].

For the USB communication to work, a crystal is mandatory. The internal timer of the STM would not be precise enough to be able to establish the connection. 8 MHz crystal ATS08ASM-1 with load capacitance 20 pF was selected, the datasheet can be found at [5]. In the figure 3.18, the connection of the crystal is shown as well as the connection of the reset button.

The outputs are addressed by barrier terminal blocks. For the X, Y and Z axis, there are six pins connected, for the motor Phi there are only four pins. Table 3.1 shows the pin output.

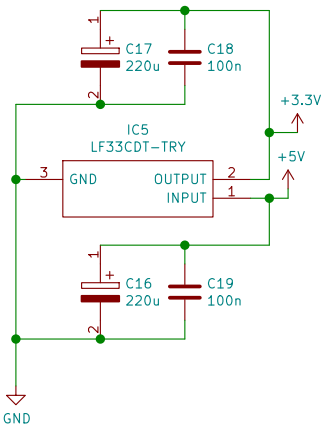


Figure 3.17: Positive voltage regulator connection.

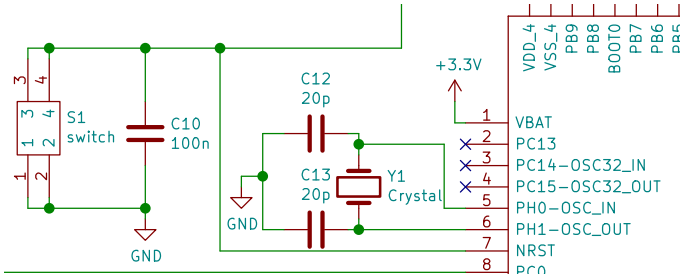


Figure 3.18: Crystal and reset button connection to STM32.

These signals are wired from STM through 3-state quad buffer (datasheet can be found at [27]. The motor driver communicates on 5 V, and the STM operates on 3.3 V. The buffer is necessary to make the signal 5 V at logical 1.

The motor driver logical signal current is 16 mA maximum. The driver also contains 270 Ω resistor which would result in 18.5 mA current in case of the 5 V input. Therefore the 100 Ω resistor has been added between buffer and output resulting in 13.5 mA maximum logic signal current.

There are also inputs from the inductive sensors. Table 3.2 shows all the input-outputs for one axis. The part is separated by optocoupler TLP291-4, whose datasheet can be found at [44]. The separation is needed because the part is powered by 24 V. The power is drawn from the barrier terminal blocks for sensors, and the signal from the sensor is input the same way. There is one block for each axis; the pin output is shown in table 3.2. Both end sensors for one axis are connected to the same terminal block - the side is determined by software. The sensor output is "normally open", and in case of the proximity of the robot, the signal is grounded on the sensor side. The optocoupler then stops powering up the NPN transistor on the other side, which is normally grounding the signal. The program then receives a rising edge interrupt due to the internal pull-up.

Pin	1	2	3	4	5	6
Motors X, Y and Z						
Signal	PUL +	PUL -	DIR +	DIR -	ENA +	ENA -
Motor Phi						
Signal	PUL	DIR	5 V	ENA	-	-

Table 3.1: The signal to pin correspondence for X, Y and Z axis.

Pin	1	2	3	4	5	6
Signal	24 V	SIG_A	GND	GND	SIG_B	GND

Table 3.2: The sensor connection header.

Printed circuit board

The gerber files for printed circuit board can be seen in the appendix A in the figures A.3, A.4, A.5, A.6 and A.7. The assembled PCB is shown in the figure 3.19. The legend for the numbers can be found in table 3.3.

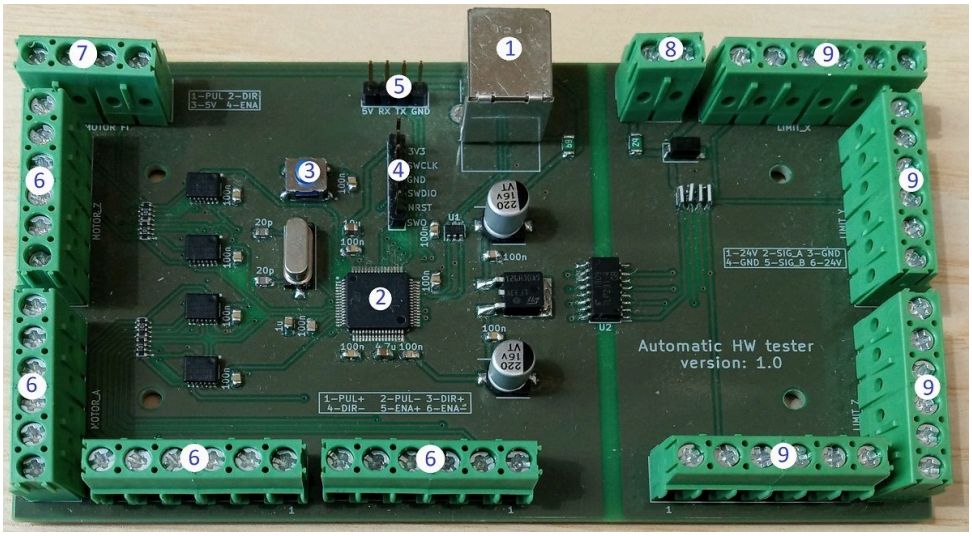


Figure 3.19: Assembled PCB.

The programming and debug pin output is prepared for the ST-Link connection. The pin output for UART was planned as the backup option for communication with PC in case of problems with USB communication. Since no problem was encountered, the UART communication is not implemented in the current version.

Things to change in a future version

After assembly and implementation, few possible improvements come to light.

Separating enable signals. The most pressing one is the enable signal separation. As for now, the enable is joined for all the motors. However, the

Number	Factuality
1	USB A connector
2	STM32F401
3	Reset button
4	Programming and debug pin output
5	UART pin output
6	Outputs headers for X, Y and Z axis motors
7	Output header for Phi motor
8	24 V and GND input for sensors
9	Headers for connecting the sensors

Table 3.3: Legend for figure 3.19

drivers are different for X, Y and Z axis from the Phi axis. It turned out that the Phi axis needs logical 1 for enable and the other ones need logical 0. It is currently solved by disconnecting enable from the drivers since logical 0 and "not connected" has the same effect. However, the ability to turn off the motor from the software, in case of not using it, could be beneficial, as well as turning on or off the motors separately.

Pull-down resistors for outputs. Currently, the pull-downs are enabled via software after micro-controller boots. A problem occurs in case of micro-controller restart. While the system is booting up, the idle state of the pins is logical 1 instead of expected logical 0. It could cause sending a few spurious steps to motors. It is not a real problem since the look for the home position is the first thing the program does after booting up, so the position is not affected. The movement would not be that obvious either, since, in the 16 micro-step settings, which is currently used, one pulse corresponds to 2.5 µm movement.

Change the resistor arrays for separate resistors. After PCB production was commissioned, there was a problem with the order of parts and the resistor arrays could not be obtained. The array has a specific footprint, and it was not easy to populate other resistors on its place. For future changes, the commonly used resistors would be a better choice.

Replace incorrect footprints. The footprints for the buffers are smaller than they should be. It was necessary to bend the pins to be able to place them there. Another problematic footprint is for the four-pin barrier terminal. The footprint through holes for the pins are smaller than they should be. For the assembly, it was necessary to grind the pins of the terminal a little.

Flip the order of the pinout for the motors. The pinout for the motor driver is the same as the output from the board. It means that the connection is crosswise.

Connection of barrier terminals. There is too much unnecessary space around the block terminals which results in bigger PCB. The cost of the board is based on its area so it could be beneficial to shrink it.

3.3 System construction

The system was assembled from the parts mentioned in chapter 3.1 CNC design. All parts cut from an aluminium can be seen in the figure 3.20. The part numbered as 1 is the connection from X to the Z-axis, number 2 is the connection from Y to X-axis with a motor on this side and number 3 is the side without the motor. The 4 is the transformation from Z to Phi axis with 5 for lifting it. Number 6 is the motor holder for X-axis and numbers 7 for Y-axis. The underlays for linear rods are numbered 8, 9 and 10 in different heights and number 11 is the underlay for the bearings in Z-axis.

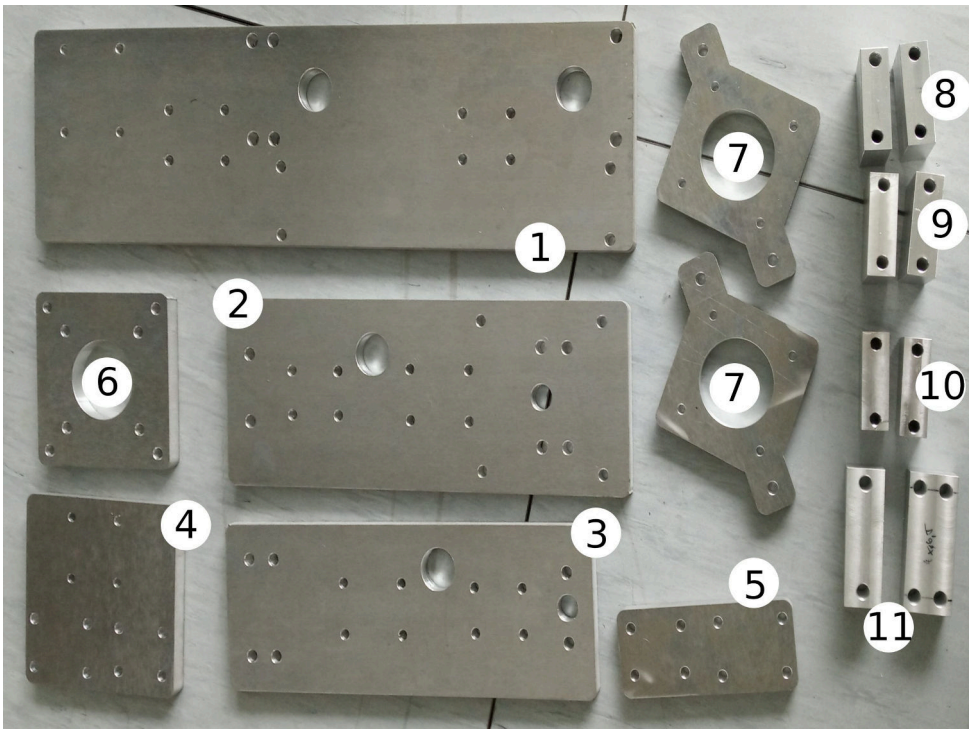
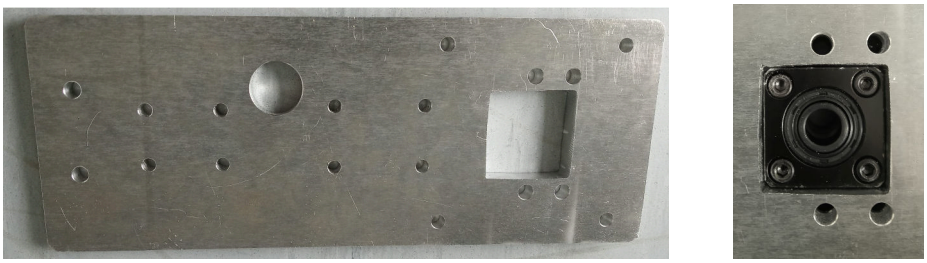


Figure 3.20: The aluminium cut out parts.

Unfortunately, the model for the number 2 part was not correct. The part design was assuming that the ball screw shaft would be pushed through, but not with the bearing protrusion. Therefore an additional cut out has to be made. The cut can be seen in the figure 3.21. This change must be reflected in the model in case of another robot build.

The connection of the robot carriage can be seen in the figure 3.22. In this figure, parts 1, 2, 3, 4, 5, 10 and 11 can be seen in their right place. However, the connection in this state was relatively wobbly, and the second linear seat had to be added next to the first one. It means that the model



(a) : The cut-out part. **(b) :** Fitting the bearing.

Figure 3.21: The cut-out space for bearing.

for part number 1 needs to be fixed as well. Four additional holes for the fixation of the seat are mandatory. The side of the seat is smaller than half of the space between the seat screw-threads. That means that the holes can be added to both sides without the need to move the existing ones.

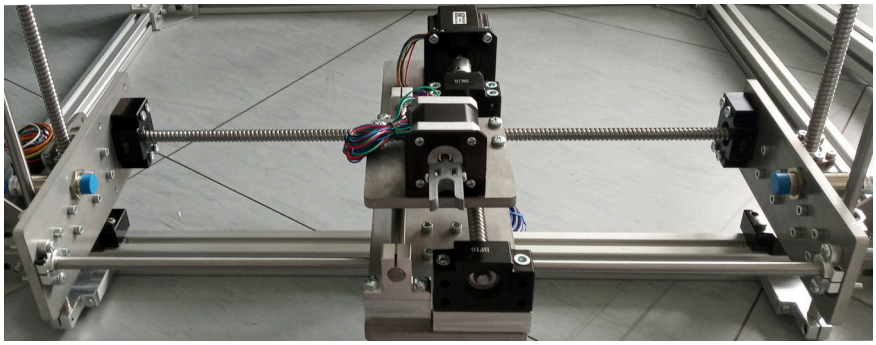


Figure 3.22: The assembled carriage of the robot.

On the left side of the figure, there is a place for the motor connection. For this, part number 6 with the spacers is used. The detail of the connection can be seen in figure 3.23. The spacers are used so that the connection of the motor shaft with the ball screw end with the flexible coupling is possible.

(a) : Side view. **(b) :** View from above.

Figure 3.23: Connection of the motor for X axis.

For the Y-axis motor connection, the number 7 part was used. The

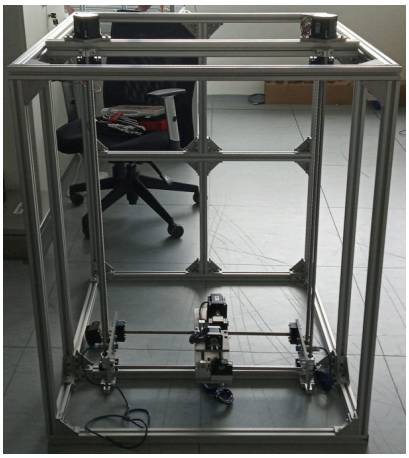
connection can be seen in figure 3.24. The profile had to be drilled to make space for shaft and ball screw connection with the flexible coupling. The drilled space has 21 mm in diameter, which would be difficult to fit to the 30×30 mm profile without the loss of the strength of construction. That is why the 40×40 mm profile was used as the upper rung.

(a) : Side view.

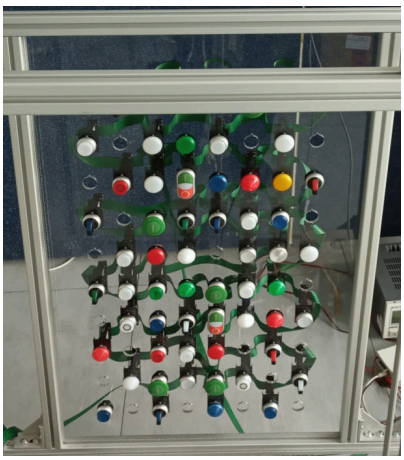
(b) : Stretching the coupling.

Figure 3.24: Connection of the motor for Y axis.

The figure 3.25a shows the skeleton and the connection of all axes. There were no working area and control electronics yet. However, the figure shows the layout of the robot. The figure 3.25b then shows the working area attachment in the front of the robot. The area is already filled with the control elements for the testing.



(a) : The construction of the robot without a working area.

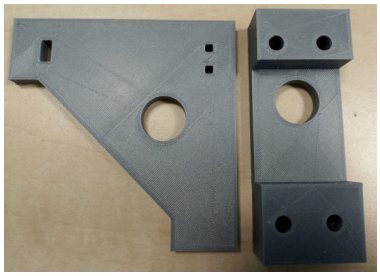


(b) : Working area attachment in the front of the robot.

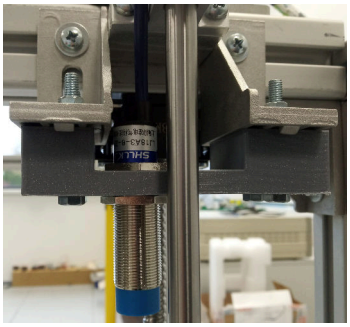
Figure 3.25: Construction of the robot.

The next part of the assembly was sensors. The ones in the X and Z axis already had a designed space in the aluminium cut-outs. However, the Y-axis

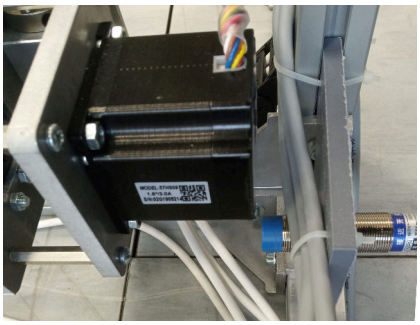
sensors did not. Figure 3.26 shows the placement. Figure 3.26a shows the 3D printed parts designed as holders for the Y-axis motors. The 3.26b shows the way the upper sensor was placed, and the 3.26c figure shows the lower sensor placement. For the lower one, the cables for the motors had to be sorted so that they were not in a signal way. The sensor was placed so that the motor for X-axis would be detected if the robot was low enough.



(a) : 3D print of the sensor holders.



(b) : Upper sensor.



(c) : Lower sensor.

Figure 3.26: Y axis sensor placement.

Another part is the mount of the camera, which is shown in figure 3.27. Its placement is in the back cross of the robot. The camera can be moved up and sideways by sliding the profile, which is vital so that the camera can be secured at the centre level of the working area to assure the visibility of the entire thing.

Figure 3.27: Connection of the camera.

The placement of control electronics is in the left-back bottom. The parts are secured on the 8 mm thick plexiglass which is inserted to the aluminium profile groove. Another profile had to be added to support the plexiglass from the third side. The remaining side was secured by T-slots with washers and screws to disallow movement.

In figure 3.28, the layout and the wire connection of the control electronics are shown. Number 1 is the 24 V power supply for powering up the motors, its drivers and the proximity sensors. The power cord for the supply had to be pinned down to the plexiglass so that there was no chance to rip it out by accident.

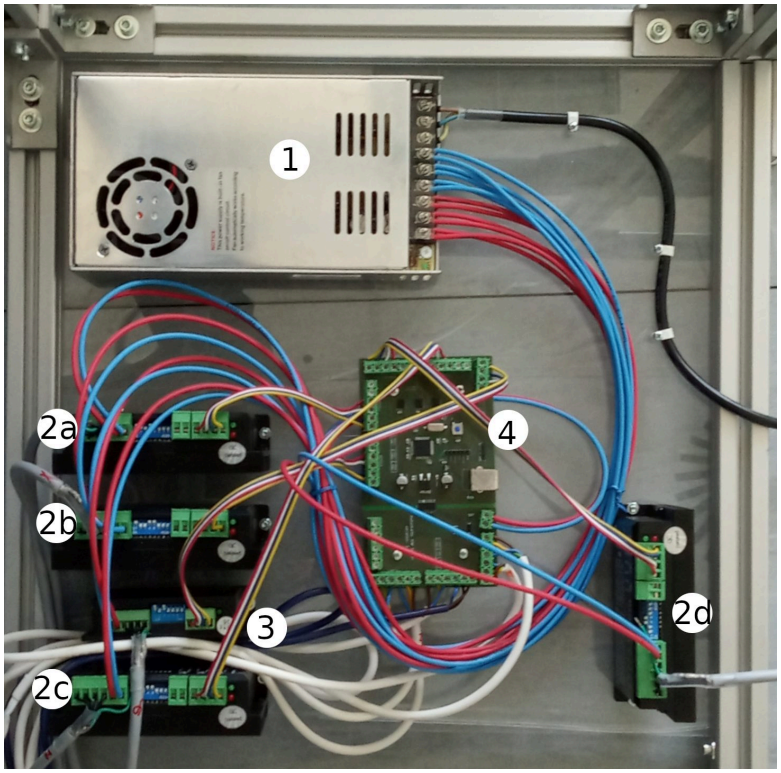
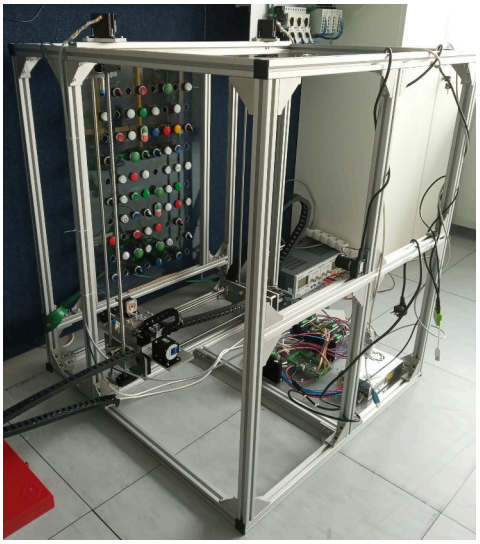


Figure 3.28: The connection of power supply, motor drivers and control board.

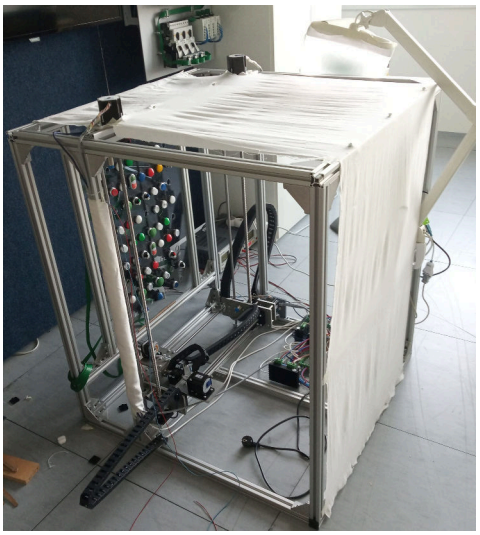
Number 2 are the motor drivers for the linear movement axes. The 2a and 2d are the Y-axis drivers, the 2b is X-axis driver, and the 2c is the Z-axis driver. The number 3 is the rotation axis driver. The last part, numbered as 4, is the control board for the pulse generation.

The entire construction with the described parts is shown in the figure 3.29a. However, image recognition was problematic with the intensity of the light, and it's glare from above. The shade and the additional light from sides had to be added. The figure 3.29b shows the temporary solution which is currently used.

There was also a problem with the flexible coupling. The robot was vibrating too much, and the coupling security screws were falling out repeatedly. In most connections, the coupling was able to hold the connection without it,



(a) : Construction of the robot without the additional light.



(b) : The construction with added lights and shade.

Figure 3.29: The final construction of the robot.

but in the Y-axis, it sometimes caused a release of the connection, and one side of the robot was not moving. That is why the thread-locker glue was used on the security screws.

3.4 Cost calculation

The hardware parts can be split into several sections. Table 3.4 shows the subsection cost and total cost of the robot. In appendix B, the sections are written down in tables B.1, B.2, B.3, B.4 and B.5 in more detail.

Project section	Cost [CZK]	Approximate cost [€]
Camera and accessories	7 297	278
Motors and drivers	11 045	420
Motor connection	396	16
Motor accessories	2 302	88
Aluminium and X profiles	4 360	166
Fastenings	6 588	251
Movement leads	4 803	183
Energetic chains	864	33
Custom made parts	11 535	439
PCB components	677	26
Total cost	**49 867**	**1 900**

Table 3.4: Cost calculation

The aluminium X profiles, described in B.3, were bought as 2 m long and then cut in the laboratory; therefore, the price could not be determined for separate cuts. In the same table, the movement lead part is described too. The ball screw set contains a ball screw, ball nut, the housing for the nut and end bearings. Also, the fastening part contains the "Others" row, which covers the miscellaneous items brought gradually, like nuts, washers and nuts. The cost is approximate. In the beginning, the goal was to fit the cost under 60 thousand CZK, which was accomplished.

Chapter 4

Programming

This chapter describes multiple programming challenges. The first part which needed to be programmed was the control board for motor movement. Then there was the communication protocol with the PC program and the image recognition algorithms for detecting the state of the tested switches and lights. There was also a user interface to be done, using a terminal for the most part, but with an option to enable a graphical interface so that system configuration could be done correctly.

4.1 STM firmware

This chapter describes code implementation for the motor control board. The board can find the robot's home position, calculate its actual position and move it to the desired position using a dynamic speed curve. The board processes an incoming position request, calculates the number of steps in each axis and the speed curves to get there as smoothly as possible. For STM firmware implementation, the datasheet [41] and reference manual [40] was used.

4.1.1 Code initialization

The STM32CubeMx configuration tool was used to generate the firmware project's skeleton (for more information see [35]).The Cube program is a tool for configuring STM32 microcontroller family. The HAL libraries were chosen over the Low-Level library alternative.

Within the Cube tool, two middlewares were chosen, the FreeRTOS and the USB device. The USB is used in the Device on the Go mode.

Another part was configuring the pinout. There are multiple settings for different output pins. The portion of motor control pins (direction and enable) is set as push-pull outputs with internal pull-downs. The sensor inputs, named as LIMIT, use an external interrupt mode with rising edge detection, with no internal pull-up or pull-down. Timer outputs produce PWM signals for driving the motor step signal, with internal pull-down resistors.

The RCC pins serve as quartz crystal clock input. There are also system pins for the Single Wire Debug port to be able to upload the program and

run the debug mode.

Numerous internal components were configured, which are not linked to any pin. One of them is a watchdog for the program to detect program freeze and restart it in that case.

Finally, many interrupts were enabled. Some interrupts are used for sensors inputs, some for the counters in timers and some for the USB communication.

All screenshots from the Cube tool can be found in appendix C. The pin configuration can be seen in figure C.1, and the clock settings can be found in figure C.2.

Timer chaining

There could be problems using interrupts to count the steps sent to the motor since two different axes need to run simultaneously, along with the communication thread and interrupts from the sensors. That is why chained timers were used.

For each axis to run, two timers are used. One timer generates the PWM output for the motor driver, and the second one is set as a step counter affected by the first one. The interrupt is generated from the second timer after the required number of steps are performed. More information about the chained timers is provided in the Motor control portion of chapter 4.1.2.

4.1.2 Program parts

The integrated development environment used to program the STM was Atollic TrueSTUDIO (more information can be found at [36]).

The firmware runs in several threads. One thread updates the watchdog counter, another handles the USB communication and the last one is used for the motor control.

The USB thread communicates with the motor control one by using a global position and sensor indication structures. Both structures are put in a C module and are not visible to others. The update and reading are done via publicly visible getters and setters wrapped in mutexes.

The USB communication part consists mainly of the parser for incoming messages and sending responses. The communication protocol can be found in chapter 4.2.

Another big part of the program is the interrupt handlers. There are external interrupts for the proximity sensors, which detect the end of the movement space and stop the motor movements and update the position accordingly.

The other type of interrupt handlers is the counter ones. The timer, set as the counter for the PWM pulses, will generate an interrupt when the steps are done.

Motor control

This thread controls the movement of all four axes. The first thing the system does after power-on is looking for the home position. At low speed, it will first find the zero position of Z-axis to avoid the collision with any components placed in the test area. Then it runs the X and Y axes simultaneously. The movement's speed is relatively slow since the robot does not have any idea how far it is from the end of the axis. It also needs to stop at the exact position where the proximity sensor triggers the interrupt, without any oversteps.

After the homing, the position is tracked in the number of steps from the zero position. If there is a request for a movement, the thread looks at the steps and sets the timers accordingly. It is possible to run the X and Y axes simultaneously; the other axes need to be moved sequentially.

Configuring the axis movement consists of a few sub-parts too. The main idea is shown in figure 4.1. The system needs to configure the counter timer so that the correct input is used. That means setting the slave mode to the appropriate ITR input source. It then sets the number of steps, to be done into the counter period. If the number of steps is higher than the 16-bit value, which is a maximum that can be set to TIM4 and TIM9, the system splits it to several parts. During the next interrupt, instead of stopping the motors, it sets the remaining portion of the steps.

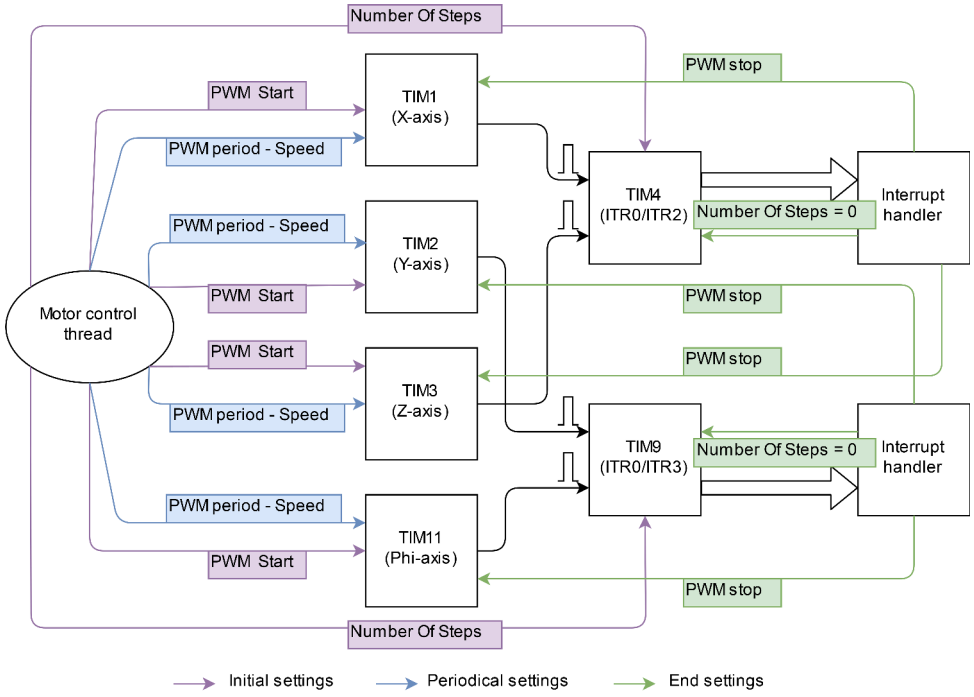


Figure 4.1: Motor control - timers settings diagram.

The second part is to configure the starting speed of the PWM output to the corresponding timer. This output is linked to the pulse pin for the motor drivers as well as to the counter timer. After that, the PWM peripheral is started. The thread will then periodically recalculate the speed (period of

the PWM) so that the speed curve is relatively smooth and the robot can move off and stop without any missed or extraneous steps.

The speed is calculated as frequency and then converted to period as inverse value. The movement starts as ten slow steps which are done at a constant speed, and then it performs a linear speedup for next 400×micro-steps, which correspond to two turns of the motor, at which point the maximum speed is reached. The slowdown starts when two turns of the motor are remaining, and uses the same linear function in reverse, to slow down the movement.

In the current implementation and setting at eight micro-steps for the linear motion axes, the maximum PWM period used is 1024/micro-steps. With the timer settings, it approximately corresponds to the frequency of 977 Hz. The minimum period is set to 10, which approximately corresponds to the frequency of 12.5 kHz.

The above means that the linear motion speed varies from about 2.44 $\mathrm{mm \cdot s^{-1}}$ to 31.25 $\mathrm{mm \cdot s^{-1}}$. Higher speeds were tested, but the motion was losing some steps then. Possibly, a slower ramp-up in the movement speed could achieve better results. For now, the system can move from the home position to the farthest position in about 22 s.

Figure 4.2 shows the timer's settings for 20 000 steps (which corresponds to the adjacent component's position in the X-axis). The orange curve shows the period set to the timer register, that means it is relative to the timer settings. The blue curve shows the actual step frequency. For the motor rotation speed, the number needs to be divided by 200×micro-steps, which, in this case, is 1600 steps. The period value can only be an integer, which results in a somewhat jagged curve. It is more visible in the frequency curve than the timer period curve.

4.2 Communication protocol for the control board

The USB communication between the control board and the main computer appears as a virtual COM port. After establishing the communication, several types of commands can be sent to the control board.

4.2.1 Messages for the control board

Position message. The position message format is **"<x>,<y>,<z>,<phi>\r"**. In the place of the <>, the coordinate in a number of steps is set. For example: "40185,42217,13000,200\r" corresponds to the component position 2,8,1,0. The computer part of the program calculates the transformation from component coordinates to the robot (step) coordinates.

"POSITION\r". This message retrieves the current robot position.

"HOME\r". The request to move to the home 0,0,0,0 position and adjust it according to the sensor signal.

"<axis>+\r", "<axis>-\r" for example "X+\r". This type of message is used in case of a manual position adjustment. It is only sent from the GUI

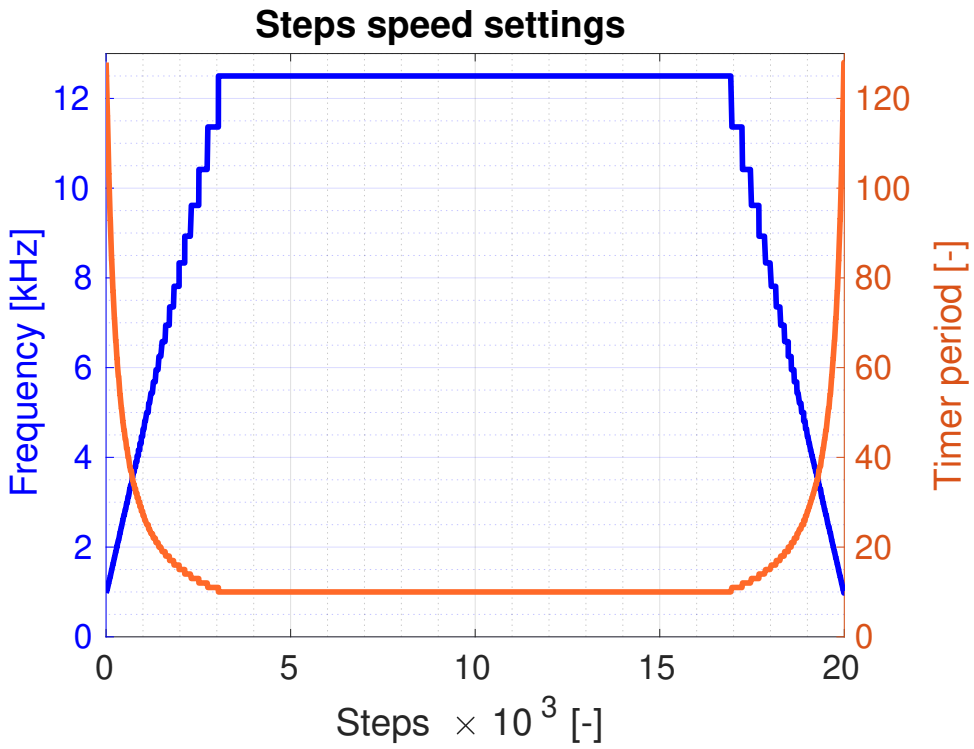


Figure 4.2: Motor settings.

interface when the user uses the arrow buttons to adjust the position of the first component placement.

"STOP\r". This message is sent after the user releases the button.

4.2.2 Responses from the control board

Position message. For the format of this message, see the chapter 4.2.1. This type of message is sent every time the movement is finished, or the sensor indicated the moving area's end. The message contains the current position of the robot. This message is also sent as a response to the "POSITION\r" and "STOP\r" messages.

"Motors are busy\r". If the robot is in the process of moving, any incoming messages result in this response. The new movement can only start if the previous one is finished. The current position is not valid as long as, the previous movement is not finished.

"Position is out of boundaries\r". If the received request for the message is out of the robot boundaries, this is the response the control board will send.

"The direction is not possible\r" is a response to the position adjustment messages if the user tries to move to the direction, which is already on the border.

"Incorrect message\r". This response is sent in case any other message is received.

4.3 Computer part of the program

The computer part of the program was created for the control of the whole system as well as communication with the user. This part of the program was written in Python 3.8. [31]. The PyCharm was used as an integrated development environment (more information at [16]).

The computer takes care of the communication with the control board. It sends, receives and parses the messages. It translates the position from the user input (component) coordinates to the motor steps and also checks if the connection with the control board is still running. The disconnection detection is useful, especially in the case of the system waiting for a response; it could freeze without it.

Another part of the program takes care of the communication with the camera as well as the recognition of the component state from the image. The image recognition is discussed in chapter 4.4.

There is also part of the program reading commands from the terminal or GUI's terminal component, evaluating them and executing them.

4.3.1 Initialization

There are a few things which need to be done before the system can run any commands.

Some system constants can be changed only in the code and are defined by the physical build of the robot. However, there are also a few of them, which can be changed. In the working area replacement, the "Settings.json" file needs to be updated since the information about the physical size is placed there.

There is also a "Steps.json" file. This file stores the last adjusted position of the first component, in the number of steps, from the robot's home position. In case of a user moving the working area, this needs to be updated as well. The update is done by the graphical user interface, which is discussed in chapter 4.3.2.

The components' current placement can be entered into the "configure.txt" file or in any file passed in the program parameter. More information about the parameters can be found in 4.3.1. The program will parse the input file and create a matrix containing the information about the components so that the program knows how many steps it has to do to push the buttons or the switch states' angles.

Configuration file

The configuration file format uses multiple lines. Each component is inserted as one line formatted as **<x>,<y>,<component code>,<light colour>**.

The <x>,<y> are coordinates of the component. It is a number from 1 to max value in the axis. The <component code> is the manufacturer code

described in chapter 2.1, including its plastic cup colour.

The <light colour> is an optional parameter, which can be omitted. It contains information about the type of light inserted behind the component.

In case of a vacant position, the line does not have to be inserted at all, or the <component code> can be "None".

The line can, for example, look like

```
2,1,M22-WRLK3-R,R\r\n
2,1,M22-WRLK3-R,None\r\n
2,1,M22-WRLK3-R\r\n
```

The empty command can look like

```
3,1,None\r\n
```

Or the line does not have to be included at all.

Settings file

The settings of the working area can be changed in the Settings.json file. The file contains information about the number of rows and columns where the components can be inserted as well as its dimensions in mm.

"matrixX" and **"matrixY"** values are the number of positions in the working area.

"sideX" and **"sideY"** are the outer dimensions of the plexiglass in mm.

"spaceX" and **"spaceY"** are the dimensions between components in the X-axis and the Y-axis in mm.

"firstX" and **"firstY"** are the left bottom component's positions from the plexiglass boundaries in mm.

Program parameters

The program can be started with the following parameters.

-g to open the GUI version.

-f <file path> with this parameter, the system loads a file from the given path and executes it as if typed in by the user.

-c <file path> with this parameter, the system loads the configuration file from the given path instead of reading from the default "configure.txt".

4.3.2 User Interface

In the default mode, the user communicates with the program by writing commands using a system terminal. There are several commands which can be used.

Commands

The script can recognise commands without case sensitivity. The command and the parameters are separated by space. The parameters of the commands need to be separated by a comma without any white space between them.

'connect <port>' is the command for connecting to the port, if possible, for example, "connect COM1".

'disconnect' to end the connection to the serial port. The command works only if the connection was established.

'exit' command for closing the program.

'open <file path>' command to open the file, if possible, and load the contents as commands for further parsing. It is an alternative to the -f parameter used when calling the program.

'home' to send the HOME command to the connected STM, if a connection is established.

'push <x>,<y>' or **'push <x,y,state>'** or **'switch <x>,<y>'** or **'switch <x, y, state>'** to push and release the button or switch. If there is any possibility to push or change the switch state to the given one. In the case of an omitted state, the program assumes 0.

'hold <x>,<y>' or 'hold <x, y,state>' to push the button on a switch and hold it in the position.

'release' to release the hold.

'light <x>,<y>' to look if the component has the ability to illuminate and return 1 in case the light of the component is on, and 0 in case it is off. It will also return the colour of the light in case there is a white cup on the component. Possible output colours are Red, Green, Blue, White.

'state <x>,<y>' in case of the switch component installed on the given position, returns the current position (1, 0, 2).

Graphical user interface

The second way to call the program is with the -g parameter. In that case, the system starts the graphical user interface. The look of the interface can be seen in figure 4.3. It is in a state where the system is connected to the COM8 serial, and the camera is not connected.

When the camera is connected, the middle part of the window will show the camera's video. It is possible to adjust the focus and light of the lenses using this view.

The text component on the right is the equivalent of a classical terminal. In this case, the commands are inserted there instead of the system terminal.

On the left side of the window, there are serial port and manual movement controls. The connection controls are zoomed in in the figure 4.4. This part of the GUI shows the state of the connection and adds the way to list the COM ports the system can see.

In case that the user tries to connect to a non-existent COM port, the warning shown in figure 4.5 is displayed. This situation can only happen if

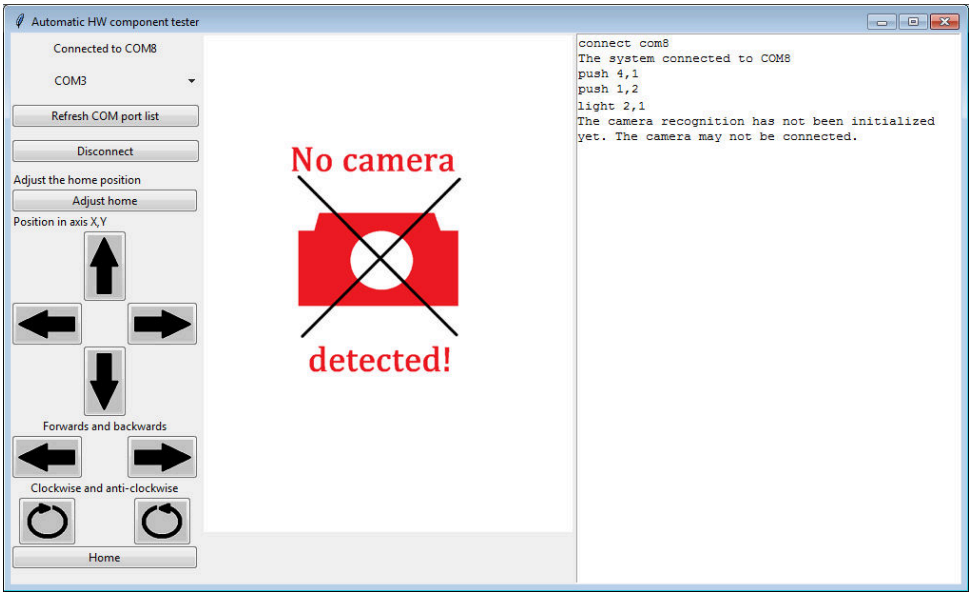
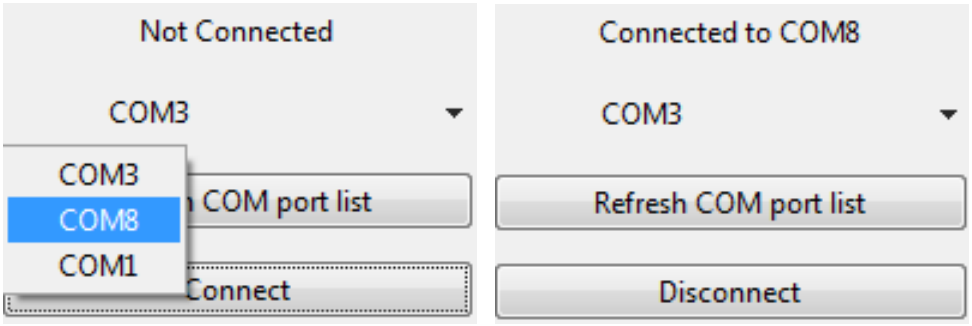


Figure 4.3: The graphical user interface without camera connected.



(a) : Choosing the connection. **(b) :** The connected state of the program.

Figure 4.4: The connection part of the GUI.

the COM port was unplugged from the computer, but the port list was not refreshed before the connect button was clicked.

The second part of the left column contains the position adjustment buttons. To activate the way to manually adjust the position, the "Adjust home" button needs to be clicked. The functionality only works when the system is connected to the COM port. The button label then changes to "Homed", to let the user end the adjustment mode.

After activating the adjustment mode, the robot moves to the last settings of the first component on the left bottom corner. The user can then move the robot using the arrows to centre the position. The X, Y axes are the ones needing adjustment. The buttons for backwards and forwards movement only make the X, Y adjustment easier, because the user can move the robot closer to the tested component to assess the position better.

Since the Phi axis sensors are not available, the system assumes that the switch tool is in a horizontal position at each boot of the control board. The

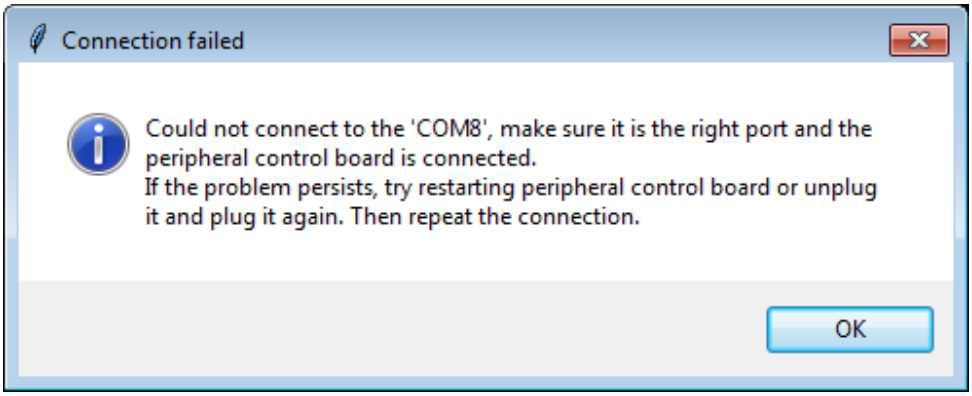


Figure 4.5: Warning that the selected COM port is no longer connected to the computer.

clockwise and anticlockwise buttons can adjust that.

The Home button sends the robot to the 0,0,0,0 position in the left bottom corner. It only works if the adjustment mode is off and the control board is connected.

4.4 Image processing

An important section of the computer program is image recognition. The robot needs to be able to find the working area in the image and detect a position of the components. It needs to recognise the state of lights and their colour. It also needs to be able to tell the state of the switch.

For the recognition, the OpenCV module built with TBB was used. The [12] book is an efficient way to learn how to use the module.

For camera interface, the simple-pyspin [20] module was used. It is a python wrapper for the Flir PySpin library, which is necessary to communicate with the Flir Chameleon 3 camera. In order to use it, the Spinnaker SDK [9] had to be installed as well.

Although the camera is mounted in the portrait orientation, its output is a landscape image. Any pictures presented here are shown without any rotation.

4.4.1 Finding the area of interest

The camera captures a slightly larger image than necessary to see the working area. The freedom in the image position is essential so that the camera's slightest movement or change in its shooting angle does not crop a vital part of the view.

The full camera view can be seen in figure 4.6. The image size is 1288×964 pixels.

The above also means that the program needs to find the actual placement of the working area in the image. It is accomplished by finding the edges of the image and detecting lines in it. Since the robot consists of aluminium profiles, which are surrounding the working area, the lines can be easily seen.

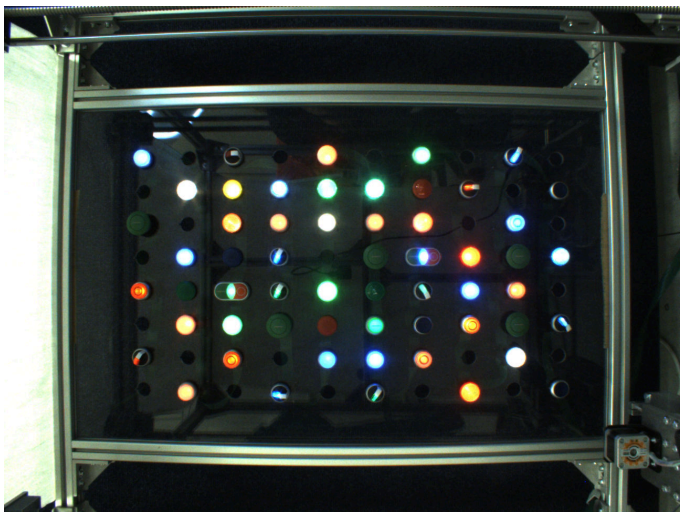


Figure 4.6: The camera image before the area of interest setting.

The problem is finding the rectangular in the image, detailed, for example, in [18].

For the detection, the Gaussian blur is used with a kernel size of 11 pixels. It makes use of Canny function possible so that the edges can be found, as seen in figure 4.7.

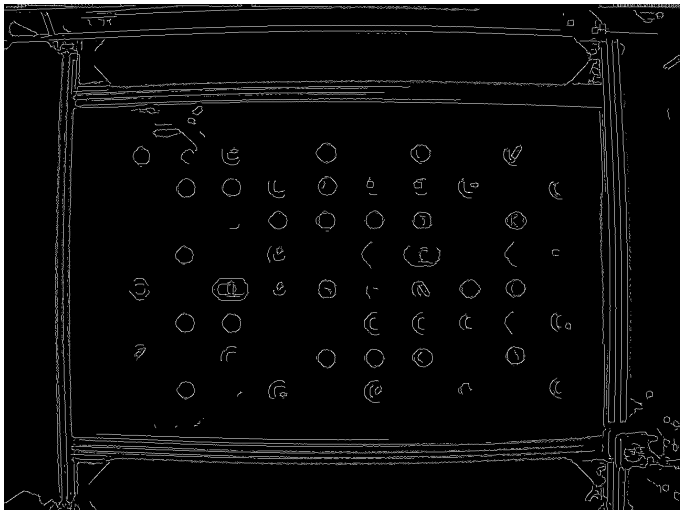
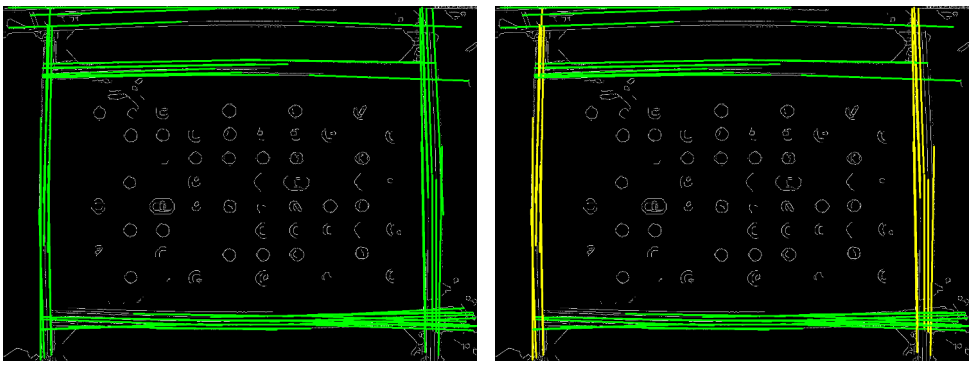


Figure 4.7: The edges detection.

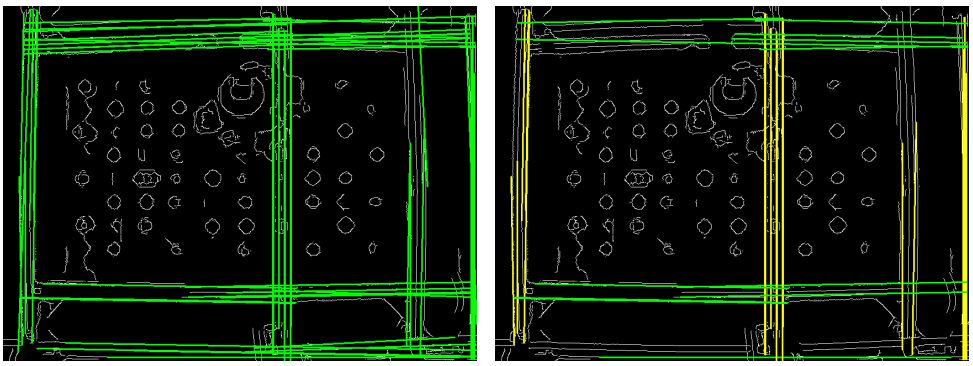
The OpenCV function HoughLinesP is used to find long enough lines in the edged figure. After that, the program finds lines, which are horizontal and vertical to consider as the possible boundaries. The detected lines and their sorting can be seen in figure 4.8. Figure 4.8a shows all the lines detected in the image. Figure 4.8b shows sorted lines in the directions; the green ones are considered for the horizontal cropping and the yellow ones for the vertical one.



(a) : Lines found in the image. **(b) :** Sorting the lines to the directions.

Figure 4.8: The line detection.

After that, the K-Means sorting of the middles of the lines is done for each of the axes separately. The sorting is done to the three groups so that the possible obstruction by the robot's cart in the image does not pose a problem. The example of such a setting can be seen in figure 4.9. In this figure, the lines' selection can be seen more clearly. The image contains some non-horizontal ones, which the algorithm does not select for the sorting.



(a) : Lines found in the image. **(b) :** Sorting the lines to the directions.

Figure 4.9: The line detection.

After the sorting to the groups, the filtering in the groups is performed. If the group has enough lines, it will repeatedly filter out the most distant value from the average of the group. In case the distance is more than the size of the aluminium profile. Then, the average is recalculated, and the filtering continues. In the end, any extreme values are separated from the group, so they do not influence the final average.

The filtering results in three values, from which the program calculates the final size of the working area. The selected group then adds a few pixels to make some space around the detected area and sets the position to the area of interest in the camera. Every image after that will contain only the focused part. The resulting cropped image can be seen in figure 4.10.

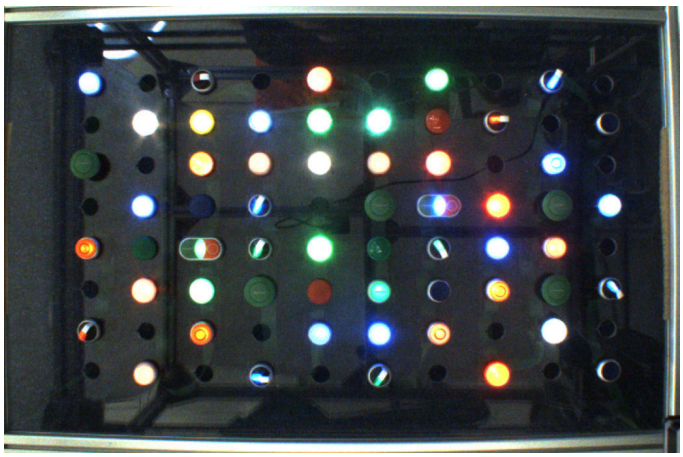
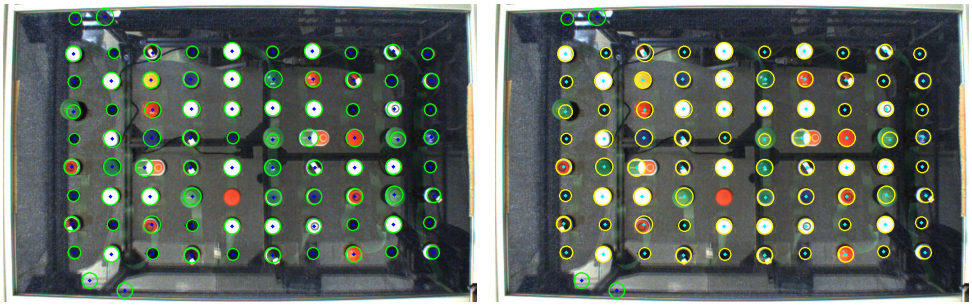


Figure 4.10: The image after the area of interest settings.

4.4.2 Component position detection

Within the working area, the components' positions need to be detected. For that, the HoughCircles function from OpenCV is used. The detection finds any circles with the given radius. The found circles are shown in figure 4.11a. Since the plexiglass reflects its surroundings, the detection can find some circles that are not a component position, but rather some reflection in the open space. It can be filtered out by knowing the size of the working space and the components' positions in mm, which can be transformed into pixels. The filtered circles can be seen in figure 4.11b, where yellow circles are the selected ones for the position calculation.



(a) : Circles detected in the image. **(b) :** Filtered circles.

Figure 4.11: The component position detection.

The centres are taken from the selected circles, and the X and Y positions are handled separately. The K-Means sorting is called on them, with the number of groups given by the glass shape, whose numbers of X and Y positions are known. Then the average position of the centres in the group is calculated, and the position's matrix is generated. The detected matrix of the component positions can be seen in figure 4.12.

For the light and position detection, each component is considered separately. The component area is calculated from the space between the

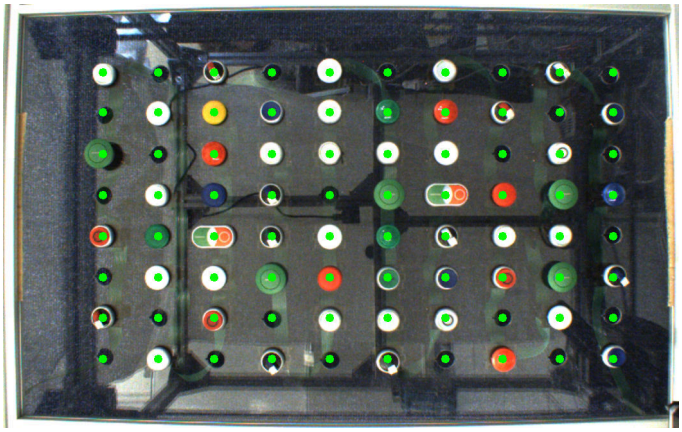


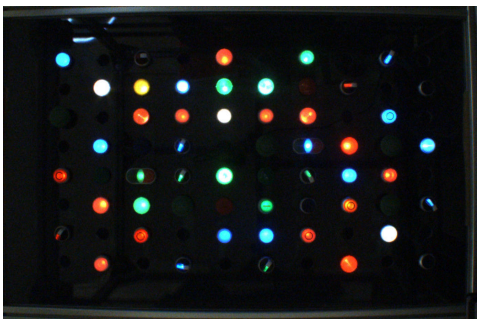
Figure 4.12: The detected positions of components.

components given by the plexiglass settings.

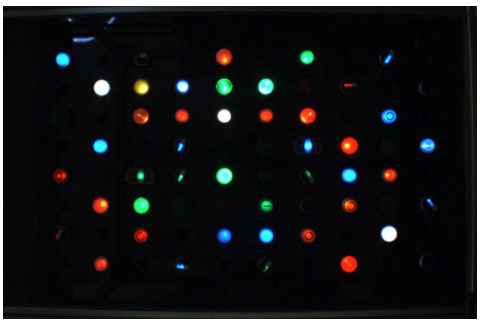
4.4.3 Light detection

For components supporting a light, the on/off detection was needed as well as recognition of the light's colour. This information is valid only if the component uses a white plastic cap which does not alter the colour of the light underneath it. Otherwise, the recognition returns some colour, but the user will only see on/off value.

Since the image needs to be bright enough for the program to see the switch position as well, the light can not be recognized using the same settings. That is why a different exposure is set for the camera to take the picture, and then the settings are returned to the previous ones with the automatic gain and exposure time. There are two different settings for recognition. The one used in every scenario is the exposure time of 10 000 µs, which can be seen in figure 4.13a. The second one is used for validation of the white colour with an exposure time of 5 000 µs, which can be seen in figure 4.13b.



(a) : Image taken with exposure time 10 000 µs.

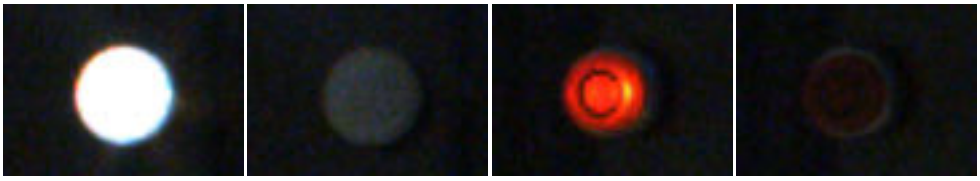


(b) : Image taken with exposure time 5 000 µs.

Figure 4.13: The image taken for light recognition.

The program will then cut the observed component. The component

examples can be found in figure 4.14.



(a) : The light component with White light on.

(b) : The light component with White light off.

(c) : The button component with Red light on.

(d) : The button component with Red light off.

(e) : The double button with Green light on.

(f) : The double button with Green light off.

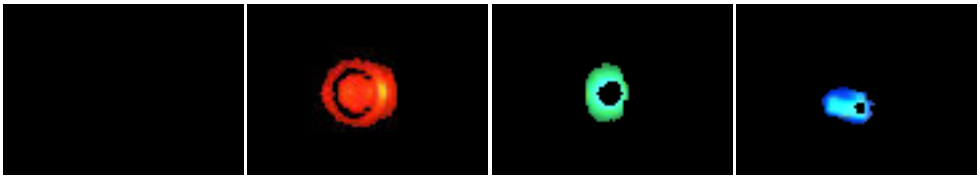
(g) : The switch component with Blue light on.

(h) : The switch component with Blue light off.

Figure 4.14: The components image with 10 000 µs exposure time.

Colour light recognition

The image of the observed component is transferred to the HSV (Hue Saturation Value) colour model. In this representation, the saturation and value are filtered by a threshold. The result of the filtration can be seen in figure 4.15. The figure 4.15a shows why the method is not suitable for white colour detection. The saturation filter will remove all white pixels.



(a) : The light component with White light on.

(b) : The button component with Red light on.

(c) : The double button with Green light on.

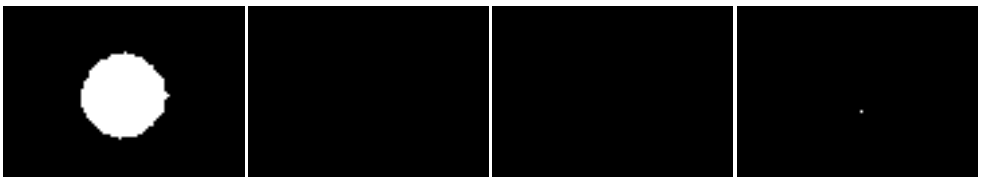
(d) : The switch component with Blue light on.

Figure 4.15: The result of the threshold application.

After that, the non-zero pixels are counted and compared with the area needed for the component. For a switch, the area is significantly smaller than for the other components. If the area condition is fulfilled, the colour left there is averaged. The resulting colour is selected as the highest value from RGB.

White light recognition

After colour recognition, white light detection takes place. The component image is transferred to grayscale, and only the pixels with the value of 255 are used. The number of pixels is compared with the area as in the previous case. The figure 4.16 shows the filtering. In this case, only the white light component shows any sign of illumination. The switch component shows only one pixel, which is bright enough to be shown on the white detection.



(a) : The light component with White light on.

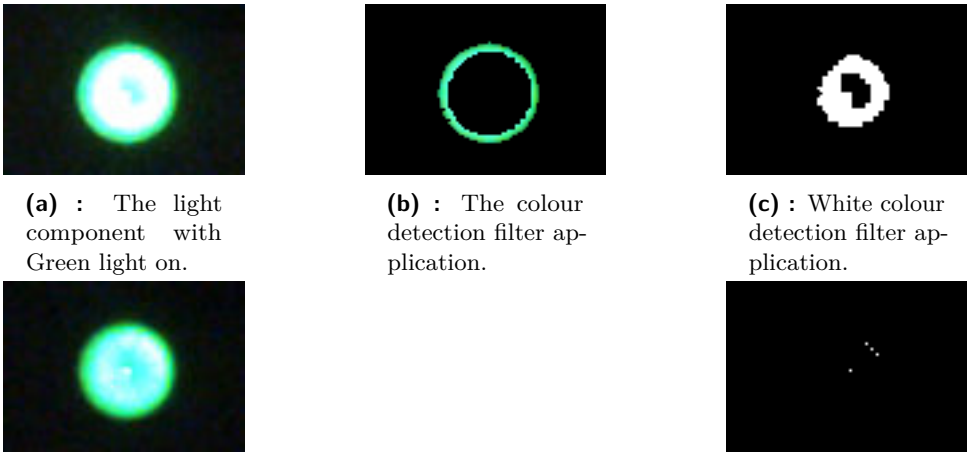
(b) : The button component with Red light on.

(c) : The double button with Green light on.

(d) : The switch component with Blue light on.

Figure 4.16: The result of the white threshold application.

However, there were multiple problems with too bright colour light. The figure 4.17 shows an example of green light with a white plastic cap on the component. The colour detection will determine that the light is on, and it has a green colour. The white detection then determines that the light is on with the white colour. In this case, another image is taken, with the lower exposure time, to determine, if the colour is green or white. The new image shows only a few white pixels, which rules out the white colour light.



(a) : The light component with Green light on.

(b) : The colour detection filter application.

(c) : White colour detection filter application.

(d) : The exposure time 5 000 µs image.

(e) : White colour detection with lower exposure.

Figure 4.17: The result of the white threshold application.

Glint problem

There was a problem with a glint from the room lights above the robot. The glint was causing the program to often not recognise the off state. When changing the threshold and area, the light was usually not detected. That is why a shade was added to the robot construction. However, that resulted in low light, and so some additional sidelights had to be added as well.

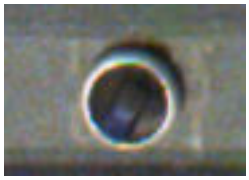
4.4.4 Switch position detection

One of the program function is switch position detection. Currently, only positions of the available ones are supported. The upright position is marked as state zero, the right position is state one, and the left position is state two. In the camera images, the positions are rotated by 90 degrees to the left.

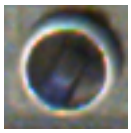
Problems with recognition.

The recognition was not reliable enough without some physical aids. The first problem was already mentioned in the previous chapter - the glint from the room light was too strong, and the component's image was usually half white. The shade and sidelights solved this problem.

The second problem was that the component itself is mostly dark. The switch handle, which should be visible to the edge detector is black on the black background, which is the reason the edge detector was not able to find anything. Figure 4.18 shows one of the attempted approaches. The component cut out is reduced to eliminate neighbouring reflections. Then the edge detector was used. The figure 4.18c shows that the edges were only visible on the switch's edge but not inside it, providing no useful data.



(a) : The switch component.



(b) : The switch component cropped around a detected circle.

(c) : The edge detection of the switch component.

Figure 4.18: The switch component without the supporting stickers.

Another attempt using edge detection was seeing something inside the switch, but none of the features leads to the ability to detect the correct position. It was usually some noise or only part of the switch's handle, which did not lead to an accurate position.

Another explored approach was the internal OpenCV brute force matcher. However, the image resolution was so low that it was not able to find anything.

Therefore, the temporary solution was provided by white stickers. The raised part of the switch which is not part of its translucent cup was marked with white stickers so that the edge would be visible.

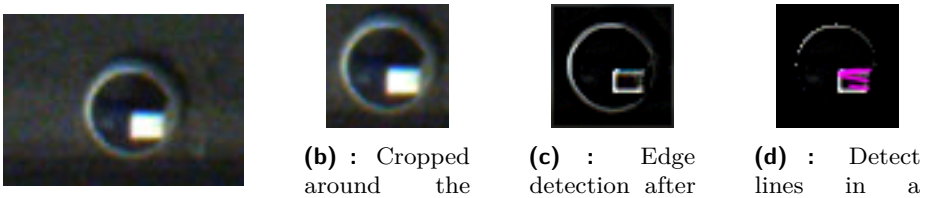
Recognition algorithm

For switch recognition, the automatic gain and exposure settings are used. The component is cut from the image taken from the camera. Then the exact position is found by detecting the circle with the specified radius. The program then crops the image so that the background around does not interfere with the algorithms.

The next step is denoising the image and finding the edges. The PIL Image edge detection turned out to be more useful than the OpenCV one since it takes colours into account. Then the component circle is masked out so that the detection of lines will not detect anything outside the component.

When the edges are created, the HoughLinesP function is used. The parameters (as threshold and minimum line length) are set based on whether the light is on or not. It is determined by using the same function, which was described in chapter 4.4.3.

The above steps are shown in figure 4.19 for the light off and figure 4.20 for light on.



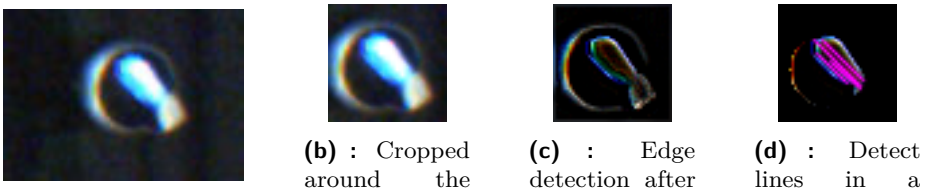
(a) : The switch component.

(b) : Cropped around the switch component.

(c) : Edge detection after denoising image.

(d) : Detect lines in a masked image.

Figure 4.19: The switch detection algorithm with a light off, state = 0.



(a) : The switch component.

(b) : Cropped around the switch component.

(c) : Edge detection after denoising image.

(d) : Detect lines in a masked image.

Figure 4.20: The switch detection algorithm with a light on, state = 1.

The lines detection algorithm is called up to three times. The first time, the threshold and minimum line length are set based on the light information. If no lines are detected, the parameters are relaxed, and the function is called again. If no lines are detected still, the system returns an error value. The caller code can try it again on a newly captured image which usually solves the problem.

If any lines are detected by the first or second try, the filtering takes place. The angles with respect to the horizontal line are calculated, and the extreme values are filtered out. If there are not enough values after the filtration, the algorithm tries to lower the parameters once again and captures new lines.

Then it repeats the filtering. The average from the remaining angles is taken into account.

The state of the switch is then split into three categories, which are shown in table 4.1.

angle [°]	<-90; -22)	<-22; 22>	(22; 90>
state	2	0	1

Table 4.1: The state categories

The reliability of recognition

The algorithm was tested repeatedly. The test was done in a changing environment and did not make long pauses between the switch's state queries. The reliability appeared to be around 95 %. This number is a little bit lower for the component with the light off than for those with the light on.

Most likely, it would be possible to improve the algorithm. However, the error rate appeared lower when the detection was not used continuously, for example, in real-life test scenarios which had more delays between requests.

4.4.5 Checking the visibility of the component

The previously described algorithms for finding the working area boundaries, finding components position and the switches position recognition are used when initialising the image recognition part of the application. Detecting component positions and the switches' state could be problematic if the robot was in the way. If it is the case, the detection needs to be repeated after the control board's connection is completed.

The user can ask for light recognition and switch position functions. It is only possible if the control board is connected. In that case, the system calculates if the robot cart is in the way. If yes, the movement in Y-axis is performed to make the recognition possible.

Chapter 5
Conclusion

This diploma thesis describes the construction of an automatic tester of device control elements. The construction consists of a four axes CNC with X, Y and Z linear axes, and a Phi rotation axis.

The hardware of the system was designed, the components selected and modelled, and the models for the cut-outs from the aluminium and for the 3D-print were created. A printed circuit board was designed, manufactured and installed.

The control board created around the STM32F401 was programmed so that the system could communicate with a computer and generate steps for the motors. The source code can be found at `https://gitlab.fel.cvut.cz/novakha5/diploma_stm/-/releases`.

The second part of the programming was done for a personal computer. The communication with the PCB and the camera was successfully implemented as well as an application user interface. The image recognition ensuring the detection of component illumination and its colour was implemented as well as the switch position detector. The program can be found at `https://gitlab.fel.cvut.cz/novakha5/diploma_pc/-/releases`.

The robot functionality was tested, with the image recognition error rate determined as 5 %.

5.1 Current state

The system can push buttons and change the state of the switches. It can determine if the component's light is on and return its light colour if the component plastic cup is not coloured. It can also determine the state of the switch.

One of the system's issues is surrounding light that needs to be regulated, which is now solved temporarily with the shade and sidelights.

During the testing phase, there was a suspicion of memory corruption in the microcontroller code. Sometimes, the code could not change the counter timer input, so the steps were not counted. It would also manifest by moving at the slowest speed without following the speed curve.

This issue is mitigated automatically when the robot triggers the proximity sensors. The system appears to be able to recover from such a state. While

looking for the cause of the problem, the dynamic allocation of the memory was found in the HAL libraries, which might be causing the problem, when used in combination with the RTOS.

5.2 Future extensions

It would be beneficial to buy more suitable lights to mount them to the robot in the future. The lights could be LED strips with a diffuser, with 24 V input to remove the need for an additional power source.

Another improvement would be to change the plexiglass to a matt opaque material which would make the image recognition more reliable.

Precise position switches could be added to the homing position so there would be no room for error in the position of the robot between two power-ups.

The future improvements in the printed circuit board were discussed in chapter 3.2.1.

The program could also be extended to recognise the 4-way switch positions if such components are delivered for testing.

Appendix A

Circuit scheme and printed circuit

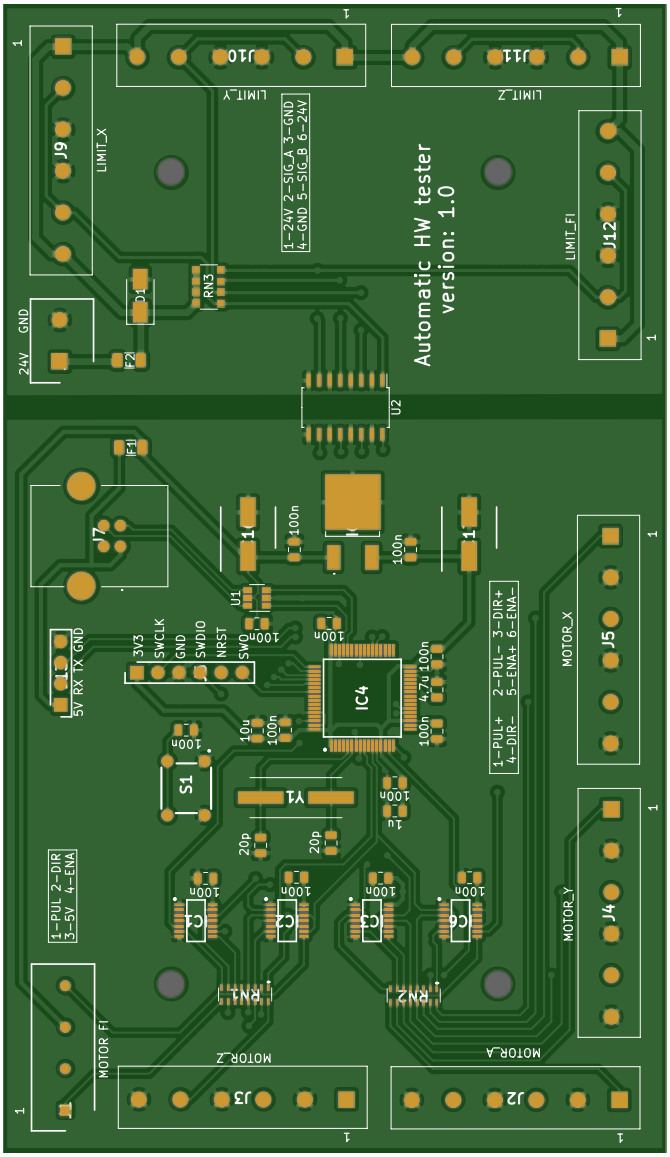


Figure A.1: Visualisation of top of the PCB, generated from Gerber files by [25].

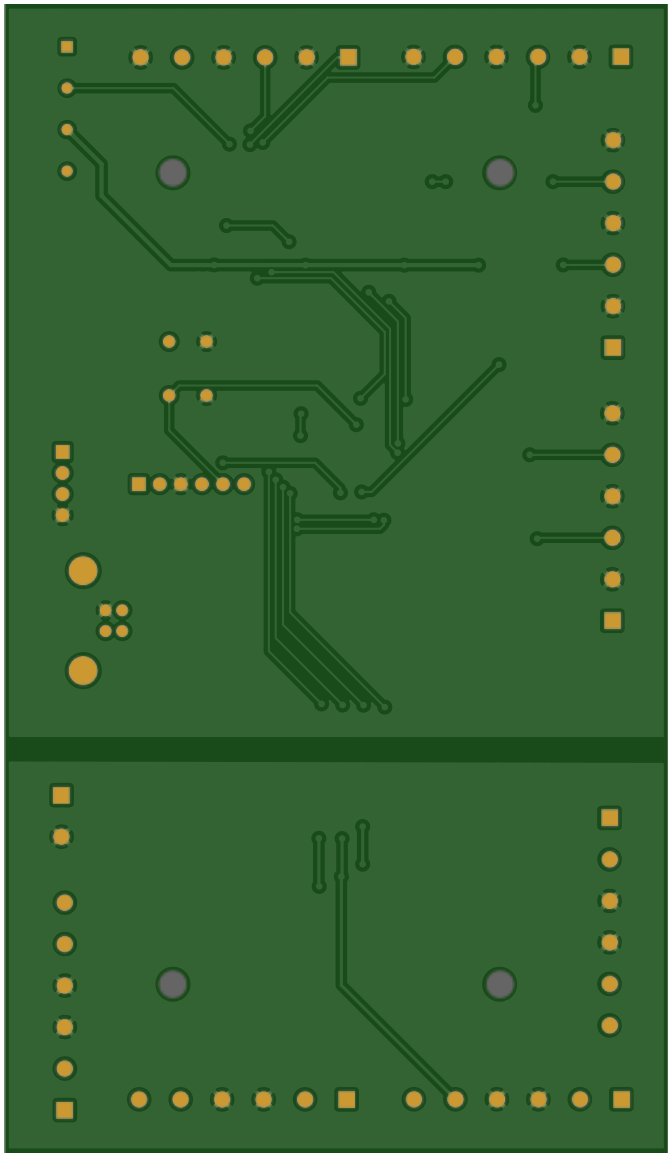
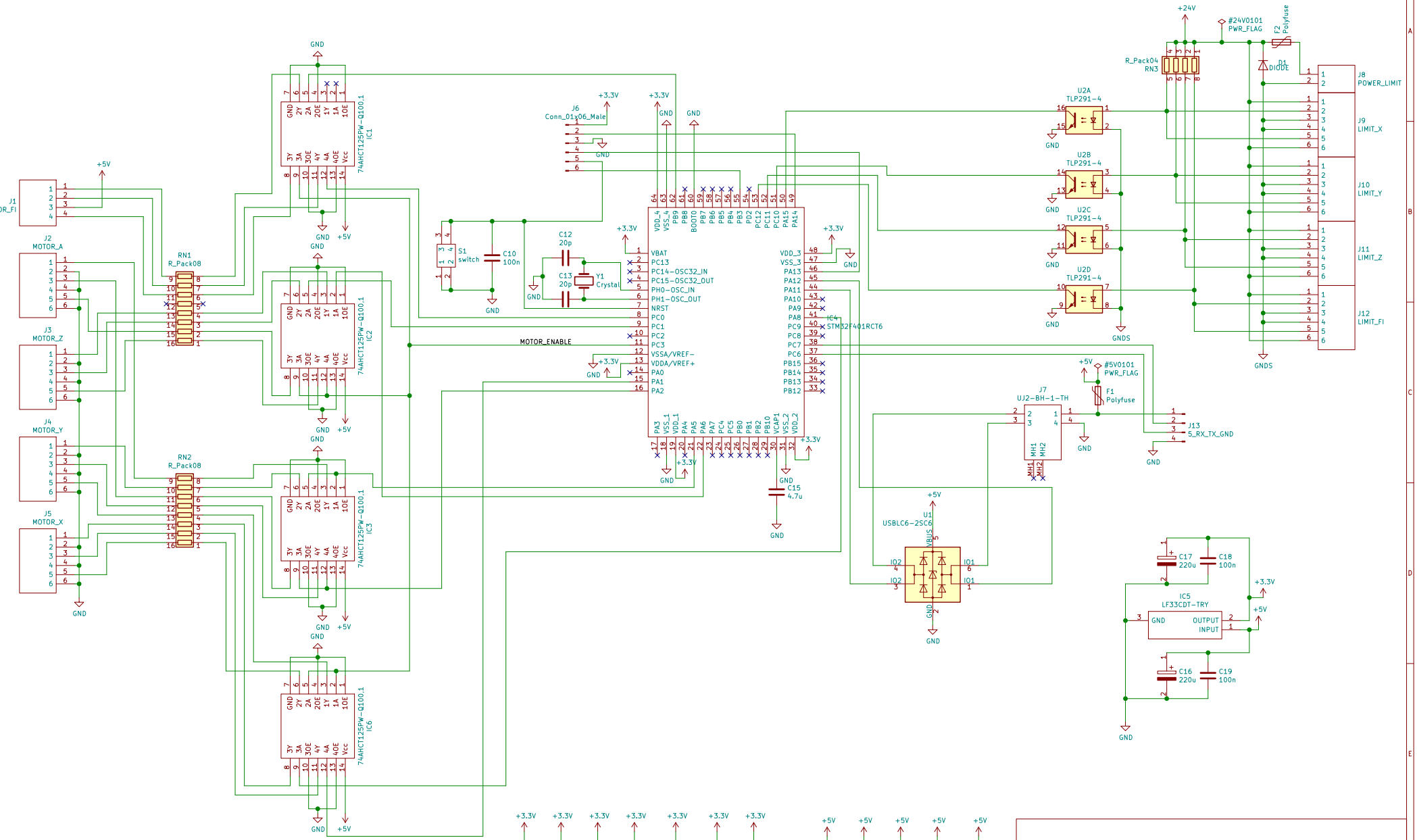


Figure A.2: Visualisation of bottom of the PCB, generated from Gerber files by [25].


J1
MOTOR_FI
J2
MOTOR_A
J3
MOTOR_Z
J4
MOTOR_Y
J5
MOTOR_X
RN1
R_Pack08
RN2
R_Pack08
IC1
74AHCT125PW-Q100.1
IC2
74AHCT125PW-Q100.1
IC3
74AHCT125PW-Q100.1
IC6
74AHCT125PW-Q100.1
J6
Conn_01x06_Male
S1
switch
C10
100n
C12
20p
C13
20p
Y1
Crystal
MOTOR_ENABLE
IC4
STM32F401RCT6
C15
4.7u
U1
USBLC6-2SC6
J7
UJ2-BH-1-TH
#5V0101
PWR_FLAG
F1
Polyfuse
J13
5_RX_TX_GND
+24V
#24V0101
PWR_FLAG
F2
Polyfuse
D1
DIODE
R_Pack04
RN3
U2A
TLP291-4
U2B
TLP291-4
U2C
TLP291-4
U2D
TLP291-4
GNDS
J8
POWER_LIMIT
J9
LIMIT_X
J10
LIMIT_Y
J11
LIMIT_Z
J12
LIMIT_FI
C17
220u
C18
100n
IC5
LF33CDT-TRY
C16
220u
C19
100n
C1
100n
C2
100n
C3
100n
C4
100n
C5
100n
C6
1u
C7
10u
C20
100n
C8
100n
C9
100n
C11
100n
C14
100n
Sheet: /
File: pcb.sch
Title: Automated HW tester
Size: A3
Date: 2020-03-16
Rev: 1
KiCad E.D.A. eeschema (5.1.4)-1
Id: 1/1

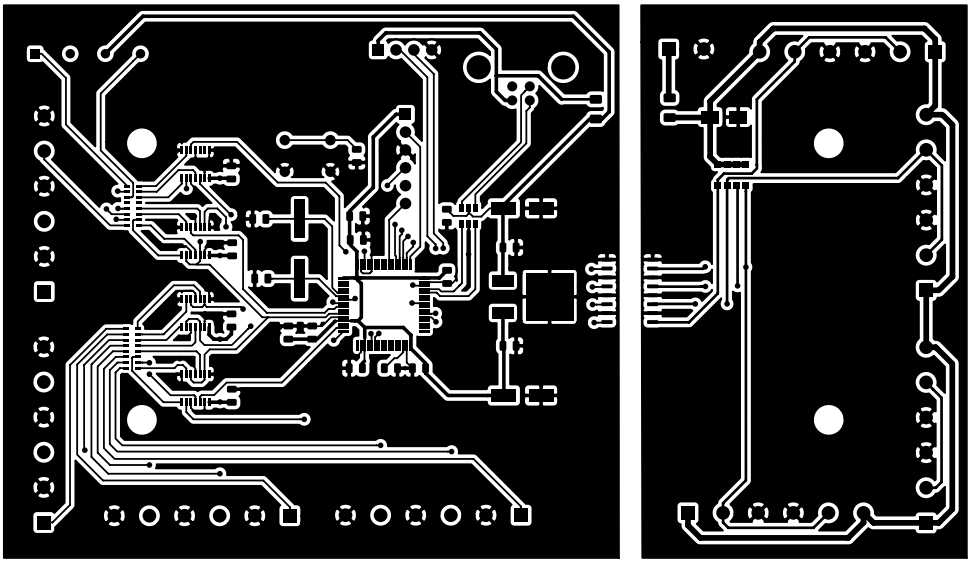


Figure A.3: Front copper layer.

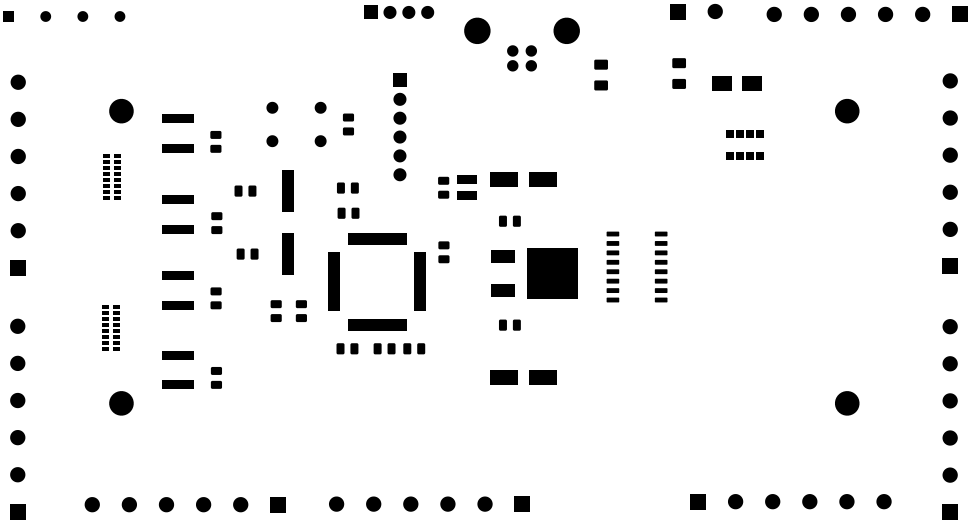


Figure A.4: Front mask layer.

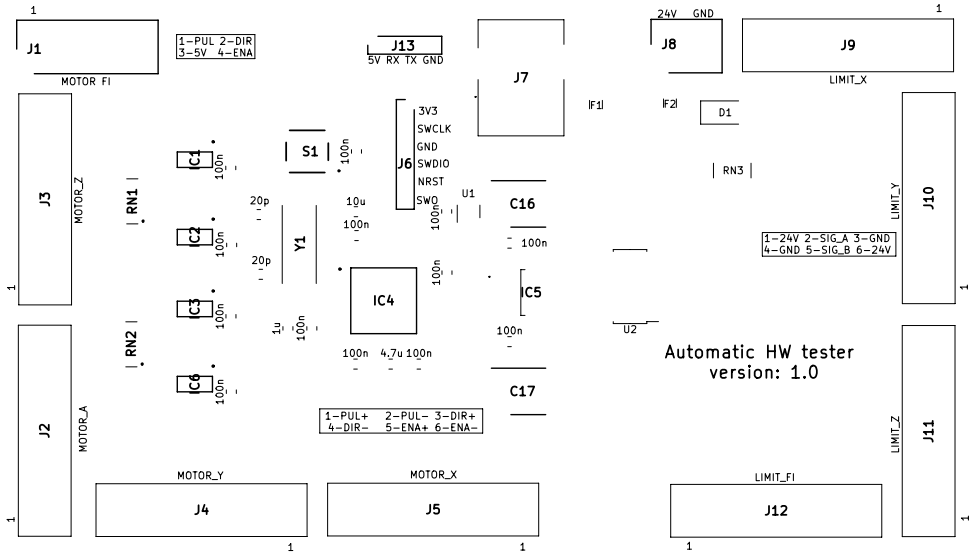


Figure A.5: Front silkscreen layer.

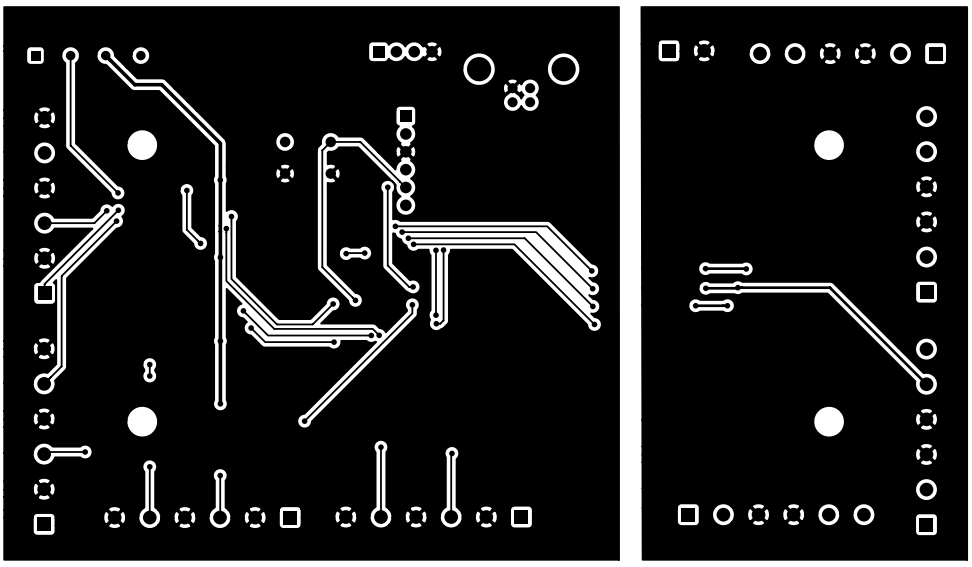


Figure A.6: Back copper layer.

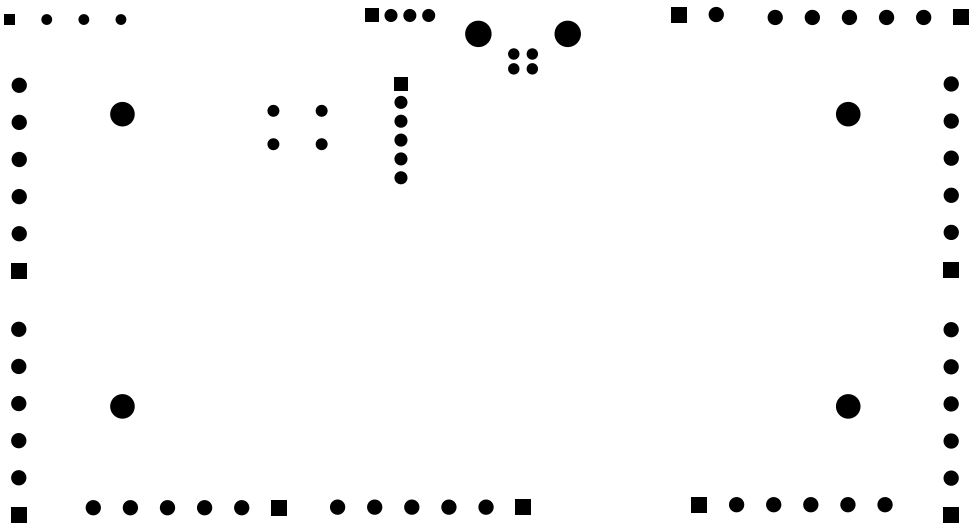


Figure A.7: Back mask layer.

Appendix B

List of components

Component	Cost together [CZK]
Camera	6 400
Lenses	738
USB3 cable	159
Total cost [CZK]	**7 297**

Table B.1: Camera and accestories

Component	Number of instances [pcs]	Cost together [CZK]
Motors and drivers		
Motors for linear movement axis	4	3 920
Motor Phi axis	1	296
Drivers for linear movement axis	4	6 360
Driver Phi axis	1	496
Total motors and drivers cost [CZK]	**11 045**	
Motor connection		
Motor mount for NEMA 23	1	88
Motor mount for NEMA 17	1	48
Flexible coupling 6.35 mm/8 mm	4	260
Total connection cost [CZK]	**396**	
Accessories		
Inductive sensors	6	752
Electric source 24 V - 14.5 A	1	1 050
Extension cables	-	about 500
Total accessories cost [CZK]	**2302**	

Table B.2: Motors and accessories

Component	Number of instances [pcs]	length [mm]	Cost together [CZK]
Aluminium X profiles			
30×30 mm	6 4 4 3 2 2 1	980 880 850 820 825 475 600	
40×40 mm	1	820	
Total profile cost [CZK]		**4360**	
Fastenings for system			
End cup	8	-	269
T-slot M6 8 mm	167	-	2 919
T-slot M6 6 mm	8	-	152
Rectangular support	51	-	2 648
Others (screws, washers, nuts)	-	-	about 600
Total fastenings cost [CZK]		**6 588**	
Movement Leads			
Ball screw set	2 1 1	1 000 600 250	2 556 1 053 802
Linear rod	2 1 1	1 030 578 210	576 162 59
Attachment of rods perpendicular	6	-	420
Attachment of rods direct	2	-	110
Linear seat	5	-	443
Total leads cost [CZK]		**4 803**	
Energetic chains			
15×30 mm	1	1 040	518
10×20 mm	2	1 000	346
Total chains cost [CZK]		**864**	

Table B.3: Robot skeleton parts

Component	Cost together [CZK]
Aluminium parts	9 565
Plexiglass	1 190
The printed circuit board (130 mm×80 mm)	780
Total cost [CZK]	**11 535**

Table B.4: Custom made parts

Component	Number of instances [pcs]	Cost together [CZK]
Micro-controller STM32F401RCT6	1	131
Linear regulator LF33CDT-TRY	1	22
USB protection STM USBLC6-2SC6	1	11
Buffer 74AHCT125PW	4	65
Optocoupler TLP291-4	1	28
Crystal ATS08ASM-1	1	10
Capacitor 20 pF	2	5
Capacitor 0.1 µF	12	30
Capacitor 1 µF	1	3
Capacitor 4.7 µF	1	3
Capacitor 10 µF	1	4
Capacitor 220 µF	2	10
Resistor 100 Ω	16	17
Resistor 220 kΩ	4	9
Fuse 24 V 500 mA	1	3
Fuse 48 V 390 mA	1	3
Diode 45 V	1	11
Tactile Switch FSM4JH	1	3
USB connector UJ2-BH-1-TH	1	14
Barrier Terminal Blocks six-position	8	256
Barrier Terminal Blocks four-position	1	18
Barrier Terminal Blocks two-position	1	10
Pin header six-pin	1	7
Pin header four-pin	1	4
Total cost [CZK]	**677**	

Table B.5: PCB components

Appendix C

Firmware initialization

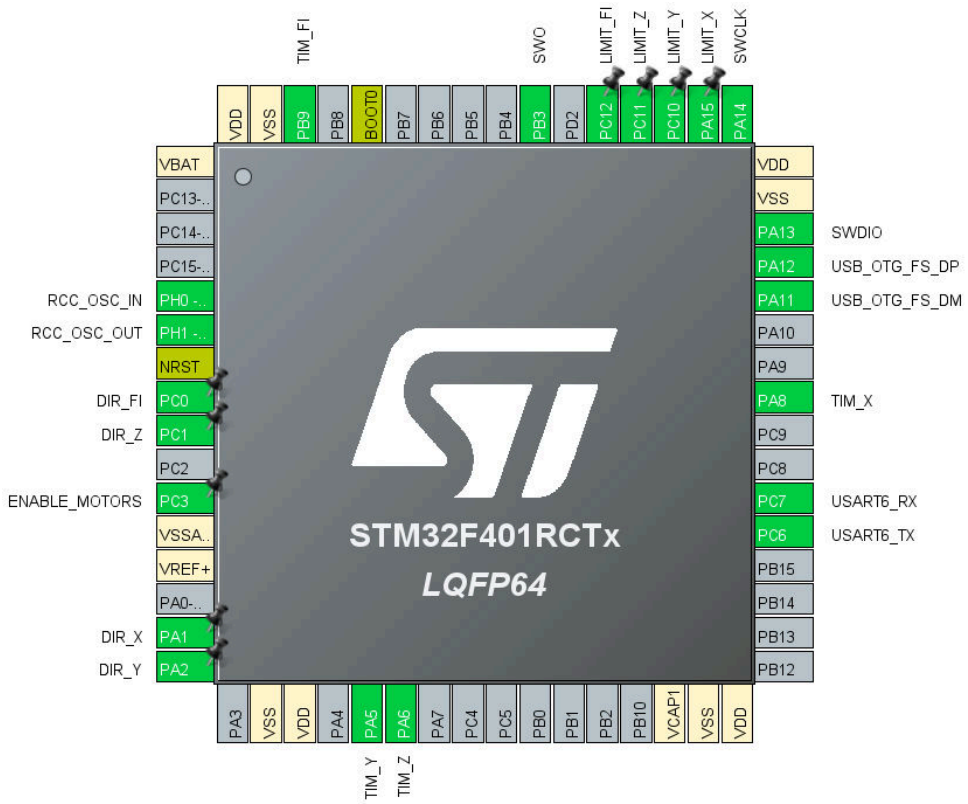


Figure C.1: The pin configuration in STM32CubeMx program [35].

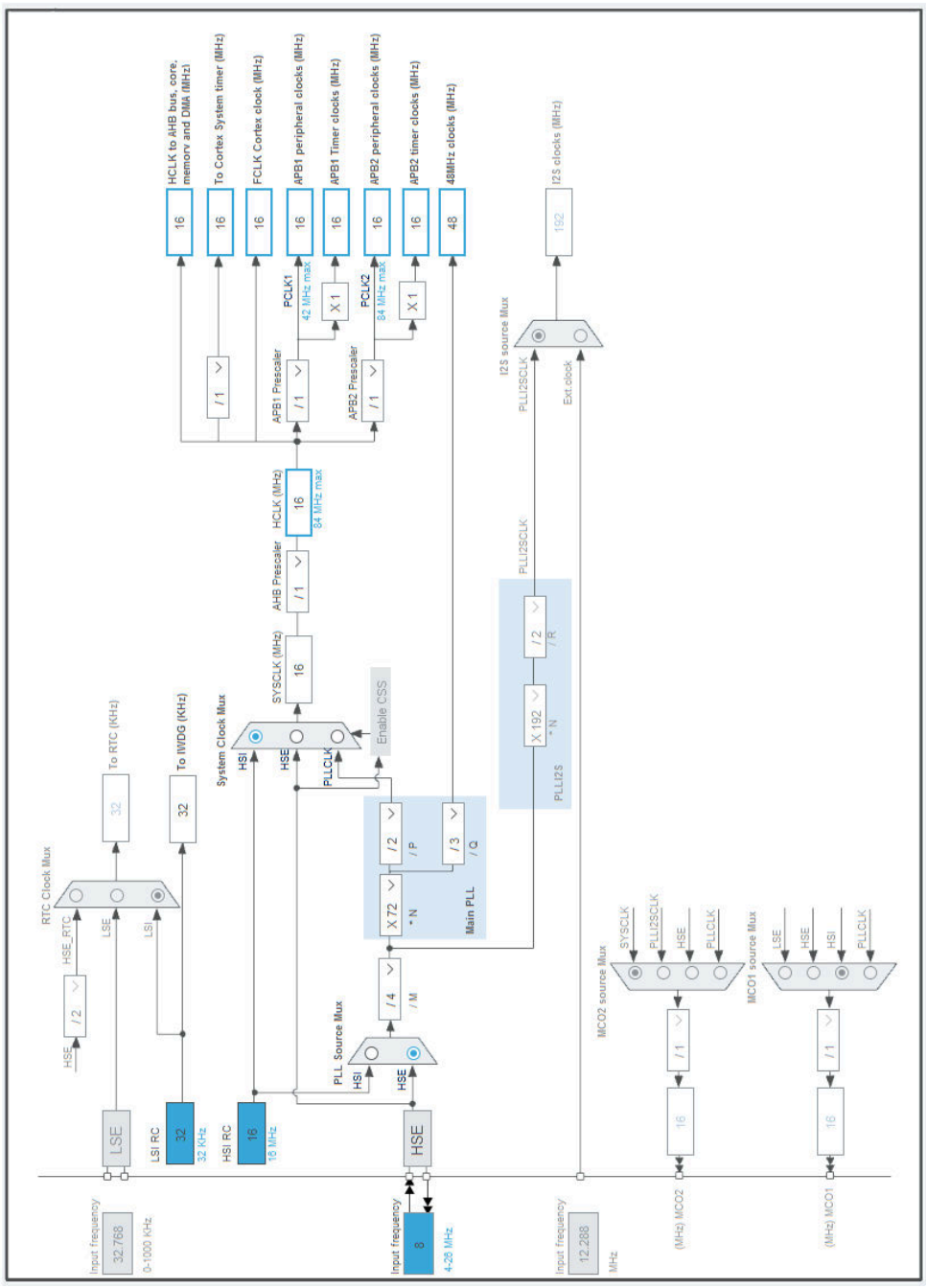


Figure C.2: The clock configuration in STM32CubeMx program [35].

Appendix D

Contents of the enclosed CD

- The assignment
- The thesis itself in pdf
- The **LaTeX** folder with LaTeX source codes
- **OpenScad_models** folder containing 3D models for the aluminium cuts, plastic components and the plexiglass working area
- **Skeleton** file containing all the modelling to plan the shape of the CNC in the FreeCAD
- **PCB** folder with KiCAD files for the Control board
- **STM_program_part** folder with source codes for the microcontroller and the project files for Atollic and STM32CubeMX
- **PC_program_part** folder containing source codes of the computer part of the thesis in Python
- **STM_doc.zip** contains zipped html Doxygen documentation for the STM part of the program
- **PC_program_doc.zip** contains zipped html Doxygen documentation for the computer part of the program
- **Video** containing demonstration of the robot's operation

Appendix E

Bibliography

[1] Accelonix. Takaya: Flying Probe Test. Available at https://accelonix.co.uk/test-and-programming/takaya-flying-probe, accessed: 3.1.2021.

[2] BG Ingénierie. 3 axis RFID tester . Available at https://www.bginge.com/product/3-axis-rfid-tester-73, accessed: 3.1.2021.

[3] Computar. *T0412FICS-3*, May 2004. Available at https://computar.com/file?id=174.

[4] Cosber. Automatic/Robotic Headlight Tester. Available at http://www.vis-id.com/automatic_headlight_tester.html, accessed: 3.1.2021.

[5] CTS electronic components. *ATS/ATS-SM Series Low Cost Quartz Crystal*. rev.K, available at https://www.alldatasheet.com/datasheet-pdf/pdf/597333/CTS/ATS08ASM-1.html.

[6] Eaton. *Product Range Catalog, Command and Indication, Build it in.*, October 2016. Available at https://ecat.eaton.com/flip-cat/COMCON1_EN/blaetterkatalog/pdf/complete.pdf#page=21&zoom=130,317,840.

[7] Kutting electronics. Flying Probe Test (FPT) - In-Circuit Test (ICT). Available at https://www.kuttig.eu/en/ems/inspection-test/flying-probe.html, accessed: 3.1.2021.

[8] FLIR. *IMAGING PERFORMANCE SPECIFICATION, FLIR CHAMELEON3, USB3 Vision*, January 2017. Version 4.1, available at https://flir.netx.net/file/asset/13195/original/attachment.

[9] Inc. FLIR® Systems. Spinnaker SDK. Available at https://www.flir.eu/products/spinnaker-sdk/, accessed: 23.12.2020.

[10] FreeCAD Team. FreeCAD Your own 3D parametric modeler. Available at https://www.freecadweb.org/, accessed: 18.11.2020.

[11] GadgEon. Robot Arm Based Testing. Available at https://www.gadgeon.com/iot-services/robot-arm-based-testing/, accessed: 2.1.2021.

[12] Adrian Kaehler Gary Bradski. *Learning OpenCV*. O'Reilly Media, Inc., September 2008. First edition, ISBN: 9780596516130.

[13] Helen. Choosing the right motor for your project – dc vs stepper vs servo motors. Available at https://www.seeedstudio.com/blog/2019/04/01/choosing-the-right-motor-for-your-project-dc-vs-stepper-vs-servo-motors/, accessed: 2.1.2021.

[14] CSc. Ing. Jan Fisher. *Optoelektronické senzory a videometrie*. Vydavatelství ČVUT, 2002.

[15] Instron. 6-Axis Robotic Automated Testing System. Available at https://www.instron.jp/en/products/testing-systems/automated-testing-systems/at6-6-axis-robotic-automated-testing-system, accessed: 3.1.2021.

[16] JetBrains s.r.o. PyCharm The Python IDE for Professional Developers. Available at https://www.jetbrains.com/pycharm/, accessed: 22.12.2020.

[17] KAV SystemsEngeneering. Flying Probe test for Prototyping. Available at https://www.kavsys.it/2016/10/31/flying-probe/, accessed: 3.1.2021.

[18] Ilya Kavalerov. Using Pattern Recognition to Automatically Crop Framed Art. Available at https://artsy.github.io/blog/2014/09/24/using-pattern-recognition-to-automatically-crop-framed-art/, accessed: 26.12.2020.

[19] KiCad Team. KiCad EDA. Available at https://kicad.org/, accessed: 16.12.2020.

[20] Dustin Kleckne. simple-pyspin 0.1.1. Available at https://pypi.org/project/simple-pyspin/, accessed: 23.12.2020.

[21] Stephen Lambrechts. Samsung built a robot butt just to test its smartphones' durability. Available at https://www.techradar.com/news/samsung-built-a-robot-butt-just-to-test-its-smartphones-durability, accessed: 3.1.2021.

[22] Leadshine Technology Co., Ltd. *57HS Series Hybrid Stepping Motors*. Available at http://www.cncshop.cz/PDF/57HS09-22.pdf.

[23] Marius Kintel. OpenSCAD The Programmers Solid 3D CAD Modeller, May 2019. Available at https://www.openscad.org/, accessed: 19.11.2020.

[24] Matric. ICT TESTING VS FLYING PROBE TESTING. Available at https://blog.matric.com/ict-testing-vs-flying-probe-testing, accessed: 3.1.2021.

[25] Mike Cousins. Tracespace view. Available at https://tracespace.io/, accessed: 17.12.2020.

[26] Mike Lynch. The Basics of Computer Numerical Control), May 2019. Available at https://www.cncci.com/post/the-basics-of-computer-numerical-control, accessed: 2.1.2021.

[27] Nexperia. *74AHC125; 74AHCT125 Quad buffer/line driver; 3-state*, 2020. Available at https://assets.nexperia.com/documents/data-sheet/74AHC_AHCT125.pdf.

[28] Nur Hidayanti Binti Ambrizala, Awais Farooqib, Osama I. Alsultanc, Nukman Bin Yusoff. Design and Development of CNC Robotic Machine Integrate-able with Nd-Yag Laser Device. *Procedia Engineering*, 184:145–155, 2017.

[29] OptoFidelity. Touch Panel Testing with 6-axis Robot, September 2014. Available at https://www.optofidelity.com/news/touch-panel-testing-6-axis-robot, accessed: 2.1.2021.

[30] Astech projects. Automated Drop Testing using Collaborative Robotic Technology. Available at https://www.astechprojects.co.uk/automated-drop-testing/, accessed: 3.1.2021.

[31] Python. Python 3.8.0, October 2019. Available at https://www.python.org/downloads/release/python-380/, accessed: 22.12.2020.

[32] Tapster Robotics. Tapster 2. Available at https://www.tindie.com/products/hugs/tapster-2/, accessed: 3.1.2021.

[33] Paulo Rocha, Rogério de Souza e Silva, and Maria Tostes. Prototype cnc machine design. 11 2010.

[34] CNC shop s.r.o. Katalog produktů, 2010. Available at http://cncshop.cz/katalog, accessed: 2.1.2021.

[35] STMicroelectronics. STM32Cube initialization code generator . Available at https://www.st.com/en/development-tools/stm32cubemx.html, accessed: 18.12.2020.

[36] STMicroelectronics. TrueSTUDIO, A powerful eclipse-based C/C++ integrated development tool for your STM32 projects. Available at https://www.st.com/en/development-tools/truestudio.html, accessed: 20.12.2020.

[37] STMicroelectronics. *USBLC6-2 Very low capacitance ESD protection*, October 2011. Available at https://cz.mouser.com/datasheet/2/389/usblc6_2-1852789.pdf.

[38] STMicroelectronics. *LFXX Very low drop voltage regulator with inhibit function*, May 2017. Available at https://cz.mouser.com/datasheet/2/389/lfxx-1849555.pdf.

[39] STMicroelectronics. *AN4488 Application note: Getting started with STM32F4xxxx MCU hardware development*, October 2018. Available at https://www.st.com/content/ccc/resource/technical/document/application_note/76/f9/c8/10/8a/33/4b/f0/DM00115714.pdf/files/DM00115714.pdf/jcr:content/translations/en.DM00115714.pdf.

[40] STMicroelectronics. *RM0368 reference manual*, December 2018. Available at https://www.st.com/content/ccc/resource/technical/document/reference_manual/5d/b1/ef/b2/a1/66/40/80/DM00096844.pdf/files/DM00096844.pdf/jcr:content/translations/en.DM00096844.pdf.

[41] STMicroelectronics. *STM32F401xB STM32F401xC*, April 2019. Available at https://www.st.com/resource/en/datasheet/stm32f401rc.pdf.

[42] SYS MOTOR. *17HS4417P1-X4*, November 2013. Available at https://www.sys-motor.com/Account/Plug-ins/kindeditor/attached/file/20190125/20190125074249_8241.pdf.

[43] The-engineer. CNC Basics (Building a Cnc Machine Part 1). Available at https://www.instructables.com/CNC-basics-Building-a-cnc-machine-part-1/, accessed: 2.1.2021.

[44] Toshiba. *TLP291-4 Technical Information*, 3 2015. Available at https://cz.mouser.com/datasheet/2/408/TLP291-4-1209261.pdf.

[45] YUEQING HENGWEI ELECTRONICS CO.,LTD. *Cylinder Inductive Proximity Switch Series*. Available at https://www.gme.cz/data/attachments/dsh.775-128.1.pdf.

[46] ZwickRoell. roboTest I Robotic Testing System. Available at https://www.zwickroell.com/en/automated-testing-systems/robotest-i, accessed: 3.1.2021.