

MECHATRONICS RESCUE TEAM

Team members:

DEJAN POLUTNIK, TIAN OPREŠNIK, LUKA KALIŠNIK, JURE BALON, MAKS CIZEJ,
NIK ZABUKOVNIK

Mentor:

Matej Veber

SUMMARY

We are six mechatronics students from Slovenia who have decided to participate in the international robotics competition RoboCup Rescue RMRC 2025 because we are interested in the development of mechatronics and robotics. Our modular robot is a collection of our knowledge, which we have acquired from our professors. Part of this competition is the robotic perception of the environment and the symbols of the danger of hazardous substances, which we focused on when developing this robot. First, we familiarized ourselves with all the challenges offered by the competition, and then we set about designing a rescue robot, which is fully 3D printed and meets all the requirements of the competition. In addition, we gained a lot of new knowledge, worked in a team, and gained valuable experience for our future professional careers.

Key words: computer vision, microcomputer, Python, QR code decoding, object detection, Robocup, rescue, RMRC.

TABLE OF CONTENTS

SUMMARY	2
TABLE OF PICTURES.....	4
TABLES	5
1 MECHATRONICS RESCUE TEAM	1
1.1 LOGISTIC INFORMATION	1
1.2 INTRODUCTION	1
1.2.1 TEAM MEMBERS	2
2 RESCUE ROBOT	3
2.1 HARDWARE	4
2.1.1 BASIC CONCEPT	4
2.1.2 CHASSIS.....	6
2.1.3 ROBOT ARM	8
2.2 ELECTRONICS	10
2.2.1 MICROCOMPUTER	11
2.2.2 MCP3008.....	15
2.2.4 VENTILATORS.....	18
2.2.5 VOLTAGE INDICATOR	20
2.2.6 MOTORS	21
2.3 COMPUTER VISION	23
2.3.1 CAMERA.....	24
2.3.2 PROGRAMMING COMPUTER VISION	26
2.3.3 CONTROLLING THE RESCUE ROBOT WITH VR GLASSES	42
2.4 SOFTWARE.....	47
3 LIST OF BOUGHT COMPONENTS:.....	58
4 FUTURE PLANS	59
5 CONCLUSION.....	61

TABLE OF PICTURES

Figure 1: Past years rescue robot.....	5
Figure 2: This years rescue robot	4
Figure 4: CAD-model of Rescue Robot 2	5
Figure 3: CAD-model of Rescue Robot	5
Figure 6: Development chassis.....	7
Figure 7: Ideal robotic arm	7
Figure 9: Developed Planetary gear	8
Figure 10: Robot gripper	9
Figure 11: Electrical shematic	10
Figure 12: Raspberry Pi 4B	11
Figure 13: RBPi Pins	12
Figure 15: Power supply scheme.....	13
Figure 14: MicroUSB power supply	13
Figure 16: Connection of power supply to Raspberry Pi	14
Figure 17: Power supply with Samsung batteries and Buck converter	14
Figure 18: MCP3008 shematic	15
Figure 19: H-Bridge shematic	16
Figure 20: H-bridge working principle.....	17
Figure 21: L293D shematic	18
Figure 22: L293D chip	18
Figure 23: Ventilator shematic	19
Figure 24: Ventilator	19
Figure 25:Voltage indicator on chassis	20
Figure 26: Dynamixle shematic.....	21
Figure 27: U2D2 MODULE.....	21
Figure 28: MQ Sensor	23
Figure 29: RPI WWCAM2.....	24
Figure 30: Camera connector	24
Figure 31: Camera connected to Raspberry Pi	25
Figure 32: Working principle of camera	25
Figure 33: Advanced Options.....	26
Figure 34: Expand Filesystem	27
Figure 35: QR code program.....	28
Figure 36: Code for reading QR code in real time	Napaka! Zaznamek ni definiran.
Figure 37: QR code program.....	29
Figure 38: Program that calls other programs in a sequence.....	30
Figure 39: Program for graphical user interface.....	31
Figure 40: Program for capturing image	31
Figure 41: Program for reading QR code	32
Figure 42: Reading QR codes with a computer.....	33
Figure 43: Cascade-Trainer-GUI.....	36
Figure 44: Program for image recognition	37
Figure 45: Object detection program 1st part.....	38

Figure 46: Bottle detection	40
Figure 47: Car detection	41
Figure 48: Detecting a plant	41
Figure 49: Python libraries	47
Figure 50: Program - 1 part	48
Figure 51: Program - 2 part	49
Figure 52: Program - 3 part	50
Figure 53: Program - 4 part	51
Figure 54: Program - 5 part	52
Figure 55: Program - 7 part	53
Figure 56: Program - 6 part	53
Figure 57: Program - 8 part	54
Figure 58: Program - 9 part	55
Figure 59: Program - 10 part	55
Figure 60: Program - 11 part	56
Figure 61: Program - 12 part	56
Figure 63: Program - 13 part	57
Figure 62: Program - 14 part	57
Figure 64: Future plans.....	Napaka! Zaznamek ni definiran.

TABLES

Table 1: Decoding with Raspberry Pi.....	34
Table 2: Decoding with computer	35

1 MECHATRONICS RESCUE TEAM

1.1 LOGISTIC INFORMATION

Team name:	Mechatronics Rescue Team
Organisation:	School Centre Celje, Secondary School of Mechanical Engineering, Mechatronics and Media
Country:	Slovenia
Mentor:	Matej Veber
Contact Person:	Matej Veber
	Phone: 0038631349481
	Email: matej.veber@sc-celje.si

1.2 INTRODUCTION

The name of our team is Mechatronics Rescue Team. We come from Slovenia's School Centre Celje, a Secondary School of Mechanical Engineering, Mechatronics and Media. Last year, our predecessors from Slovenia achieved first place in Eindhoven, Netherlands, second place in Bordeaux, France and third place in Sydney 2019. They got this far thanks to the advanced movement of the caterpillar tracks, which allowed the rescue robot to drive over challenging terrain. Building on this innovative concept, we enhanced the computer vision system, allowing the robot to better perceive its surroundings. This year, we are taking it a step further by incorporating virtual goggles to allow us to see from the robot's perspective. We have fixed all the weaknesses of last year's robot, added more powerful motors to its arm, and upgraded to a new aluminum alloy gripper for improved performance.

1.2.1 TEAM MEMBERS



Nik Zabukovnik,
assembly



Jure Balon,
programming



Maks Cizej,
wiring



Jaka Kališnik,
programming



Dejan Polutnik,
CAD modelling



Tian Oprešnik,
CAD modelling

2 RESCUE ROBOT

Our predecessors, who competed last year, faced significant challenges with the torque of the motors driving the wheels, making it difficult to overcome obstacles. As a result, they had to rely on their 3D-printed robotic arm for support, which eventually led to the gripper breaking under the robot's weight. To address this issue, we decided to upgrade to more powerful motors this year. We chose the DYNAMIXEL XM-430-W-350-R, which provides 4.1 Nm of torque, compared to last year's AX18-A, which had only 1.8 Nm of torque. Additionally, we added a planetary gear reducer on the third axle to achieve partial self-locking and increase the torque. Previously, the third axle was unable to support any load because the transmission from the servomotor was direct.

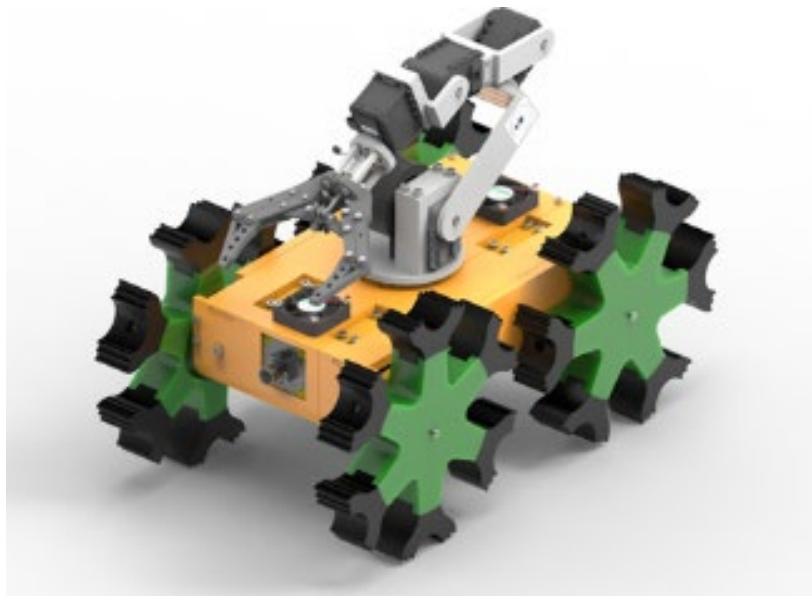


Figure 1: Past years rescue robot

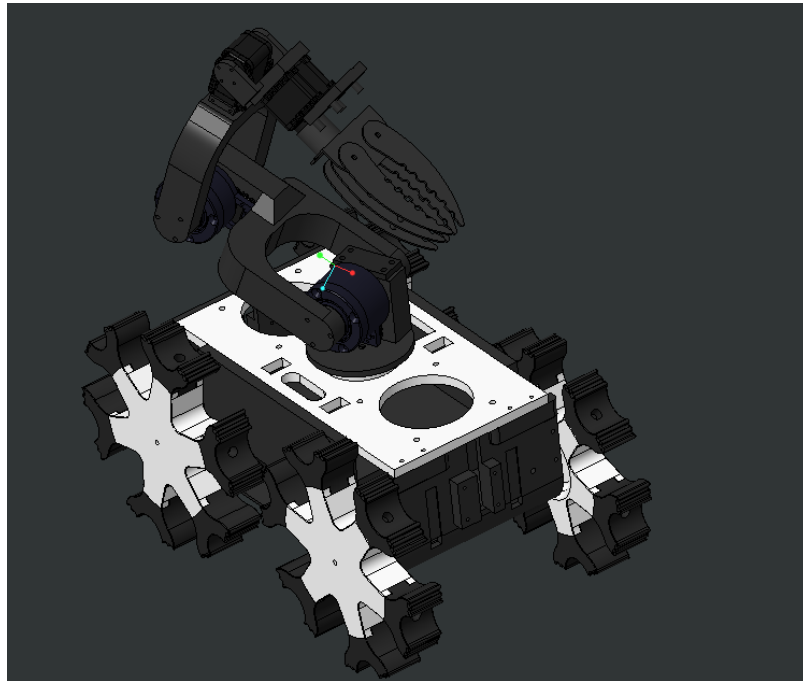


Figure 2: This years rescue robot

The main goal was to make the robot fully modular, allowing for quick and easy component replacements. We designed the entire structure with this in mind. Our first version was generally 3D printed, with only non-printable components—such as the circuit board, motors, and other essential parts—being separately integrated. The entire robot was designed in Creo Parametric and Solidworks Electrical. The black-and-white color scheme was chosen to give the robot a modern, innovative, and robust industrial look.

2.1 HARDWARE

2.1.1 BASIC CONCEPT

After learning from our predecessors, we decided that the best approach was to base our robot on special wheels.

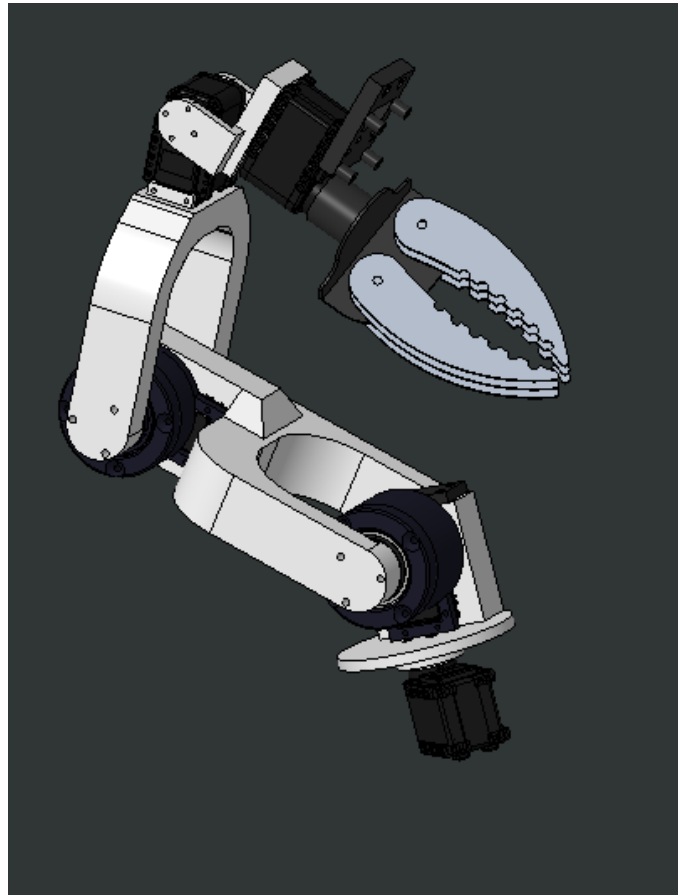


Figure 5: CAD- assembly of robotic arm

2.1.2 CHASSIS

We created an extended version of last year's chassis, making it slightly taller and incorporating two levels. On the bottom, there are four motors, whose main function is to move the rescue robot around the terrain. The first chassis design is shown in the picture below.

However, during testing, we encountered a major issue; the robot could not overcome obstacles taller than 10 cm and would get stuck due to insufficient elevation. We solved this problem by angling the four motors slightly to increase elevation. Our updated design is shown in the bottom picture.

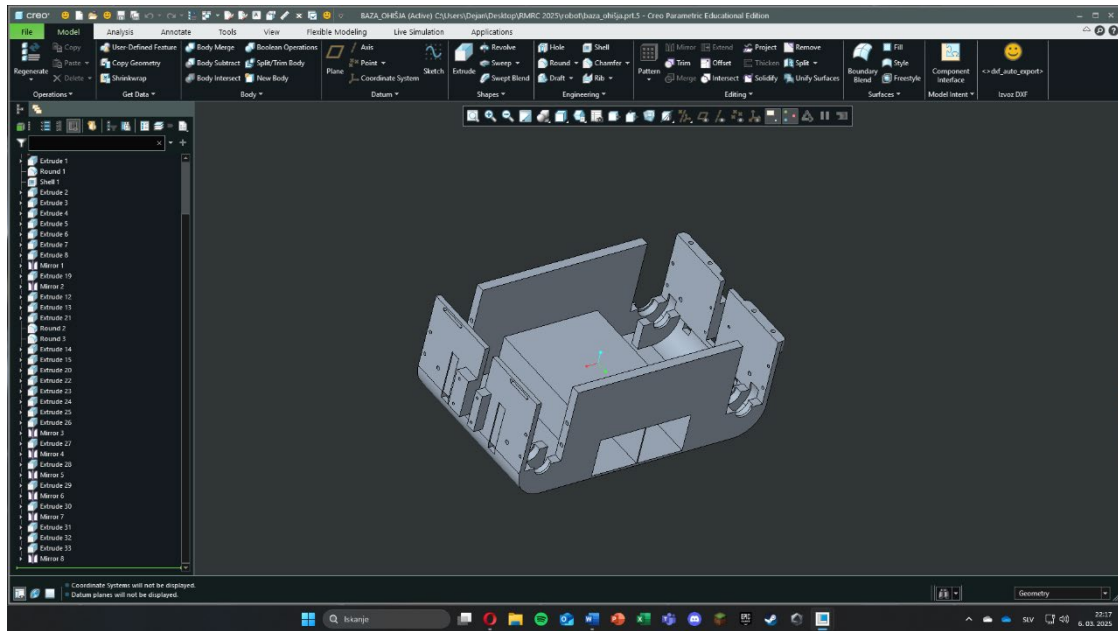


Figure 5: Development chassis

However, in solving one problem, we encountered another. Angling the motors caused them to be misaligned with the axis, meaning the circumference of the tracks changed as they moved in a parallelogram motion, causing them to lose tension. To fix this, we added a tensioner to keep the caterpillar tracks properly tightened. Tensioners are shown in the picture below.

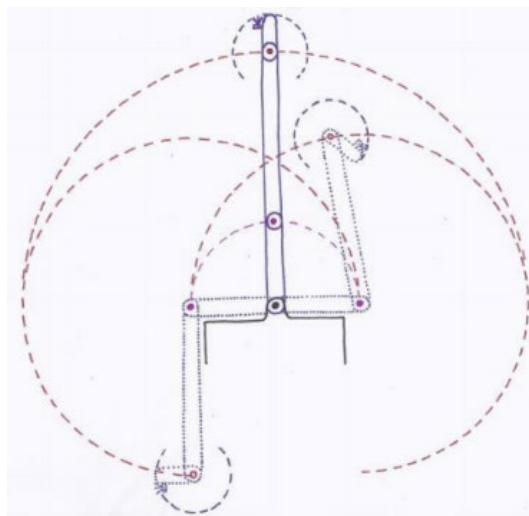


Figure 6: Ideal robotic arm

2.1.3 ROBOT ARM

The primary purpose of the robotic arm is to open hatches, valves, and small doors, as well as to hold a camera allowing the operator to monitor the robot's surroundings on a screen. However, the problem with having a camera on the arm is that it is difficult to see the surroundings because it shakes too much. A possible concept for an ideal robotic arm is shown in the picture below.

Our goal was to build a robot arm that is highly flexible and robust at the same time. We were striving to create a robotic arm similar to the one shown in the previous picture (ideal robotic arm). However, we were unable to position the arm in the middle of the surface of the chassis because the upper part was interchangeable (battery part). In the end, we developed this construction of the robotic arm.

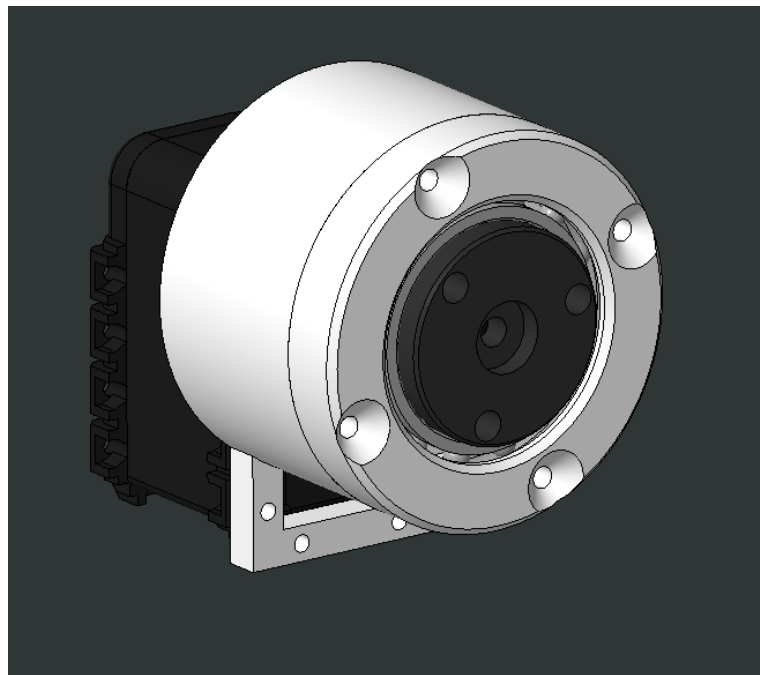


Figure 8: Developed Planetary gear

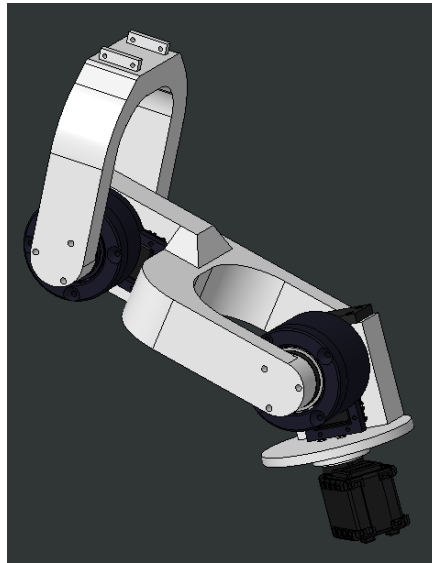


Figure 9: CAD-model

The robotic arm is attached to the chassis and features five axes. We used DYNAMIXEL XM-430-W-350-R. The first axis has a worm gear mounted on it. This motor moves the entire arm up and down, meaning it needs to be strong and generate high torque. We chose a worm gear because it effectively holds the arm in position without putting unnecessary stress on the motor. The only drawback is that the arm does not move very quickly, but this is also an advantage because it is more precise. The second motor extends the arm and also controls the third axis, allowing it to rotate the gripper 360 degrees. At the end of the arm, we have a gripper.

We purchased our gripper on Amazon, and it is called Perfeclan G6 DIY Metal. We selected this gripper because it provides strong grip strength and was ideal for our needs. The only disadvantage is that it uses a DC motor, which means we cannot determine its exact position, making operation slightly more complex.



Figure 7: Robot gripper

2.2 ELECTRONICS

A crucial part of our robot is a custom-designed circuit board. We started by sketching the circuit on a piece of paper, then we gathered all the necessary components and began assembling it. We did not use any software for designing the electronics. The main reason for this decision was to demonstrate that it is possible to make an electronic circuit without relying on advanced computer programs.

The following picture shows the circuit board of our rescue robot.

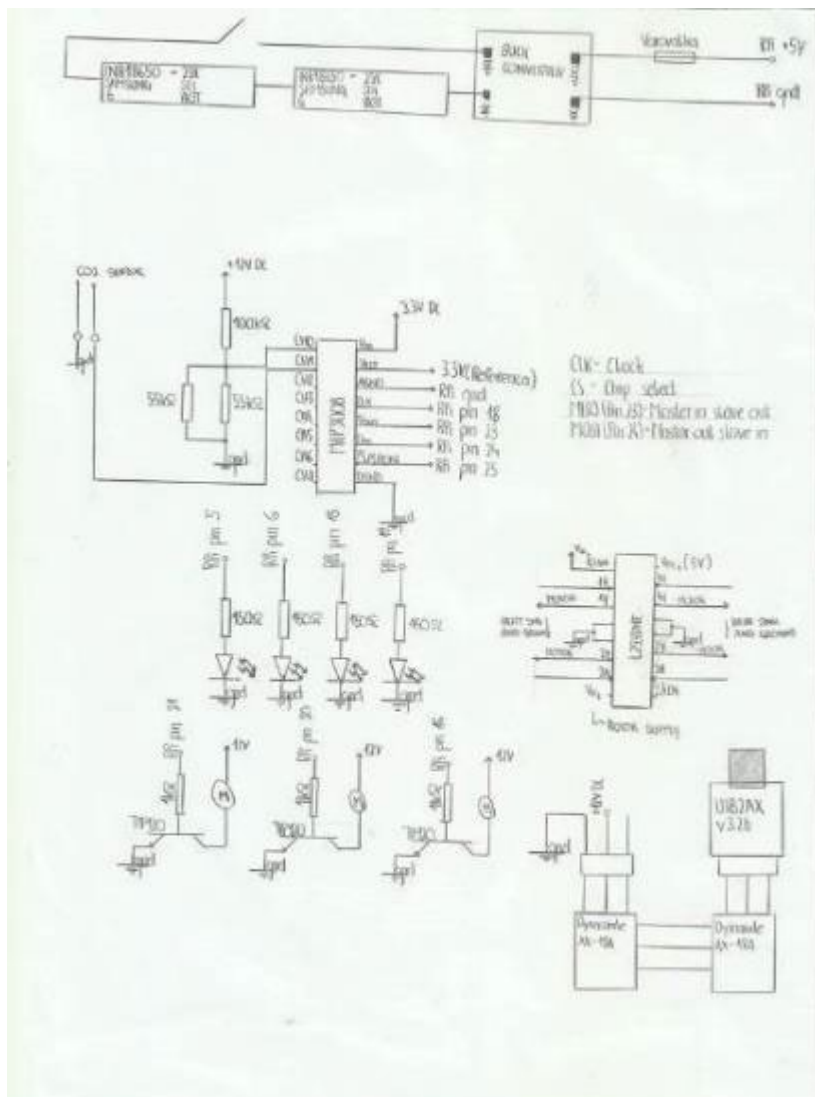


Figure 8: Electrical schematic

2.2.1 MICROCOMPUTER

We chose the Raspberry Pi, a credit card-sized microcomputer. So far, two versions have been released: Model A and Model B. Model A does not have an Ethernet connection has only one USB port, and consumes 300 mA of current, making it a more affordable option than Model B. For our project, we chose Raspberry Pi 3 Model B+, mainly because it has more USB connectors and a dedicated camera connector, which is essential for computer vision. Additionally, the built-in Wi-Fi module allows us to remotely connect to the microcomputer. Raspberry Pi uses the Raspbian operating system.

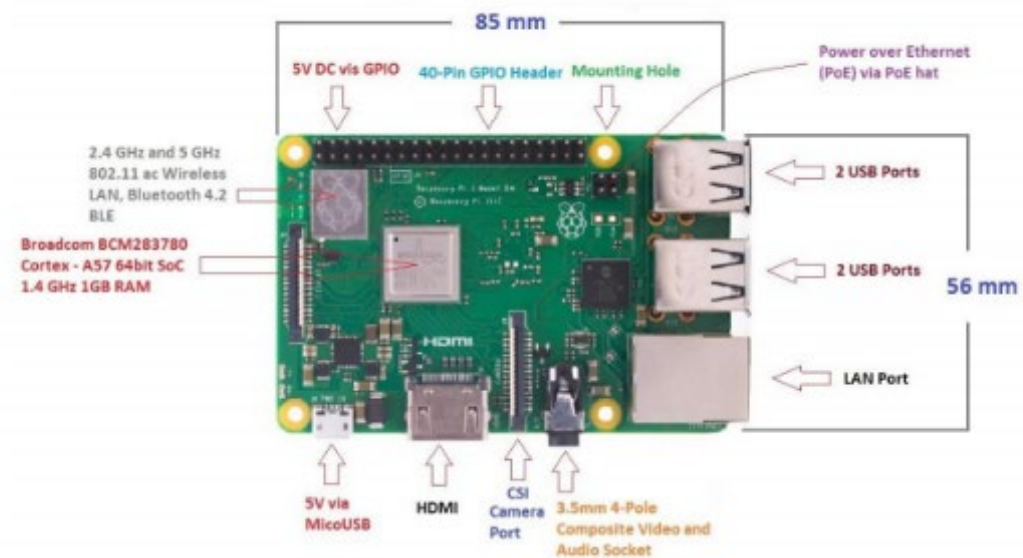


Figure 9: Raspberry Pi 4B

The microcomputer also features a GPIO (General Purpose Input/Output) interface, which enables us to read inputs and control outputs such as transistors, relays, and other components.

GPIO pins allow the user to control devices using the I2C protocol, UART, and SPI.





















Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I2C)		DC Power 5v	04
05	GPIO03 (SCL1 , I2C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I2C ID EEPROM)		(I2C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Figure 10: RBPi Pins

When connecting inputs and outputs to the microcomputer, we must carefully follow the schematic above, which shows the GPIO pin layout. It is important to avoid short contact, as they can cause the Raspberry Pi to malfunction. When we were making a circuit board, we encountered this exact issue. The software functioned perfectly, but we were unable to toggle the outputs on and off, despite sending commands from the Raspberry Pi.

2.2.1.1 Power supply

The microcomputer is powered through a standard 5V MicroUSB connection, which can receive power either from a computer's USB output or via a USB interface connected to a household power source. The Raspberry Pi requires at least 700 mA of current for normal operation.

For testing purposes, we primarily used this MicroUSB power supply.



Figure 16: MicroUSB power supply

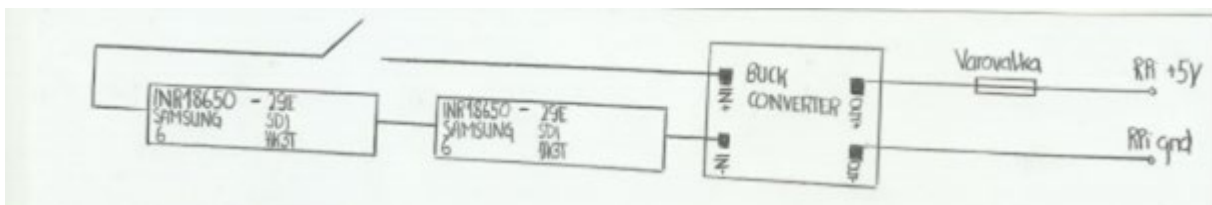


Figure 13: Power supply scheme



Figure 19: Connection of power supply to Raspberry Pi

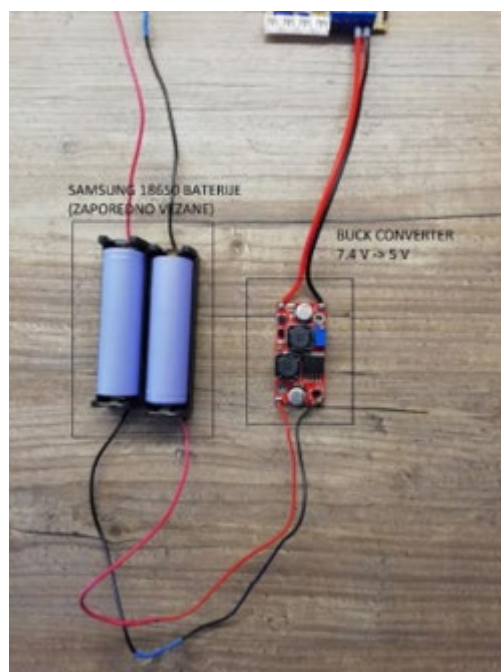


Figure 20: Power supply with Samsung batteries and Buck converter

Raspberry Pi can also be powered via GPIO (General Purpose Input/Output interface). To do this, we connect 5V to pin 2 and ground (0V) to pin 6. On our rescue robot, we used this kind of configuration. The following pictures show how to connect a power supply to the Raspberry Pi.

We used two Samsung 18650 batteries to supply power. These are connected in series, providing a total of 7.4V (since each battery supplies 3.7V). We then reduced the voltage to 5V using a buck converter. Since the input power must theoretically equal the output power, reducing the voltage results in an increase in current.

2.2.2 MCP3008

Raspberry Pi does not have a built-in ADC (analog to digital converter), meaning it cannot read analog values directly. To solve this, we built our own ADC using the MCP3008 chip. We chose MCP3008 because it has extensive library support, making our work as programmers much easier.

We need to read analog values from a CO₂ sensor and using it is the only way to achieve this.

The picture below illustrates how to connect the MCP3008.

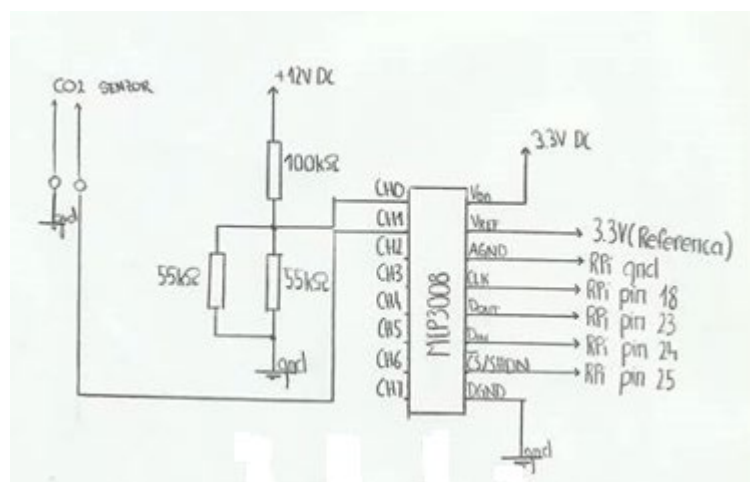


Figure 23: MCP3008 schematic

CLK = CLOCK
 CS = CHIP SELECT
 MISO (Pin 23) = MASTER IN SLAVE OUT
 MOSI (Pin 24) = MASTER OUT SLAVE IN

This chip operates using SPI (Serial Peripheral Interface) communication, which is why it has CLK, CS, MISO, and MOSI pins. These pins enable data exchange between the chip and the microcomputer.

When the microcomputer needs information, it sends a request to the MCP3008 chip. The chip then processes the request and responds with values ranging from 0 to 1023 (10-bit resolution).

2.2.3 L293D

This is an H-Bridge chip, which allows us to control the motor's direction. Although we could build an H-Bridge using four transistors (two NPN and two PNP), we opted for the chip version due to limited space on our solder board.

The basic schematic of the H-bridge is shown in the picture below.

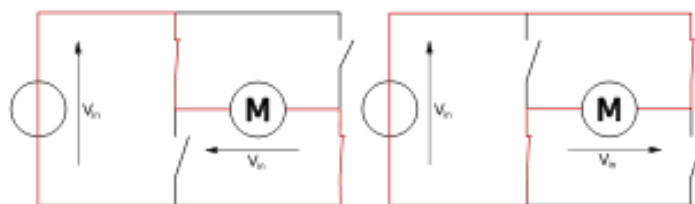


Figure 24: H-Bridge schematic

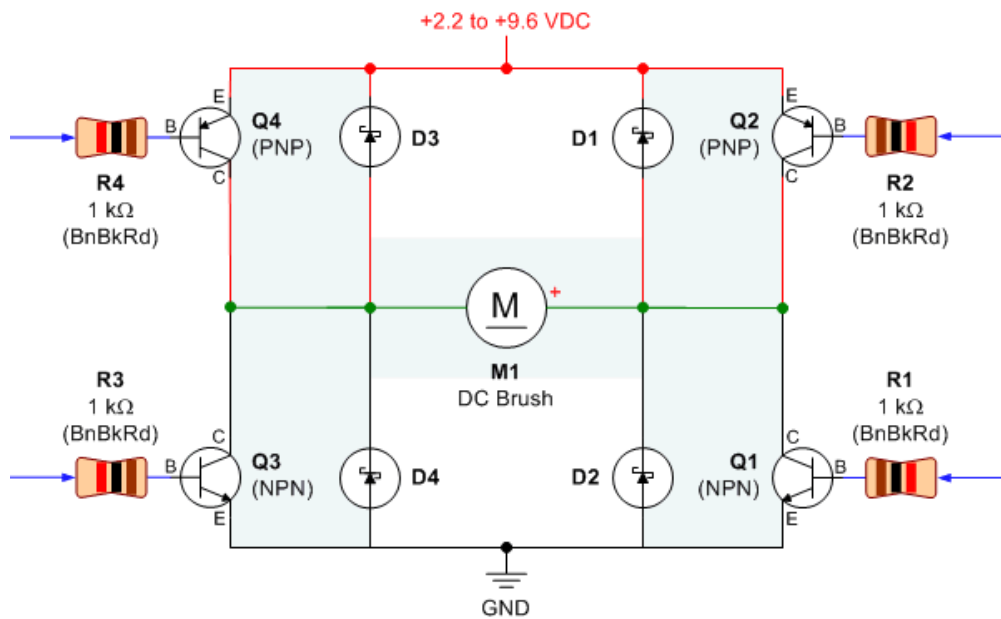


Figure 25: H-bridge working principle

This chip has an integrated H-Bridge circuit, so we only need to connect it properly to use it.

We need the H-Bridge to control the DC motor on the gripper. This motor does not consume a lot of current; therefore, this chip (L293D) is suitable for our needs.

The picture below shows how we connected the L293D chip.

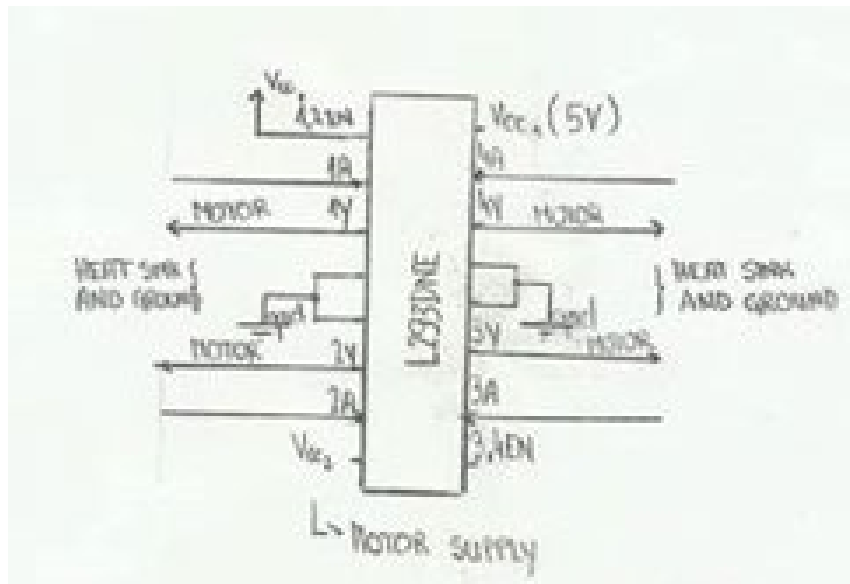


Figure 26: L293D schematic

(Personal archive)



Figure 29: L293D chip

2.2.4 VENTILATORS

We also added three transistors to our circuit to control the ventilators, which play a crucial role in cooling the microcomputer and other electronics. We used three TIP120 Darlington transistors because they already have built-in protection diodes. They protect transistors from voltage spikes that occur when the transistor is turned off. A motor acts as an inductor, meaning it resists sudden changes in current. A motor acts as an inductor, meaning it resists

sudden changes in current. When the current changes, it induces a voltage, creating a high-voltage spike that can potentially damage the transistor. This is why protection diodes are essential in circuits involving motors. The schematic for the ventilator circuit is shown in the picture below.

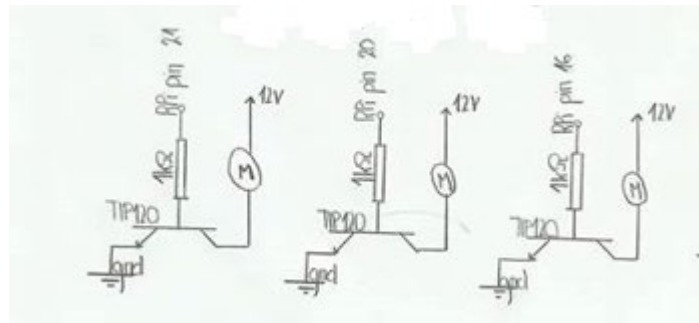


Figure 30: Ventilator schematic

We chose the smallest ventilators that we could find, mainly due to limited space within our chassis.



Figure 33: Ventilator

On our robot, we used only two ventilators, but we added one additional transistor in case we decide to add a third ventilator later. The following picture shows the placement of our ventilator, with another one positioned on the opposite side of the chassis.

2.2.5 VOLTAGE INDICATOR

On our robot, we also used four LED diodes to indicate the voltage level of our 12V battery. This circuit is simple, utilizing four LED diodes and four resistors, whose values are shown in the following picture.

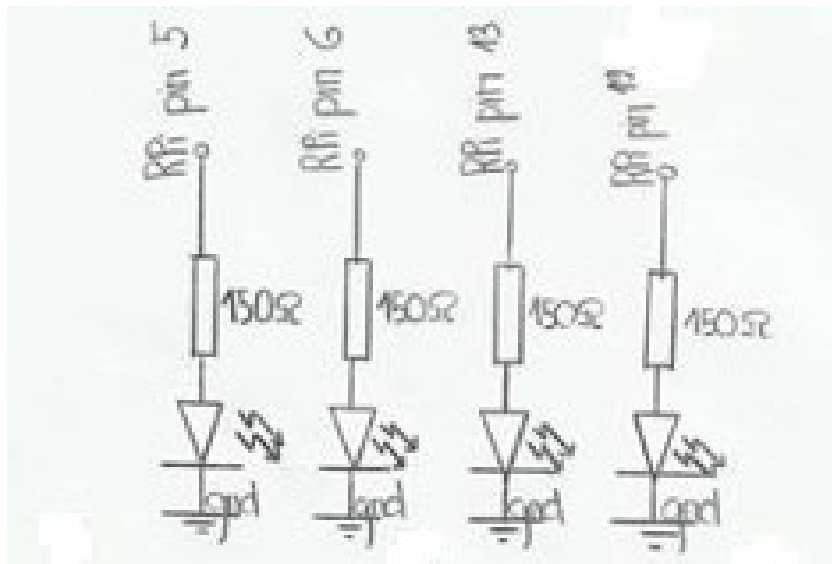


Figure 36: Voltage indicator on chassis

2.2.6 MOTORS

For our robot, we used the Dynamixel XM-430-W-350-R and Dynamixel AX-18A motors because they provide high torque and speed. These are smart actuators that communicate with the microcomputer via the I2C interface. To use them, we purchased a USB to USB2AX connector which enables seamless communication. The XM-430-W-350-R motors were used to drive the robot's wheels, while the AX-18A motors were used to control the robotic arm.

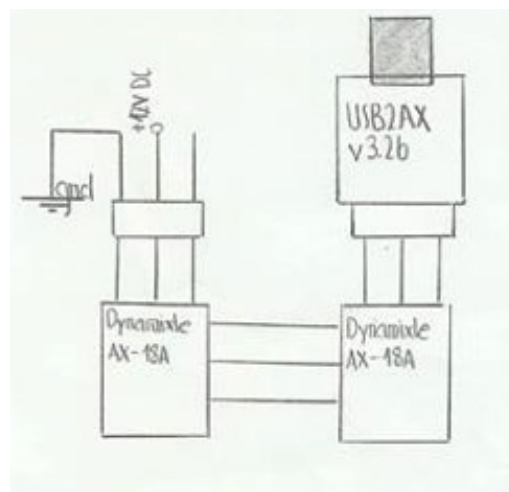


Figure 39: Dynamixel schematic



Figure 40: U2D2 MODULE



Figure 28: Dynamixel XM430-W350-R

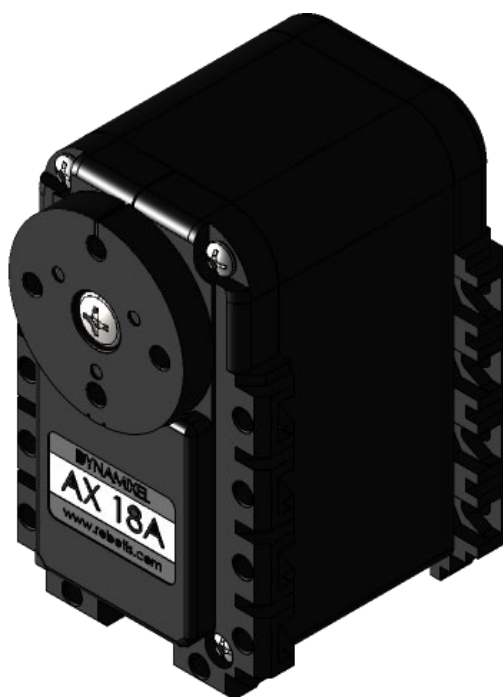


Figure 29: Dynamixel AX-18A

2.2.7 CO₂ Sensor

One of the competition tasks is to measure CO₂ levels in the air. We selected a cost-effective sensor that meets our requirements—the MQ sensor, which provides both analog and digital output to the microcomputer. Although we expected this sensor to be accurate, it turned out to be unreliable in practice. Despite its limitations, we still decided to use it in our robot.

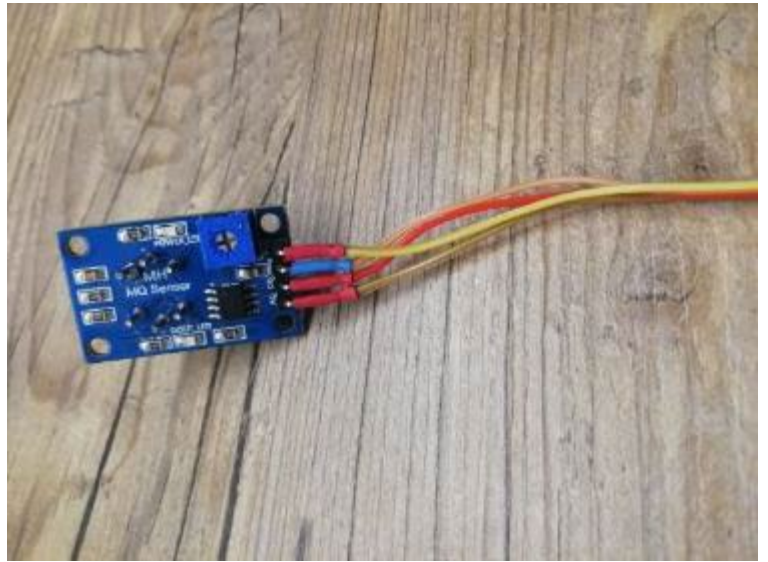


Figure 30: MQ Sensor

2.3 COMPUTER VISION

As mentioned earlier, we also integrated computer vision. This was the first task we set for ourselves, knowing it would take a lot of time, especially since none of our team members had prior experience programming in Python.

2.3.1 CAMERA

We chose a camera specifically designed for the Raspberry Pi microcomputer. The camera model we selected is RPI WWCAM2.



Figure 31: RPI WWCAM2

In order to use the camera, we had to properly connect it to the Raspberry Pi.



Figure 32: Camera connector

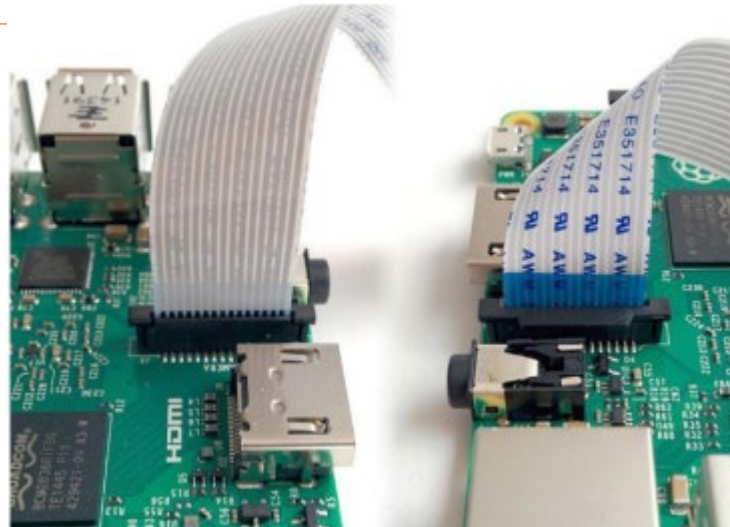


Figure .33: Camera connected to the Raspberry Pi

2.3.1.1 How does a camera work

During operation, the camera continuously captures images and sends them to the GPU (Graphics Processing Unit), even if the program is not actively using the camera. After startup, the program must wait for a second to allow the camera to start sending data. This delay enables the sensor to adjust to light, ensuring that the captured images are clear. We can think of the sensor as a matrix consisting of counters (sensor units) that count the quantity of red, green, and blue.

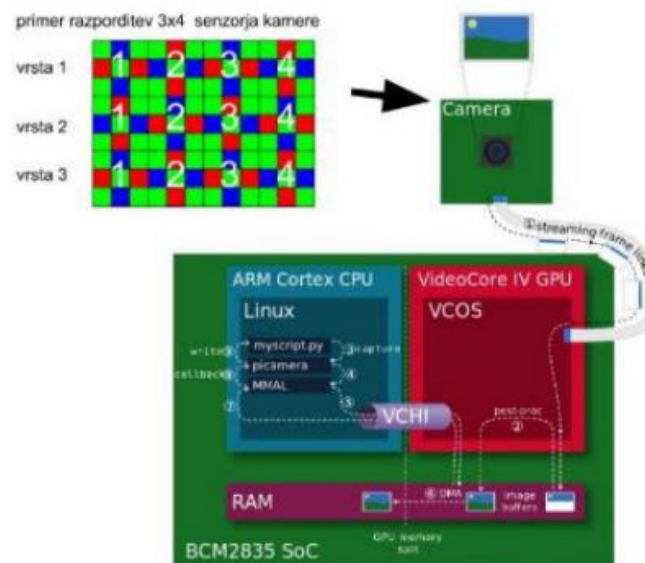


Figure 34: Working principle of camera

2.3.2 PROGRAMMING COMPUTER VISION

2.3.2.1 QR Code

To decode the QR code, we used Python 3, along with the OpenCV and imutils libraries, which provide various functions for image processing. Additionally, we used the Pyzbar library, which allows us to read one-dimensional barcodes and QR codes. Since these libraries are not pre-installed in the Raspbian operating system used by the Raspberry Pi, we had to install them manually. Before installing the OpenCV library, it is important to note that the installation requires a significant amount of space on the SD card. Therefore, it is recommended to use an SD card with at least 16 GB of storage. Installation is done through the terminal, so all commands that begin with the "\$" character should be typed in the terminal without the "\$" character. To ensure that the SD card has access to all available space, we first need to expand the file system: `$ sudo raspi-config`. Then, we chose "Advanced Options".

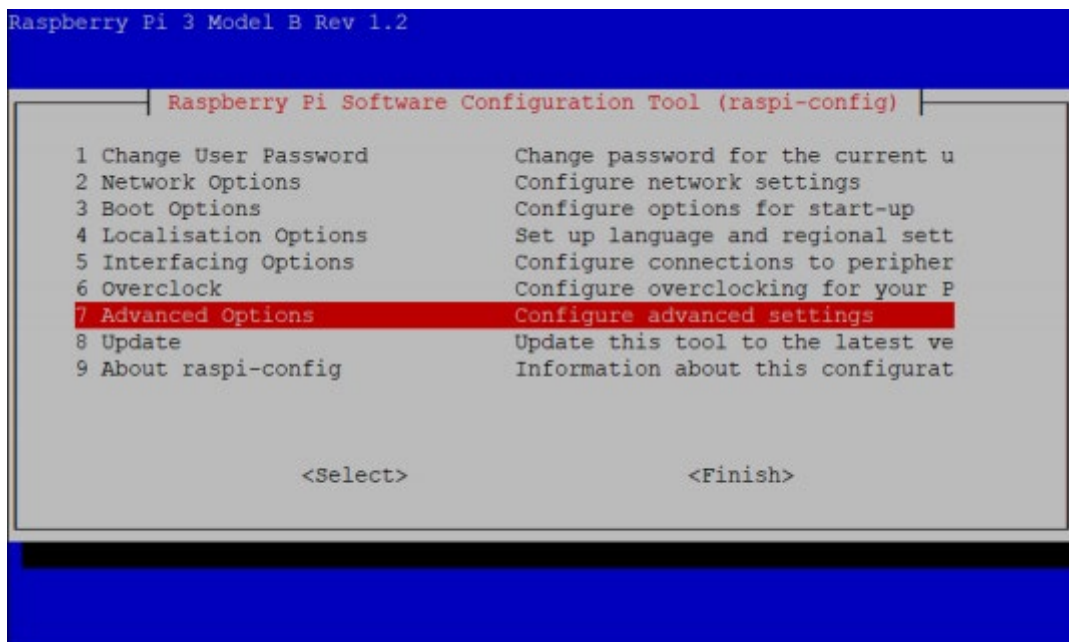


Figure 35: Advanced Options

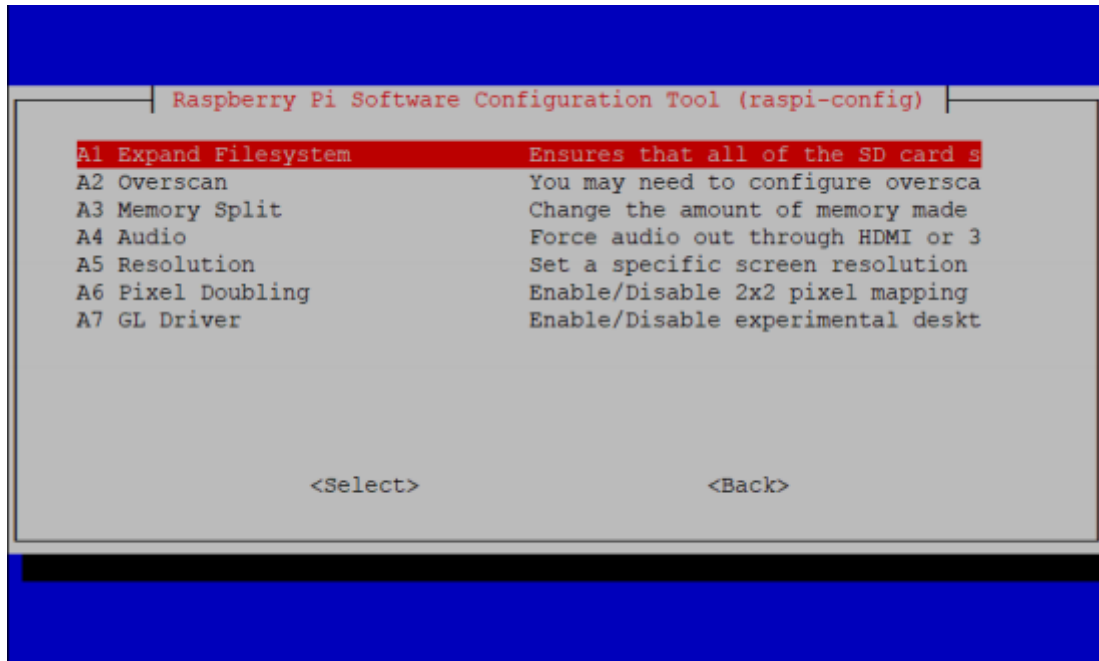


Figure 36: Expand Filesystem

2.3.2.1.1 Program for reading QR code in real time

The following picture shows the Python program for reading QR code in real time:

```

# uvoz potrebnih knjižnic
from imutils.video import VideoStream
from pyzbar import pyzbar
import argparse
import datetime
import imutils
import time
import cv2

# konstrukcija razčlenjevalnika argumentov in razčlenitev argumentov
ap = argparse.ArgumentParser()
ap.add_argument("o", "--output", type=str, default="barcodes.csv",
                help="path to output CSV file containing barcodes")
args = vars(ap.parse_args())

# inicializacija video prenosa
print("[INFO] starting video stream...")
# vs = VideoStream(src=0).start()
vs = VideoStream(usePiCamera=True).start()
time.sleep(2.0)

# odpiranje CSV dokumenta v katerega se bodo shranile najdene kode QR
csv = open(args["output"], "w")
found = set()

# glavna zanka
while True:
    # spreminitev velikosti videa tako, da je maksimalna širina videa
    # 400 slikovnih pik
    frame = vs.read()
    frame = imutils.resize(frame, width=400)

    # iskanje kod v vsaki sliki videa in dešifriranje vseh kod
    barcodes = pyzbar.decode(frame)

    # loop over the detected barcodes
    for barcode in barcodes:
        # risanje kvadratov okoli zaznanih kod QR na sliki
        (x, y, w, h) = barcode.rect
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)

        barcodeData = barcode.data.decode("utf-8")
        barcodeType = barcode.type

        # pisanje besedila, ki ga predstavlja posamezna koda QR
        text = "{} ({}).format(barcodeData, barcodeType)
        cv2.putText(frame, text, (x, y - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)

        # če besedila kode še ni v CSV datoteki, zapišemo
        # časovni žig + kodo na disk
        if barcodeData not in found:
            csv.write("{}{}\n".format(datetime.datetime.now(),
                                     barcodeData))
            csv.flush()
            found.add(barcodeData)

        # prikaz izhodne slike
        cv2.imshow("Barcode Scanner", frame)
        key = cv2.waitKey(1) & 0xFF

        # če pritisnemo tipko "q" se zanka prekine
        if key == ord("q"):
            break

# zaprtje CSV datoteke
print("[INFO] cleaning up...")
csv.close()
cv2.destroyAllWindows()
vs.stop()

```

Figure 37: QR code program

2.3.2.1.2 Program for reading QR codes with a computer

Since the Raspberry Pi has limited processing power, reading QR codes and displaying live images simultaneously can cause performance slowdowns. That is why we decided to read the QR code in a way that the Raspberry Pi would be relieved. The QR code reading process is structured as follows:

1. Raspberry Pi captures an image using its camera and sends it to the PC via RPi-Cam-WebInterface.
2. The computer displays the received image in a web browser.
3. The person operating the robot starts the QR code reader on the computer.
4. A countdown timer is displayed, indicating when the computer will capture the screen. The image is then saved and decoded.

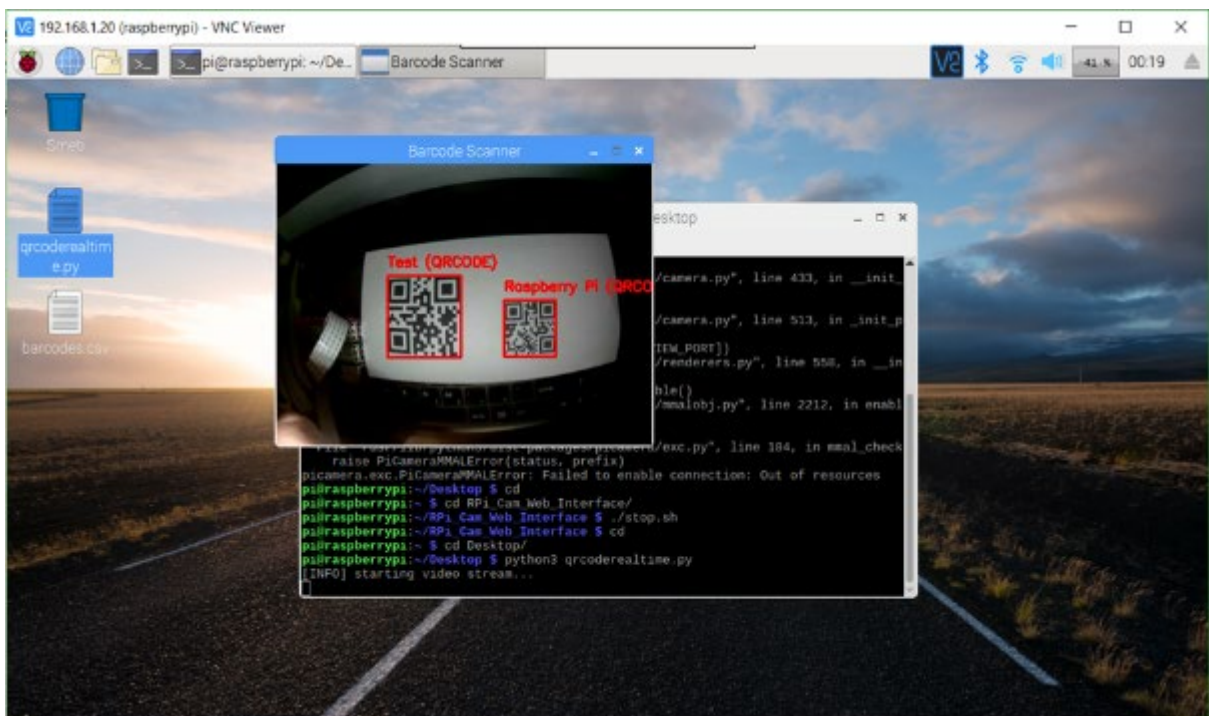


Figure 38: QR code program

Since we will run the following programs on a computer, we needed to install the OpenCV library, imutils, pyzbar, and PyAutoGUI, which allow us to capture the screen. Additionally, we needed the Tkinter library to create a graphical user interface. However, since Tkinter is included by default, we did not need to install it manually. All other libraries can be installed via the terminal using the command.

Once the necessary libraries are installed, we can run the program shown in the following picture.

This program starts the timer program. Then, we wait for 3 seconds, and it starts the screen capture program. Finally, it opens the captured screen image and the program for decoding the QR code.

```
import sc #uvoz knjižnice za zajemanje zaslona
import os #uvoz knjižnice za zagon programa, ki dešifrira kodo QR
import time #uvoz knjižnice za uporabo časa
from counter import counter #uvoz knjižnice za prikaz časovnega števca

counter() #prikaz časovnega števca

time.sleep(3) #čakanje

sc.capture() #zajem zaslona

os.system("qrsc.py --image qr.png") #zagon programa, ki dešifrira kodo QR in
#nalaganje zajete slike zaslona
```

Figure 39: Program that calls other programs in a sequence

```
import tkinter as tk #uvoz knjižnice, ki nam omogoča ustvariti
                        #grafični uporabniški vmesnik
import time #uvoz knjižnice za uporabo časa

root = tk.Tk()
time_str = tk.StringVar()
def count_down(): #odštevalnik

    for t in range(3, -1, -1):

        time_str.set(t)
        root.update()
        time.sleep(1)

def counter(): #izris okna z odštevalnikom

    label_font = ('helvetica', 40) #izbira pisave
    tk.Label(root, textvariable=time_str, font=label_font, bg='white',
             fg='blue', relief='raised', bd=3).pack(fill='x', padx=5, pady=5)

    count_down() #izvedba odštevanja
    root.destroy() #zapiranje grafičnega uporabniškega vmesnika
    root.mainloop()
```

Figure 40: Program for graphical user interface

```
# uvoz potrebnih knjižnic
import numpy as np
import pyautogui
import imutils
import cv2

def capture():
    image = pyautogui.screenshot() # zajem slike in shranitev slike v spomin

    image = cv2.cvtColor(np.array(image), cv2.COLOR_RGB2BGR) #pretvorba slike PIL/Pillow
                                                            #v NumPy "array", katerega
                                                            #podpira knjižnica OpenCV

    img = image [89:1030, 122:1798]
    cv2.imwrite("qr.png", img) # zapis slike na disk
    cv2.waitKey(0)
```

Figure 41: Program for capturing image

```
# uvoz potrebnih knjižnic
from pyzbar import pyzbar
import argparse
import cv2

# konstrukcija razčlenjevalnika argumentov in razčlenitev argumentov
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
                help="path to input image")
args = vars(ap.parse_args())

# uvoz slike
image = cv2.imread(args["image"])
image = cv2.resize(image, (960, 540))
# iskanje kod QR na sliki ter dešifriranje vseh kod
barcodes = pyzbar.decode(image)

# glavna zanka
for barcode in barcodes:
    # risanje kvadratov okoli zaznanih kod QR na sliki
    (x, y, w, h) = barcode.rect
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 0, 255), 2)

    barcodeData = barcode.data.decode("utf-8")
    barcodeType = barcode.type

    # pisanje besedila, ki ga predstavlja posamezna koda QR
    text = "{} ({}).format(barcodeData, barcodeType)
    cv2.putText(image, text, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX,
                0.5, (0, 0, 255), 2)

    # izpis tipa kode in podatkov v terminal
    print("[INFO] Found {} barcode: {}".format(barcodeType, barcodeData))

# prikaz slike z dešifriranimi kodami
cv2.imshow("Image", image)
cv2.waitKey(0)
```

Figure 42: Program for reading QR code



Figure 43: Reading QR codes with a computer

Reading the QR code using only Raspberry Pi is possible and useful. The problem occurs if we want to use the camera for another task at the same time. The best solution for this is to read the QR code with a computer since it relieves the Raspberry Pi. However, we could not use this method because the competition demands reading QR codes in real time.

Still, we made some measurements at different levels of illumination, using different decoding methods (with a computer and Raspberry Pi), and at varying distances. We used a smartphone to measure the illumination.

Illumination [lx]	Distance [cm]	Was the QR code decoded?
15	10	Yes
15	20	Yes
15	30	No
15	40	No
15	50	No
15	60	No
100	10	Yes
100	20	Yes
100	30	Yes
100	40	No
100	50	No
100	60	No
300	10	Yes
300	20	Yes
300	30	Yes
300	40	No
300	50	No
300	60	No

Table 1: Decoding with Raspberry Pi

Illumination [lx]	Distance [cm]	Was the QR code decoded?
15	10	Yes
15	20	Yes
15	30	Yes
15	40	Yes
15	50	Yes
15	60	No
100	10	Yes
100	20	Yes
100	30	Yes
100	40	Yes
100	50	Yes
100	60	Yes
300	10	Yes
300	20	Yes
300	30	Yes
300	40	Yes
300	50	Yes
300	60	Yes

Table 2: Decoding with computer

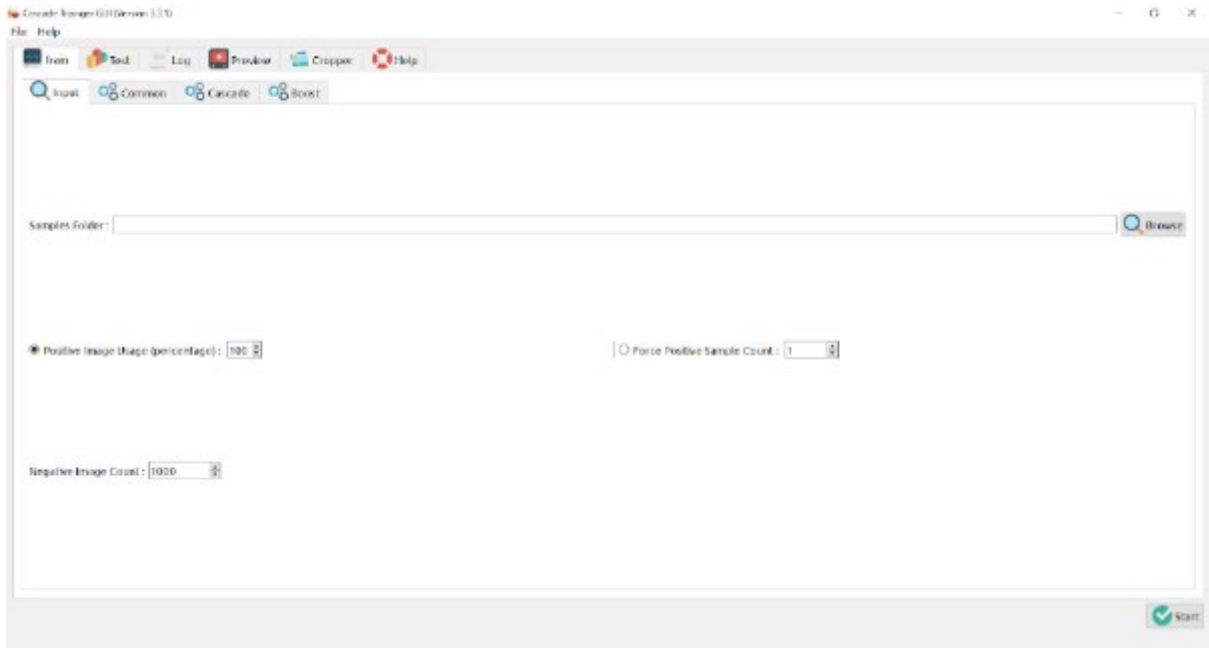


Figure 44: Cascade-Trainer-GUI

2.3.2.2 HAZMAT SYMBOLS

To detect hazmat signs, we trained our own Haar cascade classifiers using Cascade-Trainer-GUI.

To train the classifier, we need two sets of samples: positive samples, which are images that contain the object we want to detect, and negative samples, which are images that do not contain the object. Additionally, all pictures must be downsized to 50×50 pixels. Initially, we created positive samples manually by capturing hazmat signs with a camera, then downsizing and cropping the images to include only the signs. This process was time-consuming, so we decided to automate it. We developed a program that takes a single image of a hazmat sign and generates multiple variations by applying downsizing, random rotation, perspective transformation, brightness adjustments, and an overlay of a randomly drawn rectangle with varying opacity. This distortion helps improve the robustness of object detection during training.

For negative samples, we automatically downloaded images from ImageNet, a large database used for training computer vision models. During training, we used 100 positive

samples and 1,146 negative samples. Below is the code we use for detecting hazmat signs for non-flammable gas, flammable liquid, and inhalation hazards:

```
import numpy as np
import cv2

inhh = cv2.CascadeClassifier('inhalation_hazard.xml')
flal = cv2.CascadeClassifier('flammable_liquid.xml')
ngas = cv2.CascadeClassifier('nonflammable_gas.xml')

cap = cv2.VideoCapture(0)

while 1:
    ret, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    inhalation = inhh.detectMultiScale(gray, 1.3, 5)
    liquid = flal.detectMultiScale(gray, 1.3, 5)
    nongas = ngas.detectMultiScale(gray, 1.3, 5)

    for (x,y,w,h) in inhalation:
        cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]
        cv2.putText(img, 'inhalation hazard', (x,y), cv2.FONT_HERSHEY_SIMPLEX, 1,

    for (x,y,w,h) in liquid:
        cv2.rectangle(img, (x,y), (x+w,y+h), (0,0,255),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]
        cv2.putText(img, 'flammable liquid', (x,y), cv2.FONT_HERSHEY_SIMPLEX, 1,

    for (x,y,w,h) in nongas:
        cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img[y:y+h, x:x+w]
        cv2.putText(img, 'nonflammable gas', (x,y), cv2.FONT_HERSHEY_SIMPLEX, 1,

    cv2.imshow('img',img)
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

Figure 45: Program for image recognition

2.3.2.3 OBJECT DETECTION

This is not a required task in the competition; however, we were very interested in object detection, and we are sure many others are as well.

To save time, we used a pre-trained model since training a model from scratch can take several hours. Specifically, we used the MobileNet model.

2.3.2.3.1 Program for object detection

To use the program shown in the following picture, we **need** to have the OpenCV, imutils, and NumPy libraries installed, all of which have already set up.

```
# uvoz potrebnih knjižnic
from imutils.video import VideoStream
from imutils.video import FPS
import numpy as np
import argparse
import imutils
import time
import cv2

# konstrukcija razčlenjevalnika argumentov in razčlenitev argumentov
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--prototxt", required=True,
                help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
                help="path to Caffe pre-trained model")
ap.add_argument("-c", "--confidence", type=float, default=0.2,
                help="minimum probability to filter weak detections")
args = vars(ap.parse_args())

# initalizacija seznama oznak vrst, katere je bil MobileNet naučen zaznavati
# generiranje barve pravokotnika, ki se bo izrisal ob zaznavanju objektov
CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
            "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
            "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
            "sofa", "train", "tvmonitor"]
COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))
```

Figure 46: Object detection program first part

```

# nalaganje MobileNet modela iz mikro SD kartice
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])

# initalizacija video prenosa
# initalizacija števca sličic na sekundo
print("[INFO] starting video stream...")
# vs = VideoStream(src=0).start()
vs = VideoStream(usePiCamera=True).start()
time.sleep(2.0)
fps = FPS().start()

# glavna zanka
while True:
    # sprememba velikosti videa tako, da je maksimalna širina videa
    # 400 slikovnih pik
    frame = vs.read()
    frame = imutils.resize(frame, width=400)

    # pridobitev dimenzij okvirjev
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),
                                0.007843, (300, 300), 127.5)
    net.setInput(blob)
    detections = net.forward()

    for i in np.arange(0, detections.shape[2]):
        # pridobitev verjetnosti, ki je povezana z predvidevanjem
        confidence = detections[0, 0, i, 2]

        # filtiranje šibkih detekcij s tem, da zagotovimo, da je verjetnost
        # večja od minimalne nastavljene verjetnosti
        if confidence > args["confidence"]:
            # pridobitev oznak vrst iz detekcij in izračun koordinat
            # okvirja objekta
            idx = int(detections[0, 0, i, 1])
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")

            # izris oznake, pravokotnika in izpis verjetnosti
            label = "{}: {:.2f}%".format(CLASSES[idx],
                                        confidence * 100)
            cv2.rectangle(frame, (startX, startY), (endX, endY),
                          COLORS[idx], 2)
            y = startY - 15 if startY - 15 > 15 else startY + 15
            cv2.putText(frame, label, (startX, y),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)

    # prikaz slike izhoda
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    # če pritisnemo tipko "q" se zanka prekine
    if key == ord("q"):
        break

    # osvežitev števca sličic
    fps.update()

# ustavitev časovnika in prikaz informacij o številu sličic na sekundo
fps.stop()
print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))

cv2.destroyAllWindows()
vs.stop()

```

Figure 47: Object detection program second part

To load MobileNet and start the program, we used the following command:

```
$ python3 real_time_object_detection.py\  
  
--prototxt MobileNetSSD_deploy.prototxt.txt\  
  
--model MobileNetSSD_deploy.caffemodel
```

Despite the limited frames per second and low resolution of the live broadcast image, object detection with the Raspberry Pi was still successful.

The Raspberry Pi was able to accurately recognize a person, a bottle of water, a car, and a plant.

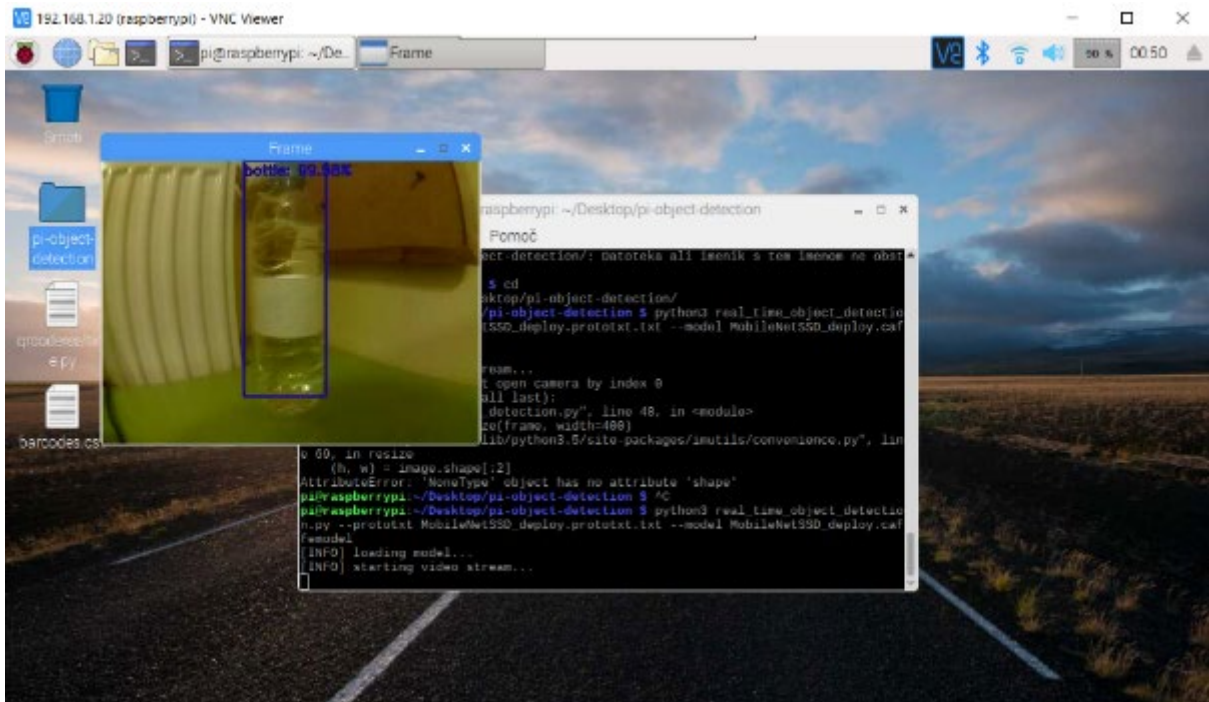


Figure 48: Bottle detection

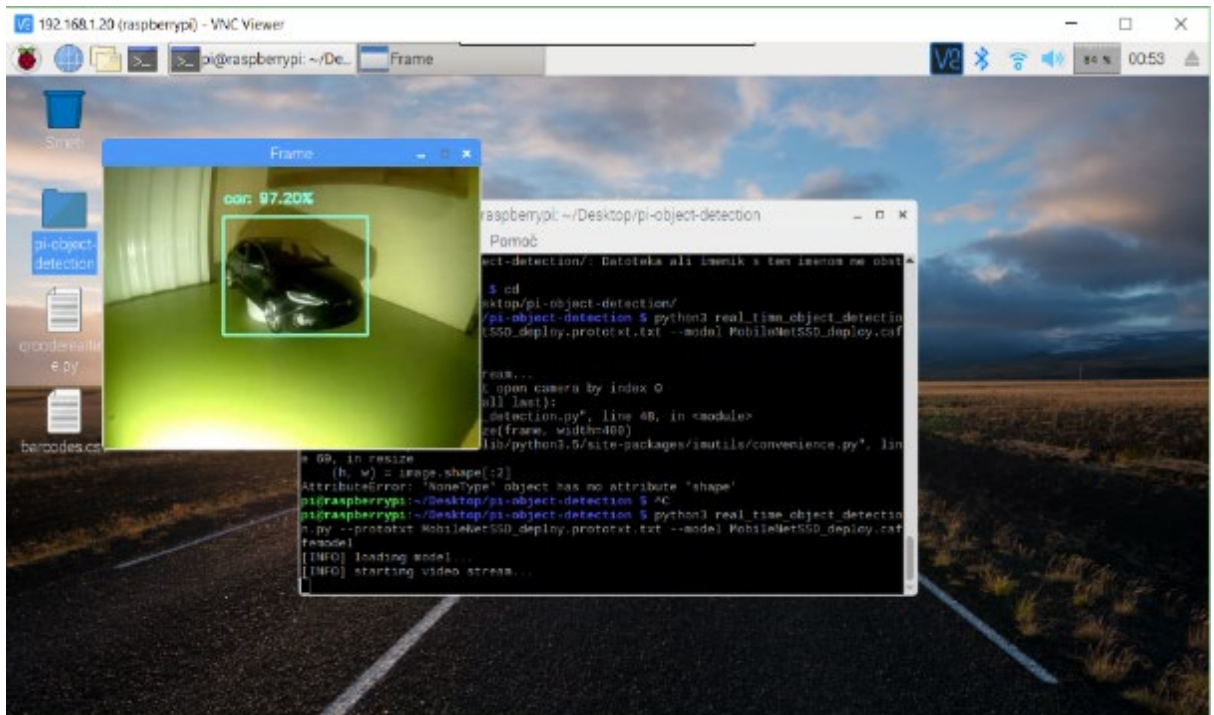


Figure 49: Car detection

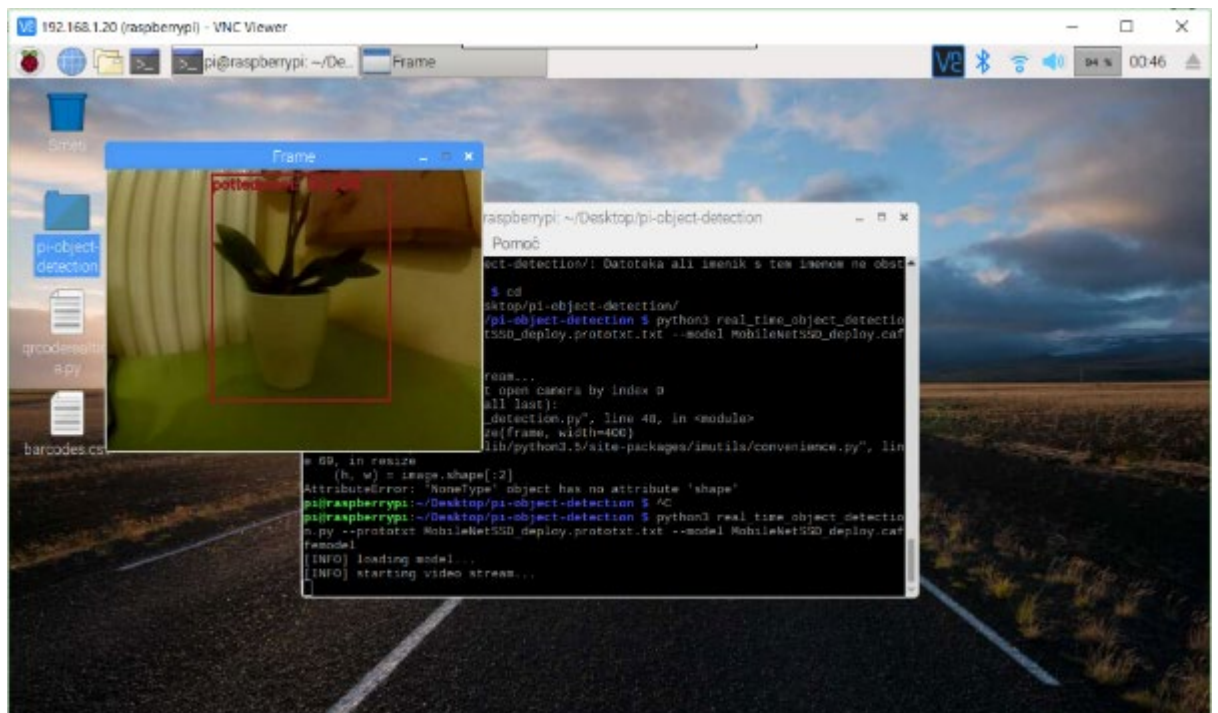


Figure 50: Detecting a plant

2.3.3 CONTROLLING THE RESCUE ROBOT WITH VR GLASSES

This year's innovation in the rescue robot is the use of VR glasses for its control. Virtual reality (VR) provides a more immersive visual experience, placing the user in a digitally created environment where they can move and interact. We tested FPV (First-Person View) control, which allows for a first-person perspective through the robot's cameras. Two different methods were used to transmit the image to the VR glasses: one with the Insta360 One X2 camera and the other with the Eachine camera. We analyzed and compared both methods for their effectiveness. This innovation has improved robot control, enhancing its adaptability in overcoming obstacles and providing a more intuitive user experience.



Figure 51: Using VR glasses

2.3.3.1 VIRTUAL REALITY GLASSES

Virtual reality (VR) glasses consist of two high-resolution screens, one for each eye, displaying slightly different images to create a 3D effect. Motion sensors track head movements, allowing users to naturally look around the virtual environment. Advanced VR headsets offer full-body tracking, enabling users to move and interact with virtual objects.

Cheaper VR glasses require a connection to a computer or gaming console for processing power, while high-end models have built-in processors for standalone use. Many VR systems include controllers for interacting with virtual objects or simulating hand movements. Leading VR headset manufacturers include Meta, Sony, Apple, Valve, and HTC, each offering different features to cater to different user needs. One of the most popular headsets is the Meta Quest 2, developed by Meta Platforms in 2020. This standalone device is priced affordably at around €200, making virtual reality more accessible to a wider audience.



Figure 52: Meta Quest 2

2.3.3.2 EACHINE CAMERA AND RECEIVER

The first method we tested did not fully meet our needs, leading us to change the camera and connection method. Initially, we used four devices—a camera, phone, computer, and VR glasses—which increased the chances of errors and video delays. To improve performance, we switched to a camera that transmits video via an analog signal, reducing interference and eliminating the need for a phone and computer. We selected the Eachine TX06 camera with a built-in video transmitter, and the Eachine ROTG01 Pro receiver, both cost-effective and commonly used for drones and RC cars. The camera captures video and transmits it via an analog signal with a range of up to 150 meters. Additionally, an app that supports analog video playback was required to view the video. The camera operates within a voltage range of 3.3V to 5.5V but can function at 3V, which helps prevent overheating. The receiver connects directly to the headset via a micro-USB to USB-C cable.



Figure 53: Eachine camera

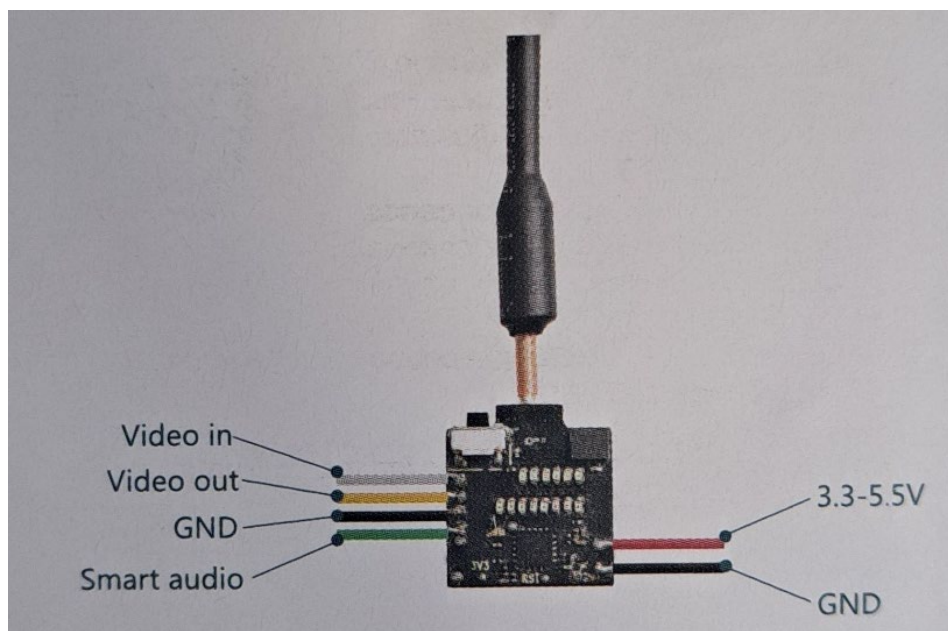


Figure 54: Eachine camera binding

2.3.3.3 INSTA 360

Insta360 is a company primarily known for its 360° cameras, though it also offers action cameras and photo/video editing software. The Insta360 X series includes four generations: ONE X1, ONE X2, X3, and X4. The latest model, X4, was released in 2024, while the first, X1, debuted in 2018. These cameras offer features such as Steady Cam mode, FlowState stabilization, AI-powered editing, water resistance up to 10m, TimeShift function, and voice control. Designed for 360° and wide-angle recording, Insta360 cameras are ideal for extreme sports and VR content. The FlowState stabilization ensures stable footage, while AI technology can automatically remove selfie sticks from recordings. Additionally, these cameras support live streaming on social media platforms. For our research project, we used the Insta360 ONE X2, which was provided by our school. It is the second model in the X series and was released in October 2020.



Figure 55: Insta360 One X2

2.3.3.4 CONCLUDED FINDINGS

The Insta360 camera provides higher image quality, supporting resolutions from 3K to 5.7K and up to 1440p in a 150° wide-angle view. In contrast, the Eachine camera has a 700 TVL resolution, roughly equivalent to 480p, making it less sharp. Both cameras display a flat image in VR glasses, limiting the benefits of 360° viewing. The key advantage of the Eachine camera is its minimal latency, which remains unaffected by Wi-Fi quality or device performance. In comparison, the Insta360 camera experienced noticeable video stuttering. Additionally, the Eachine camera is easier to connect, as it does not require intermediary devices and has had no app-related issues. Size and weight are also important factors—Insta360 is significantly larger and heavier, making it less suitable for integration into our robot. A summary of the main comparison points is provided in the table below.

COMPARISON	INSTA360 ONE X2	EACHINE
Delay	0,3 s	<0,1 s
Resolution	from 3 K to 5,7 K	700 TVL
The angle of view	from 100° to 170°, depending on the mode selected	120°

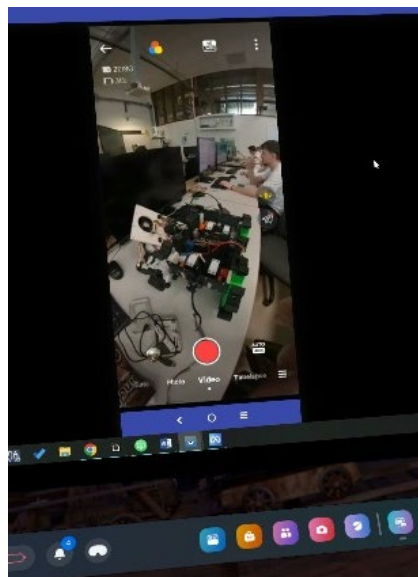


Figure 56: View through VR glasses

2.4 SOFTWARE

The main program running on the Raspberry Pi is fairly simple. We just control the motors depending on the pressed button.

Used libraries:

```
import RPi.GPIO as GPIO
import pickle #Knjiznica za shranjevanje
from evdev import InputDevice,ecodes
from pyax12.connection import Connection
from time import sleep
```

Figure 57: Python libraries

Pickle library enables us to save any data from the program into a document, which can be accessed at any time.

The Evdev library enables us to read input from any device connected to the Raspberry Pi via a USB port.

The Pyax12 library provides control over Dynamixel motors, which communicate with the Raspberry Pi via I2C communication.

For controlling our robot, we initially chose a Logitech controller, but we are considering switching to a keyboard since it offers more keys and greater flexibility. However, for testing purposes, we used the Logitech gamepad, as shown in the picture below.

The program is still not finished, but it will be soon. Since we plan to switch to keyboard controls, we did not include the button functions in the previous picture. However, you can see the basic concept of the program and the thought process behind its development. Some of the lines are commented out because we are still in the prototyping phase.

```
for i in range(0, len(Output_Pins), 1):
    GPIO.setup(Output_Pins[i], GPIO.OUT, initial = 0)

#Ustvarimo svojo funkcijo, katero bomo uporabili, ko se bodo pojavile napake
def Error_output():

    for i in range(0, len(Output_Pins), 1):
        GPIO.output(Output_Pins[i], 0)

    for i in range(0, len(Drive_Motors), 1):
        sc.set_speed(i, 0)
    sc.close()
    GPIO.cleanup()

def Motor_Stop():
    for i in range(0, len(Drive_Motors), 1):
        sc.set_speed(Drive_Motors[i], 0)

def Motor_Forward(speed):
    sc.set_speed(1, speed+1023)
    sc.set_speed(2, speed)
    sc.set_speed(3, speed+1023)
    sc.set_speed(4, speed)

def Motor_Backward(speed):
    sc.set_speed(1, speed)
    sc.set_speed(2, speed+1023)
    sc.set_speed(3, speed)
    sc.set_speed(4, speed+1023)
```

Figure 58: Program - 1 part

```
def Motor_Right(speed):
    sc.set_speed(1, speed)
    sc.set_speed(2, speed)
    sc.set_speed(3, speed)
    sc.set_speed(4, speed)

def Motor_Left(speed):
    speed = speed+1023
    sc.set_speed(1, speed)
    sc.set_speed(2, speed)
    sc.set_speed(3, speed)
    sc.set_speed(4, speed)

def Right_Tracks(pos, Speed):
    sc.set_ccw_angle_limit(7, 1023, degrees = False)
    sc.set_cw_angle_limit(7, 0, degrees = False)
    sc.goto(7, pos, speed = Speed, degrees = False)

def Left_Tracks(pos, Speed):
    sc.set_ccw_angle_limit(5, 1023, degrees = False)
    sc.set_cw_angle_limit(5, 0, degrees = False)
    sc.goto(5, pos, speed = Speed, degrees = False)

def Gripper_Rotation(pos, Speed):
    sc.set_ccw_angle_limit(10, 1023, degrees = False)
    sc.set_cw_angle_limit(10, 0, degrees = False)
    sc.goto(10, pos, speed = Speed, degrees = False)
```

Figure 59: Program - 2 part

```
|
import RPi.GPIO as GPIO
import pickle #Knjiznica za shranjevanje
from evdev import InputDevice,ecodes
from pyaxl2.connection import Connection
from time import sleep
#import Function #Lastna knjiznica

#Nastavitve
enable_position_control = 0
enable_position_control = 0
enable_max_min_control = 0

#Programu povemo na katerem USB vmesniku imamo priključeno tipkovnico
Gamepad = InputDevice("/dev/input/event4")

#Programu povemo na katerem USB vmesniku imamo priključeno tipkovnico
sc=Connection(port="/dev/ttyACM0", baudrate=1000000)

#Prikazemo podatke podane tipkovnice
print (Gamepad)

#GPIO 16 -> Ventilator st.1
#GPIO 20 -> Ventilator st.2
#GPIO 21 -> Ventilator st.3

Output_Pins = [16, 20, 21] #V seznam shranimo nase izhode
Drive_Motors = [1,2,3,4] #Ustvarimo seznam v katerih imamo shranjene pogonske motorje

Number_Of_Gears = 4 #Stevilo prestav
Number_Of_Gears_Tracks = 3 #Stevilo prestav gosenic(premikanje v paralelogramu)

GPIO.setmode (GPIO.BCM)
```

Figure 60: Program - 3 part

```

#Pozicija na zacetku programa
def Beginning_State():
    sc.set_ccw_angle_limit(10, 1023, degrees = False)
    sc.set_cw_angle_limit(10, 0, degrees = False)
    sc.goto(10, 0, speed = 1023, degrees = False)

    #Vklop ventilatorjev
    for i in range(0, len(Output_Pins), 1):
        GPIO.output(Output_Pins[i], 1)

def Eight_Joint(pos, Speed):
    sc.set_ccw_angle_limit(8, 1023, degrees = False)
    sc.set_cw_angle_limit(8, 0, degrees = False)
    sc.goto(8, pos, speed = Speed, degrees = False)

def Ninth_Joint(pos, Speed):
    sc.set_ccw_angle_limit(9, 1023, degrees = False)
    sc.set_cw_angle_limit(9, 0, degrees = False)
    sc.goto(9, pos, speed = Speed, degrees = False)

try:

    #Sporocila uporabniku o delovanju robota
    print("")
    print("FUNKCIJA TIPK: ")
    print("")
    print("D-PAD GOR/DOL      ->  Pomik robota NAPREJ / Pomik robota NAZAJ")
    print("D-PAD LEVO/DESNO   ->  Pomik robota LEVO/ Pomik robota DESNO")
    print("Tipka RB/LB        ->  Pomik DESNIH gosenic NAPREJ / Pomik LEVIH gosenic NAPREJ")
    print("Tipka RT           ->  Prestave HITROSTI")
    print("Tipka START        ->  Vklop VENTILATORJEV")

    print("")
    print("DYNAMIXLE AX-18A MOTORJI SO POVEZANI")
    print("")
    #print("NAPETOST BATERIJE: " + str(Voltage) + " V")
    print("")

```

Figure 61: Program - 4 part

```
count_0 = 0
count_1 = 1
count_2 = 0
count_3 = 0

position = 0 #Ukaz za pomik obeh gosenic hkrati
Count_Speed_Tracks = 1
index = 1

Beginning_State()

Speed = []

#Napolnimo prazni seznam(array)
for x in range(0, Number_Of_Gears+1,1):
    Speed.append(x)
print(Speed)

#Program za avtomatsko nastavljanje prestav glede na spremenljivko "Number_Of_Gears"
Speed_Of_First_Gear = int(1023/Number_Of_Gears)

for i in range(0, Number_Of_Gears+1, 1):
    Speed[i] = (Speed_Of_First_Gear)*i

print("")
print("Prestava št: " + str(count_1))
print("")

Speed_Tracks = []

#Napolnimo prazni seznam
for x in range(0, Number_Of_Gears_Tracks+1,1):
    Speed_Tracks.append(x)
print(Speed_Tracks)

Speed_Of_First_Gear_Tracks = int(1023/Number_Of_Gears_Tracks)
```

Figure 62: Program - 5 part

```

for i in range(0, Number_Of_Gears_Tracks+1, 1):
    Speed_Tracks[i] = (Speed_Of_First_Gear_Tracks)*i

if(enable_position_control == 1): #V primeru, da smo v nastavitvah nastavili vrednost na 1 se naj zgodi naslednje:
    print("")
    usr_in = int(input("Prosim vpišite št. 1 v primeru, da želite vklopiti pozicioniranje gosenic. "))
    if(usr_in == 1):
        usr_pos_1 = int(input("Ko boste nastavili pozicijo 1, vtipkajte 1. "))
        if(usr_pos_1 == 1):
            pos_1_right = sc.get_present_position(7)
            pos_1_left = sc.get_present_position(5)

            print(str(pos_1_right))
            print(str(pos_1_left))

            usr_pos_2 = int(input("Ko boste nastavili pozicijo 2, vtipkajte 2. "))

            if(usr_pos_2 == 2):
                pos_2_right = sc.get_present_position(7)
                pos_2_left = sc.get_present_position(5)

                print(str(pos_2_right))
                print(str(pos_2_left))

            usr_pos_3 = int(input("Ko boste nastavili pozicijo 3, vtipkajte 3. "))

            if(usr_pos_3 == 3):
                pos_3_right = sc.get_present_position(7)
                pos_3_left = sc.get_present_position(5)

                print(str(pos_3_right))
                print(str(pos_3_left))

            position_tracks = [pos_1_left, pos_1_right, pos_2_left, pos_2_right, pos_3_left, pos_3_right]
            print(str(position_tracks))

```

Figure 63: Program - 6 part

```

#Podatke shranimo v datoteko, da lahko do njih dostopamo

pickle_out = open("pozicija.pickle","wb")
pickle.dump(position_tracks, pickle_out)
pickle_out.close()

pickle_in = open("pozicija.pickle","rb")
position_tracks = pickle.load(pickle_in)

if(enable_max_min_control == 1):
    print("")
    usr_in = int(input("Prosim vpišite št. 1 v primeru, da želite določiti maksimalno in minimalno vrednost gosenic "))
    if(usr_in == 1):
        usr_pos_max = (input("Ko boste nastavili maksimalno pozicijo, vpišite MAX "))

        if(usr_pos_max == "MAX"):
            pos_1_max = sc.get_present_position(7)
            pos_2_max = sc.get_present_position(5)

            print("Maksimalna pozicija desnih gosenic: " + str(pos_1_max))
            print("Maksimalna pozicija levih gosenic: " + str(pos_2_max))

            print("")
            usr_pos_min = (input("Ko boste nastavili minimalno pozicijo, vpišite MIN "))

            if(usr_pos_min == "MIN"):
                pos_1_min = sc.get_present_position(7)
                pos_2_min = sc.get_present_position(5)

                print("Minimalna pozicija desnih gosenic: " + str(pos_1_min))
                print("Minimalna pozicija levih gosenic: " + str(pos_2_min))

            max_min_position_tracks = [pos_1_max, pos_2_max, pos_1_min, pos_2_min]
            print(str(max_min_position_tracks))

            pickle_out = open("pozicija_min_max.pickle","wb")
            pickle.dump( max_min_position_tracks, pickle_out)
            pickle_out.close()

```

Figure 64: Program - 7 part

```

pickle_in = open("pozicija_min_max.pickle", "rb")
max_min_position_tracks = pickle.load(pickle_in)
print(str(max_min_position_tracks))

for event in Gamepad.read_loop():

    #Vklop ventilatorjev s tipko start na Gamepadu
    #if(event.value == 1 and event.code == 313):
        #count_0 = count_0+1

        #if(count_0 == 1):
            #print("")
            #print("STANJE VENTILATORJEV: VKLOPLJENI")
            #print("")
            #for i in range(0,len(Output_Pins),1):
                #GPIO.output(Output_Pins[i], 1)

        #if(count_0 == 2):
            #print("")
            #print("STANJE VENTILATORJEV: IZKLOPLJENI")
            #print("")
            #for i in range(0,len(Output_Pins),1):
                #GPIO.output(Output_Pins[i], 0)
            #count_0 = 0

    #Določanje Prestav, za vsako, ko pritisnemo se prestava mora povečati
    if(event.value == 1 and event.code == 311):
        count_1 = count_1+1
        if(count_1 == len(Speed)):
            print("")
            count_1 = 1
        print("Prestava št. : " + str(count_1))

```

Figure 65: Program - 8 part

```

#Določanje prestav gosonic(premikanje v paralelogramu), vsakič, ko pritisnemo se prestava poveča
if(event.value == 1 and event.code == 310):
    Count_Speed_Tracks = Count_Speed_Tracks+1
    if(Count_Speed_Tracks == len(Speed_Tracks)):
        print("")
        Count_Speed_Tracks = 1
    print("Prestava št. : " + str(Count_Speed_Tracks))

#Pomik robota naprej/nazaj
if(event.value == -1 and event.code == 17):
    Motor_Forward(Speed[count_1])

if(event.value == 0 and event.code == 17):
    Motor_Stop()

if(event.value == 1 and event.code == 17):
    Motor_Backward(Speed[count_1])

#Pomik robota desno/levo
if(event.value == 1 and event.code == 16):
    Motor_Right(Speed[count_1])

if(event.value == 0 and event.code == 16):
    Motor_Stop()

if(event.value == -1 and event.code == 16):
    Motor_Left(Speed[count_1])

```

Figure 66: Program - 9 part

```
#Pomik desnih gosenic v paralelogramu
if(event.value == 1 and event.code == 309):
    Motor_Stop()
    print("Pomikanje desnih gosenic")
    pos = sc.get_present_position(7)
    while(1 == 1):
        if(pos > 670 and count_2 == 0 or pos < 230 and count_2 == 1):
            Right_Tracks(pos, 0)
            break
        else:
            if Gamepad.active_keys(309):
                if(count_2 == 0):
                    pos = pos+20
                    Right_Tracks(pos, 300)
                if(count_2 == 1):
                    pos = pos-20
                    Right_Tracks(pos, 1023+300)

            else:
                break

    if(event.value == -1 and event.code == 17):
        print("OK")

#Spreminjanje smeri vrtenja motorja-> desne gosenice
if(event.value == 1 and event.code == 315):
    count_2 = count_2+1
    if(count_2 == 2):
        count_2 = 0

if(event.value == 0 and event.code == 309):
    print("Prenehanje premikanja gosenic")
```

Figure 67: Program - 10 part

```
#Premikanje levih gosenic
if(event.value == 1 and event.code == 308):
    Motor_Stop()
    print("Pomikanje levih gosenic")
    pos = sc.get_present_position(5)
    while(1 == 1):
        if(pos > 810 and count_3 == 0 or pos < 460 and count_3 == 1):
            Left_Tracks(pos, 0)
            break
        else:
            if Gamepad.active_keys(308):
                if(count_3 == 0):
                    pos = pos+20
                    Left_Tracks(pos, 600)
                if(count_3 == 1):
                    pos = pos-20
                    Left_Tracks(pos, 600)
            else:
                break

#Spreminjanje smeri vrtenja motorja -> leve gosenice
if(event.value == 1 and event.code == 314):
    count_3 = count_3 + 1
    if(count_3 == 2):
        count_3 = 0

if(event.value == 0 and event.code == 308):
    print("Prenehanje premikanja gosenic")
```

Figure 68: Program - 11 part

```
#Premikanje obeh gosenic skupaj
if(event.value == 1 and event.code == 312):
    #position_tracks = [pos_1_left, pos_1_right, pos_2_left, pos_2_right, pos_3_left, pos_3_right]
    if(index == 1):
        count_2 = 1
        index = 0
    pos_1 = sc.get_present_position(7)
    pos_2 = sc.get_present_position(5)
    print(str(pos_2))
    position = position+1
    if(position == 3):
        position = 1
    print(str(position))

    if(position == 1):
        Right_Tracks(position_tracks[1], Speed_Tracks[Count_Speed_Tracks])
        Left_Tracks(position_tracks[0], Speed_Tracks[Count_Speed_Tracks])

    if(position == 2 and count_2 == 0):
        Right_Tracks(position_tracks[3], Speed_Tracks[Count_Speed_Tracks])
        Left_Tracks(position_tracks[2], Speed_Tracks[Count_Speed_Tracks])

    if(position == 2 and count_2 == 1):
        Right_Tracks(position_tracks[5], Speed_Tracks[Count_Speed_Tracks])
        Left_Tracks(position_tracks[4], Speed_Tracks[Count_Speed_Tracks])

if(event.value == 1 and event.code == 305):
    pos_8 = sc.get_present_position(8)
    pos_9 = sc.get_present_position(9)
    while(1 == 1):
        if Gamepad.active_keys(305):
            sc.set_speed(0, 400)
            pos_8 = pos_8-2
            pos_9 = pos_9+10
            if(pos_8 < 5):
                pos_8 = sc.get_present_position(8)
            if(pos_9 >= 550):
                pos_9 = sc.get_present_position(9)
            Eight_Joint(pos_8, 1023+1023)
            Ninth_Joint(pos_9, 1023)
```

Figure 69: Program - 12 part

```

else:
    sc.set_speed(0,0)
    break

if(event.value == 1 and event.code == 307):
    pos_8 = sc.get_present_position(8)
    pos_9 = sc.get_present_position(9)
    while(1 == 1):
        if Gamepad.active_keys(307):
            sc.set_speed(0, 1023+400)
            pos_8 = pos_8+2
            pos_9 = pos_9-10
            if(pos_9 <= -5):
                pos_9 = sc.get_present_position(9)
                Eight_Joint(pos_8, 1023)
                Ninth_Joint(pos_9, 1023+1023)

        else:
            sc.set_speed(0, 0)
            break

#Obracanje gripperja za 360 stopinj, ko prvic pritisnemo gre na 360, ko drugic gre na 0 stopinj
if(event.value == 1 and event.code == 313):
    count_0 = count_0+1
    if(count_0 == 1):
        Gripper_Rotation(1023, 1023)#Gripper_Rotation(pos, speed)
        print("Gripper se je obrnil za 360 stopinj!")

    if(count_0 == 2):
        Gripper_Rotation(0, 1023)
        count_0 = 0
        print("Gripper se je obrnil na 0 stopinj!")

```

Figure 70: Program - 13 part

```

if(count_0 == 2):
    Gripper_Rotation(0, 1023)
    count_0 = 0
    print("Gripper se je obrnil na 0 stopinj!")

if(event.value == 1 and event.code == 304):
    #pos = sc.get_present_position(8)
    #print(str(pos))
    Eight_Joint(0, 200)

except Function.VoltageError:
    #print("")
    #print("Prenizka napetost, zamenjajte baterijo!")
    #Error_output()

except ValueError:
    print("")
    print("DYNAMIXLE motorji se niso uspeli povezati, napaka v napajanju ali priklopu motorjev!")
    GPIO.cleanup()

except KeyboardInterrupt: #ctrl+c
    GPIO.cleanup()

```

Figure 71: Program - 14 part

3 LIST OF BOUGHT COMPONENTS:

- **Raspberry Pi Model 4 B** (recommended by OARK)
Acts as the brain of the robot, serving as the interface between user inputs and motors.
- **Dynamixel AX18-A, 12A**
Used for the robot's arm.
- **Dynamixel XM-430-W-350-R**
Used for the robot's movement.
- **G6 metal robotic gripper**
Modular gripper attached to the end of the robot's arm.
- **USB2AX v3.2a** (recommended by OARK)
Serves as the communication link between the the RPi and Dynamixel motors.
- **Raspberry Pi camera module** (recommended by OARK)
A simple camera module stationed at the end of the robot's arm.
 - 360 **Raspberry Pi camera module**
- **Various small electrical components** (LEDs, transistors, resistors...)
Used for manipulating the electric current or signaling the robot's current state.
- **MQ sensor(CO2 sensor)**
Used for detecting CO2 levels in air.
- **Thermal camera**
Used for measuring the temperature from a distance.
- **Battery Gens Bashing 5500 mAh**
Used as a power supply for the motors and other electrical consumers? (except the RPi).

COMPONENTS	NR.	PRICE 1P (EUR)	SUM (EUR)
DYNAMIXEL XM-430-W-350-R	6	289.90€	1739,7€
DYNAMIXEL AX-18A	8	94,90 €	759,20 €
RASPBERRY PI 4 MODEL B	1	38,75 €	38,75 €
G6METAL ROBOTIC GRIPPER	1	26,43 €	26,43 €
USB2AX V3.2 A	1	39,95 €	39,95 €
LOGITECH GAMEPAD F710	1	39,84 €	39,84 €
GOSTIME BRUSHLESS DC FAN	2	3,30 €	6,60 €
MICROSWITCH SW152-ND	2	1,20 €	2,40 €
PUSH BUTTON MOMENTARY SWITCH	2	1,60 €	3,20 €
RASPBERRY CAMERA BOARD	1	27,55 €	27,55 €
BATTERY GENS BASHING 5500 mAh	3	55.90 €	167,70 €
VARTA 4.5 V	2	9,99 €	19,98 €
BEARINGS 6 X 22 X 10	8	4,49 €	35,92 €
AXES	2	2,24 €	4,48 €
FILAMENT	5	38,78 €	193,90€
SMALL PARTS	20	0,40-2,00€	20,00 €
MQ CO2 SENSOR	1	2,25 €	2,25 €
THERMAL CAMERA	1	40,15€	40,15€
		Total:	3.168€

4 FUTURE PLANS

Our plans include establishing a company and participating in an entrepreneurship competition in Slovenia. Our robot is designed to assist elderly individuals in hospitals and at home. We have modified its design and construction to better fit apartments and elderly care homes while maintaining the chassis, driving capabilities, and software similar to those of a rescue robot.

To ensure the success of our project, we collaborated with an external mentor who guided us through the development process. We began by conducting surveys to gain insight into the real challenges elderly people face and what solutions they truly need. Our research revealed that elderly care homes are overcrowded and there is a significant shortage of staff. With our robot, we aim to address this issue by helping compensate for the lack of personnel in care homes and enabling elderly individuals to live independently in their own homes for a longer period.

Our company is called iCare, and customers will have the option to purchase or rent the robot through a monthly subscription. We will develop a prototype, create a financial report, design a marketing strategy, establish an organizational structure, and create a logo. Additionally, we will prepare a presentation, set up a booth with all promotional materials, and ensure that we have everything a company needs to succeed.

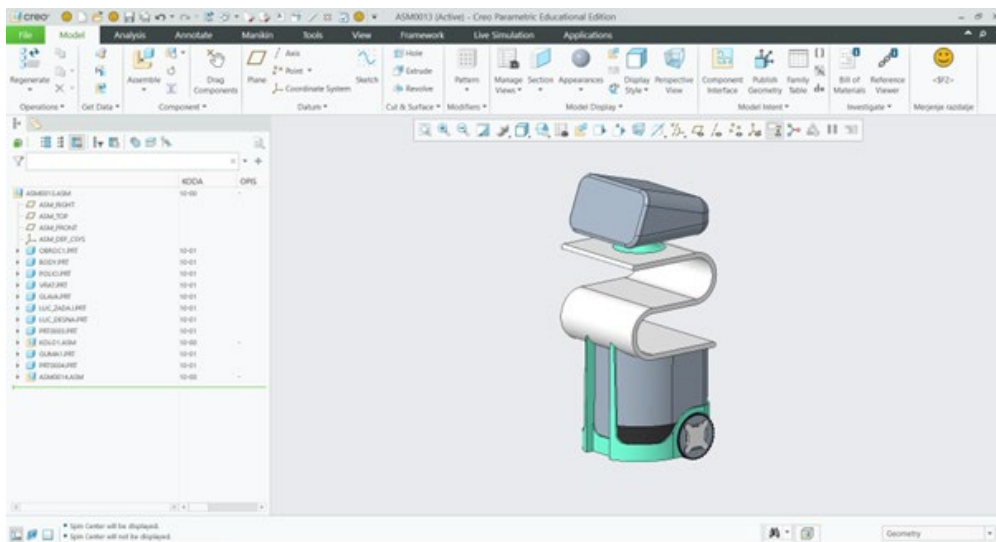


Figure 72: iCare robot - not finished

5 CONCLUSION

The project was a major challenge at the beginning since we knew that we would face many difficulties. We had a basic concept of the robot from last year, but we upgraded it. The first major challenge was the programming aspect and CAD-designing as we had to learn these programs on our own since they are not part of our school curriculum. During the school year, we also conducted a research paper on computer vision for which we received golden recognition for our hard work. In conclusion, we gained a lot of experience, not only in programming and CAD design but also in teamwork.