Object oriented programming and software engineering

Waiter-bot

Mikołaj Suchoń 405821
Michał Jaros 407223

Index:

Project assumptions/abstract:

Our goal is to create a mobile and autonomous robot for indoor food delivery.

Robot capabilities:
Ability to transport standard size meals over a flat, indoor terrain.
Possibility of autonomous or remote control with obstacle detection and appropriate reaction

Project stages:

- Initial concept
- Selection of components
- 3D Modelling
- Simulation of robot in environment
- Programing
- Physical proof of concept
- Implementation into final prototype

Business assumptions of the project:

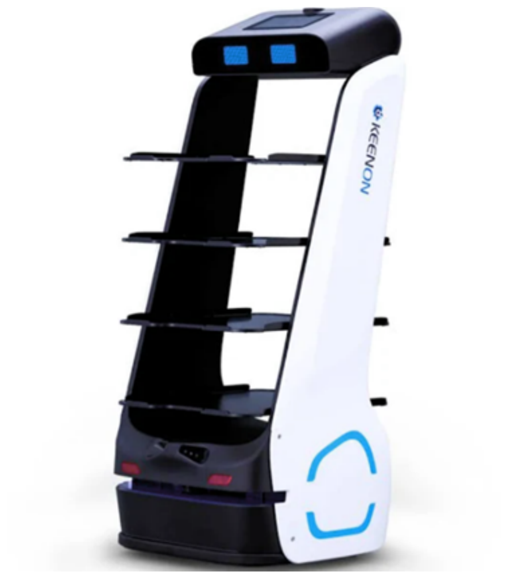
There are many models of robots on the market that can replace waiters in their work. However, these devices are more of a technological novelty than a commonly used method of serving restaurant guests. Their purpose is often to provide entertainment and attract people to the restaurant, rather than effective work as a waiter. These machines resemble people or characters from science-fiction movies, they communicate using simple sentences, they are large in size, require a lot of energy, and most importantly, their price can reach tens of thousands of PLN.

The aim of our project is to introduce robots as a method of serving restaurant guests on a large scale. The purpose of our invention is to replace the work of a waiter to a limited extent, which will lead to increase of the efficiency of service or enable staff reduction for small catering businesses.

The robot's task is to deliver meals and collect used tableware from the tables. It is supposed to be cheap to produce thanks to getting rid of the usual elements affecting the visual effect and screens needed to communicate with the robot (ordering dishes can be done through a dedicated application). The robot has to be able to move between tables using a given path and react to static and dynamic obstacles appearing on its way. The device is supposed to work continuously, which requires the possibility of easy and quick battery replacement.

State of the art:

As we mentioned before, on the market there are available several models of the waiter like robots, here are some examples of them:



Keenon, Dinerbot T5

89 000 PLN

Fig.1 Keenon Dinerbot T5

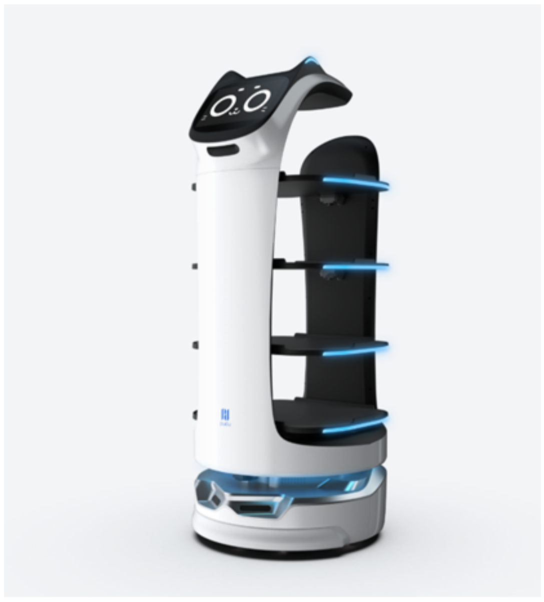


Fig.2 Pudu Robotics, BellaBot

Pudu Robotics, BellaBot

65 000 PLN

This robot became very popular in Poland in 2023 after one of the hypermarket chains, Carrefour company introduced it as their customer assistant, „Kerfuś”

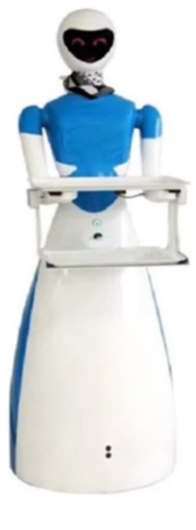


Fig.3 4S SISTER XII

4S SISTER XII

30 000 PLN

As we mentioned before all these products are huge, expensive and their main role is rather entertaining than serving customers.

Description of the target group and its requirements:

First of all, our target group are all establishments offering meals on site. However not every company will be able to fully use the potential of our product. Owners who have large premises or those with a lot of clients will benefit most from this purchase. Thanks to our device, even restaurateurs who are less wealthy would afford the right amount of service. By purchasing our solution, they will be able to serve the same number of customers while incurring much lower costs. Robots supporting one or two waiters will be cheaper to buy and operate than employing additional people. This device, however, is intended for use inside the building, so it will not find application in premises where the garden is an important part of the business. Finally, our device does not replace a human in 100%, therefore, owners of places that require the service to be at a high level, e.g. the waiter who: advises the choice of the dish, serves drinks directly to the table or cleans up used tableware, will not be interested in our product. Therefore, the main features of our target group are: large area and number of clients, low budget for stuff, places of low and medium standard. This group will include: large-format cafes, pizzerias, diners, taverns, inns, fast-food restaurants, medium-standard restaurants, etc.

Morphological analysis:

A morphological analysis was conducted in order to define the main problems of the system, possible solutions and our choices for them.

Main problems recognised by us were:

- Mobility
- Power supply
- Controler
- Motors
- Navigation

Where we settled on such solutions in order:

- Two driven wheels and two castor wheels system
- Custom power bank
- Raspberry pi 4b
- Stepper motors nema 23
- Path planning with line following and QR codes

The morphological chart with solutions considered by us and with short explanations of pros and cons is attached with this report.

Design assumptions:

Robot capabilities:

- Food compatible
- Capable of carrying at least 5 kg
- Mooves with speed between 0.25 and 1 m/s (depending on load)
- Ground clearance of at least 3 cm
- At least 2 horse working time
- Obstacle avoidance
- Over all hight at most 1 m

Servos:
80 rpm and at least 1 Nm of torque

Design of the mechanical parts:

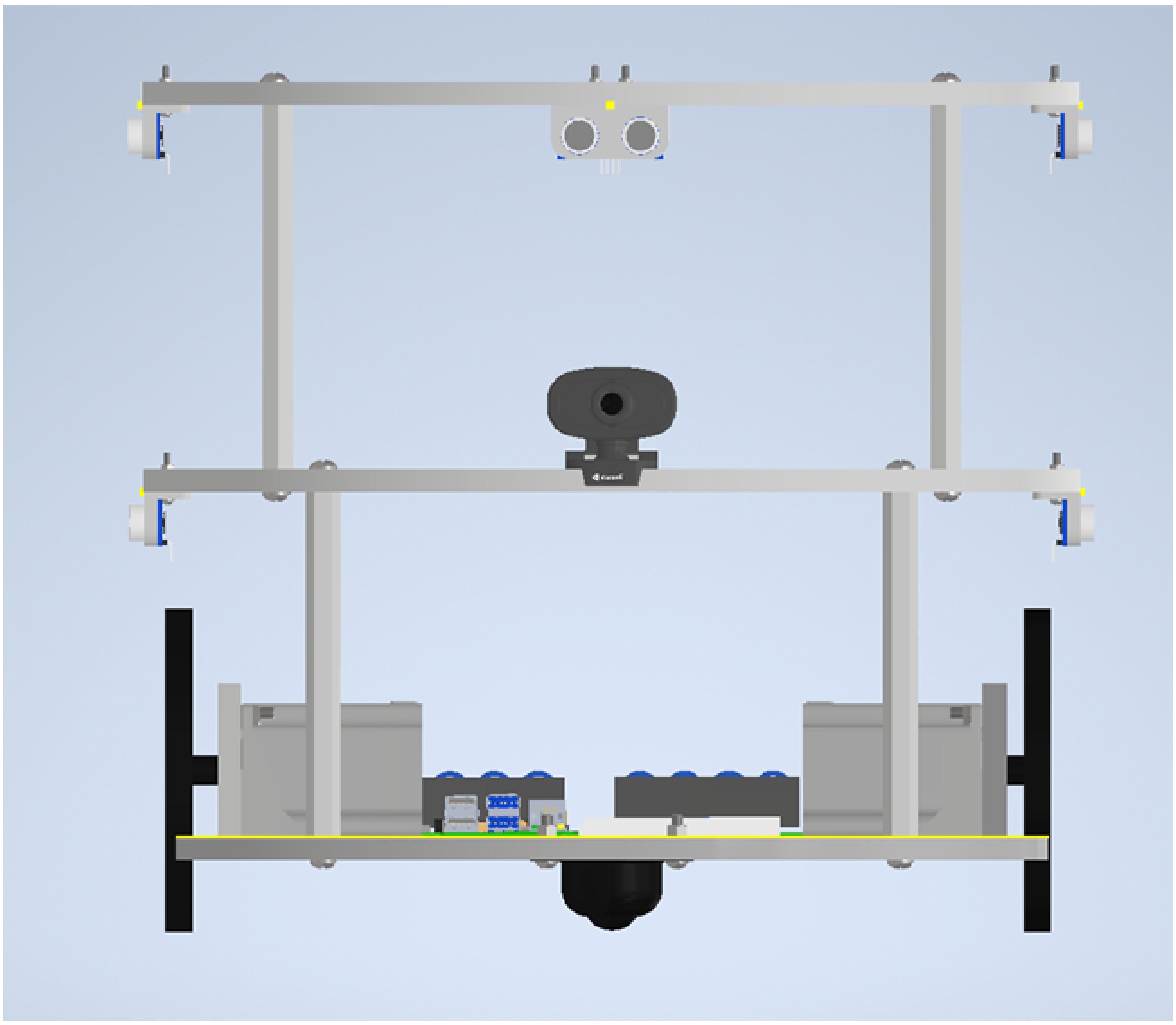


Fig.4 Front view of the robot

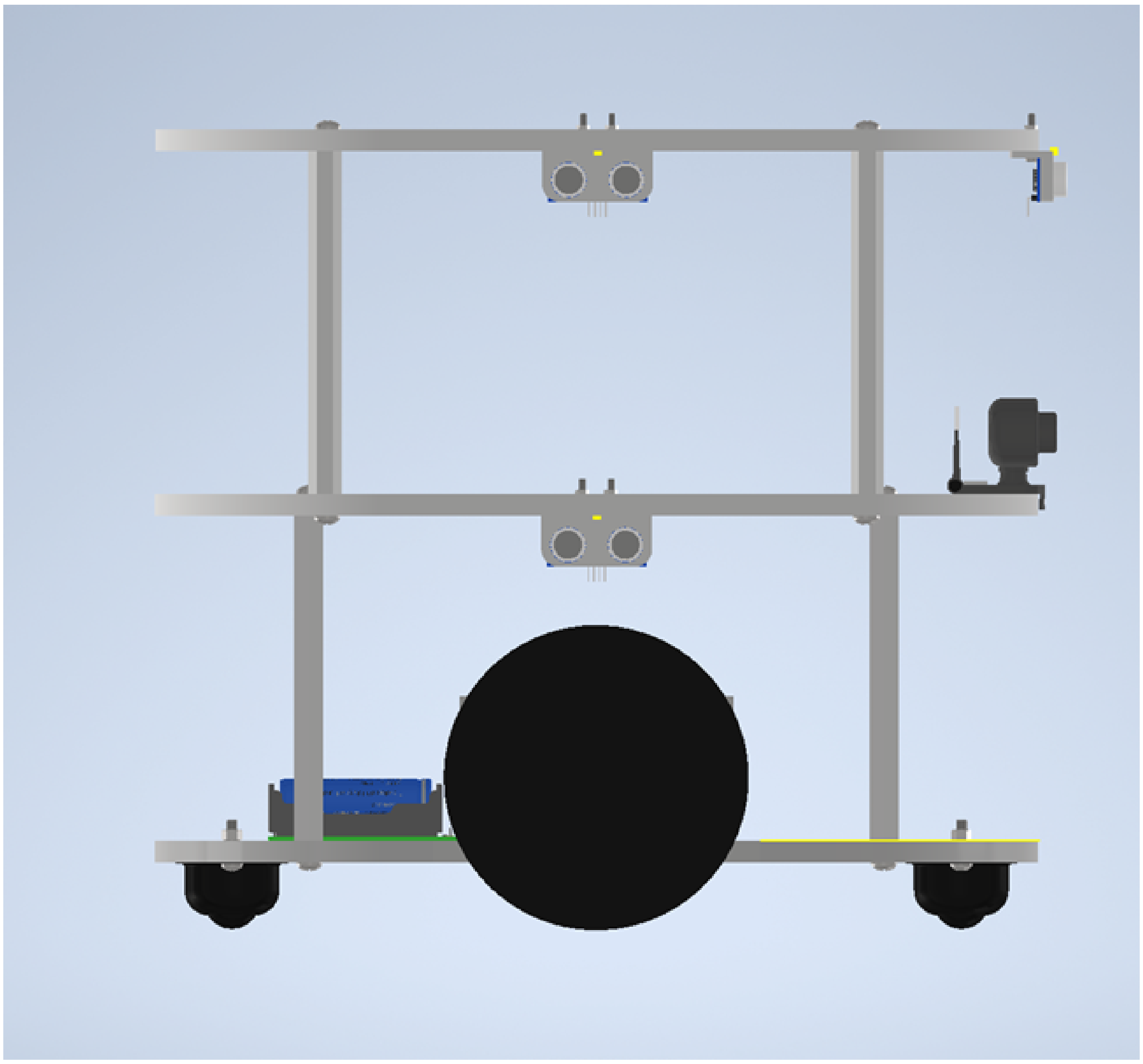


Fig.5 Side view of the robot

Our robot is constructed of 3 manufactured main parts: Base plate, Middle plate and Top plate. They are connected with metal sleeves.

Base plate – the most important part of the robot. All the plates have a circular shape. Base plate additionally has brackets for supporting the wheels. Inside each bracket mounted on the one side of the plate we have a ball bearing. To the bracket has been attached a Nema 23 stepper motor from the one side and the wheel from the other side, the connection is performed by the pressfit. These are our driving wheels. Additionally, to the plate has been attached two caster wheels by the bolt connection for keeping balance of our robot. Thanks to that the robot is able to move in forward and backward direction as well as turning around its own axes. To the base plate are attached also: 2 battery packs, Raspberry pi 4 and the board with an electrical circuit (more about it in the electrical design part).

Middle plate – middle plate has two ultrasonic proximity sensors for detecting obstacles around the robot and the web camera that performs a variety of tasks.

Top plate – this part has additional proximity sensors, thanks to sensors placed on the different heights, we can detect obstacles in a more 3D like manner. The top plate is also used for placing the dishes on top of it and is manufactured out of food-friendly material.

Working drawings of not standardised parts:

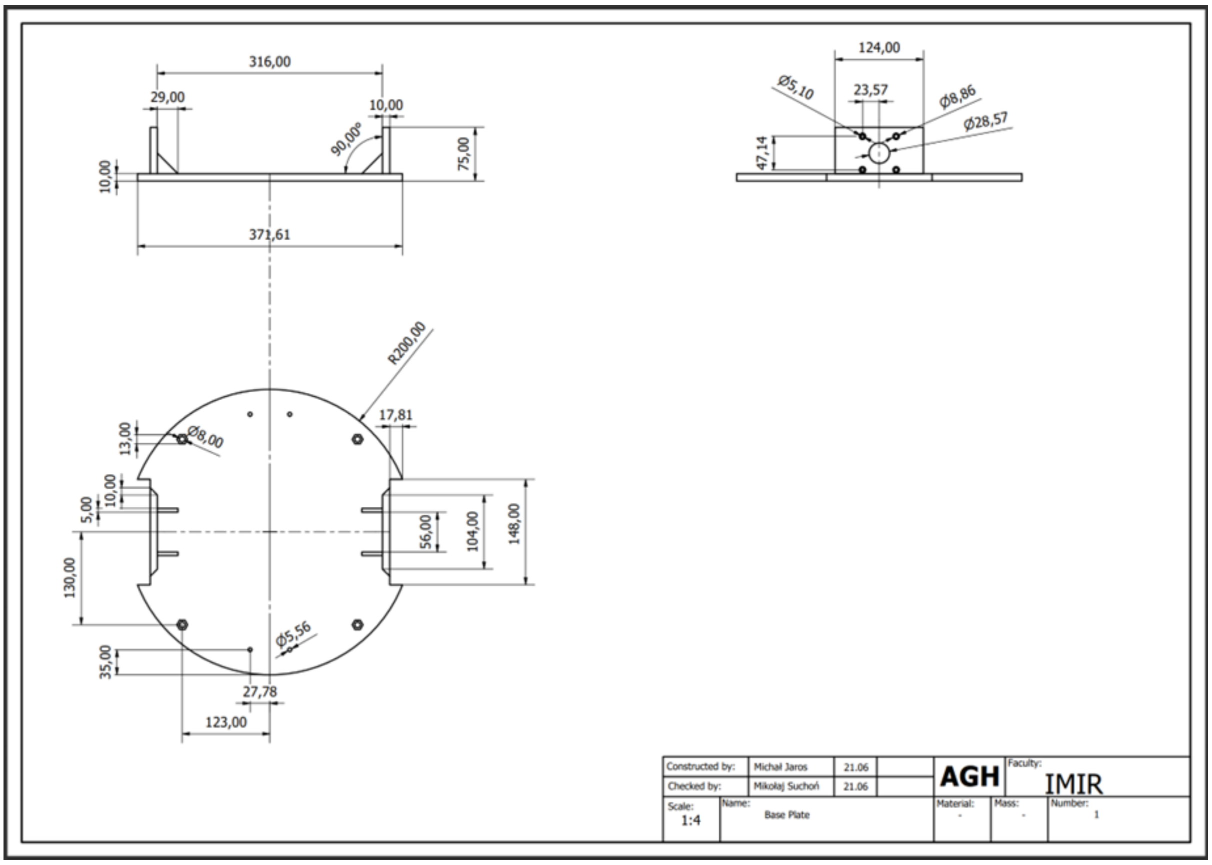


Fig. 6 Base plate working drawing

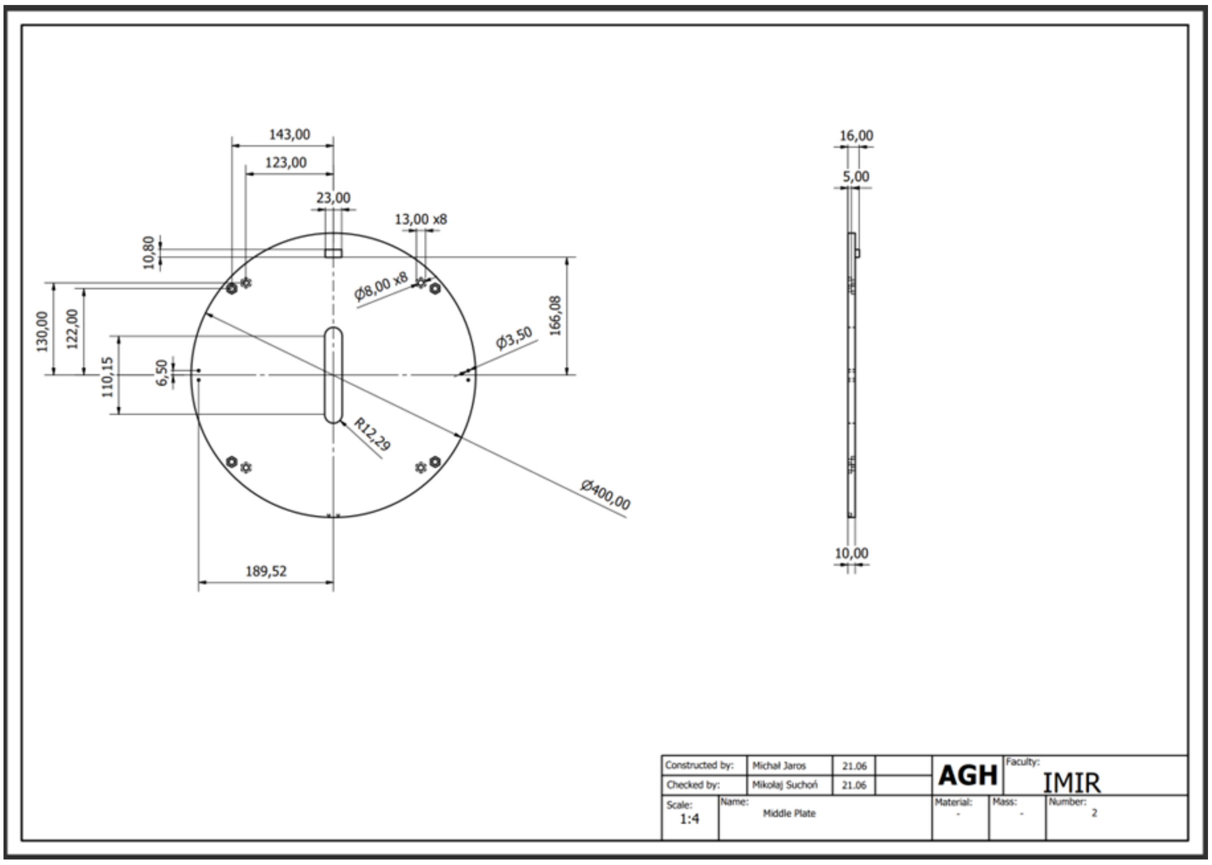


Fig.7 Middle plate working drawing

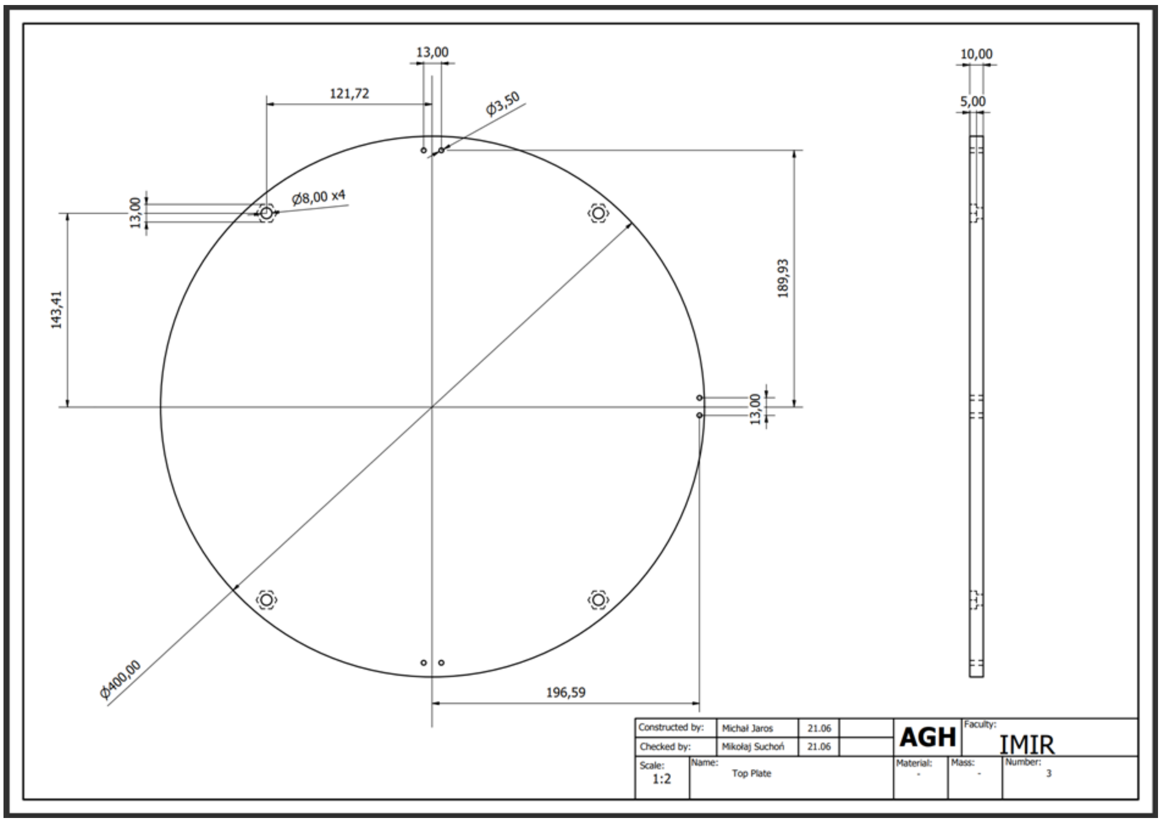


Fig.8 Top plate working drawing

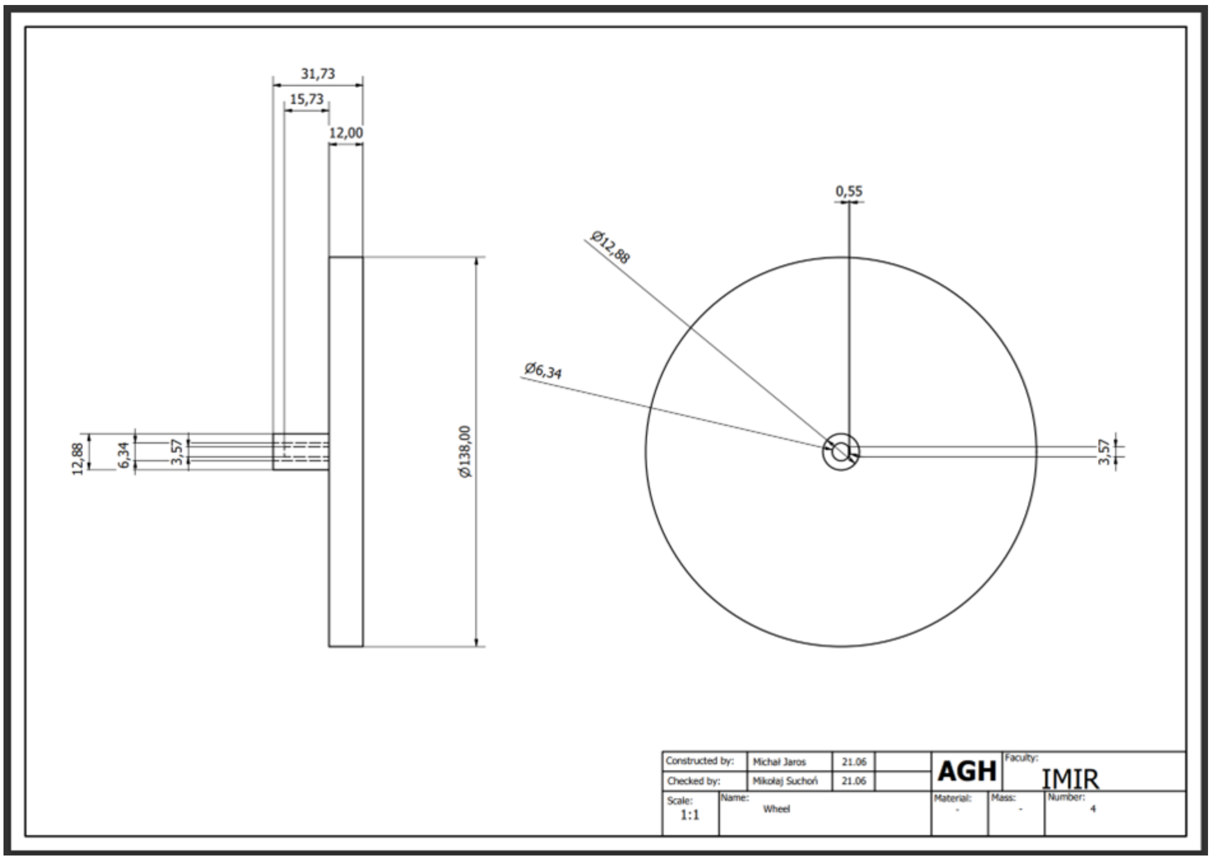


Fig.9 Wheel working drawing

Screenshots of models:

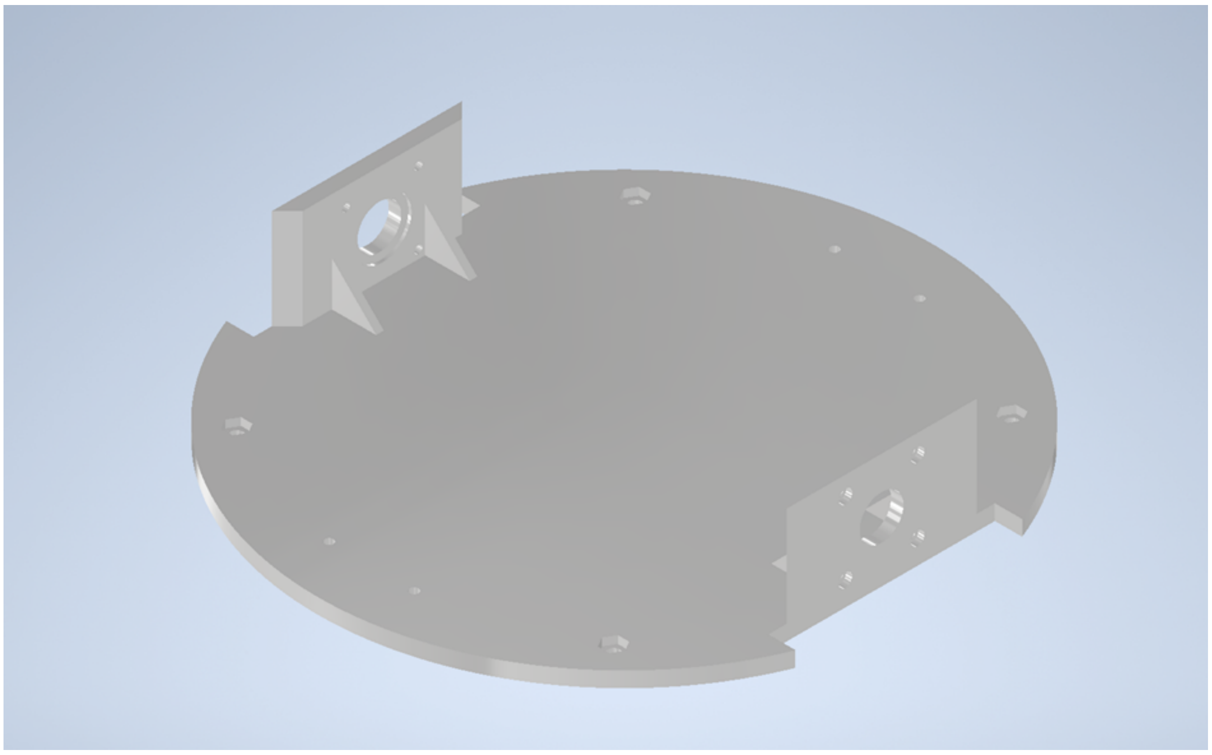


Fig.10 Base Plate

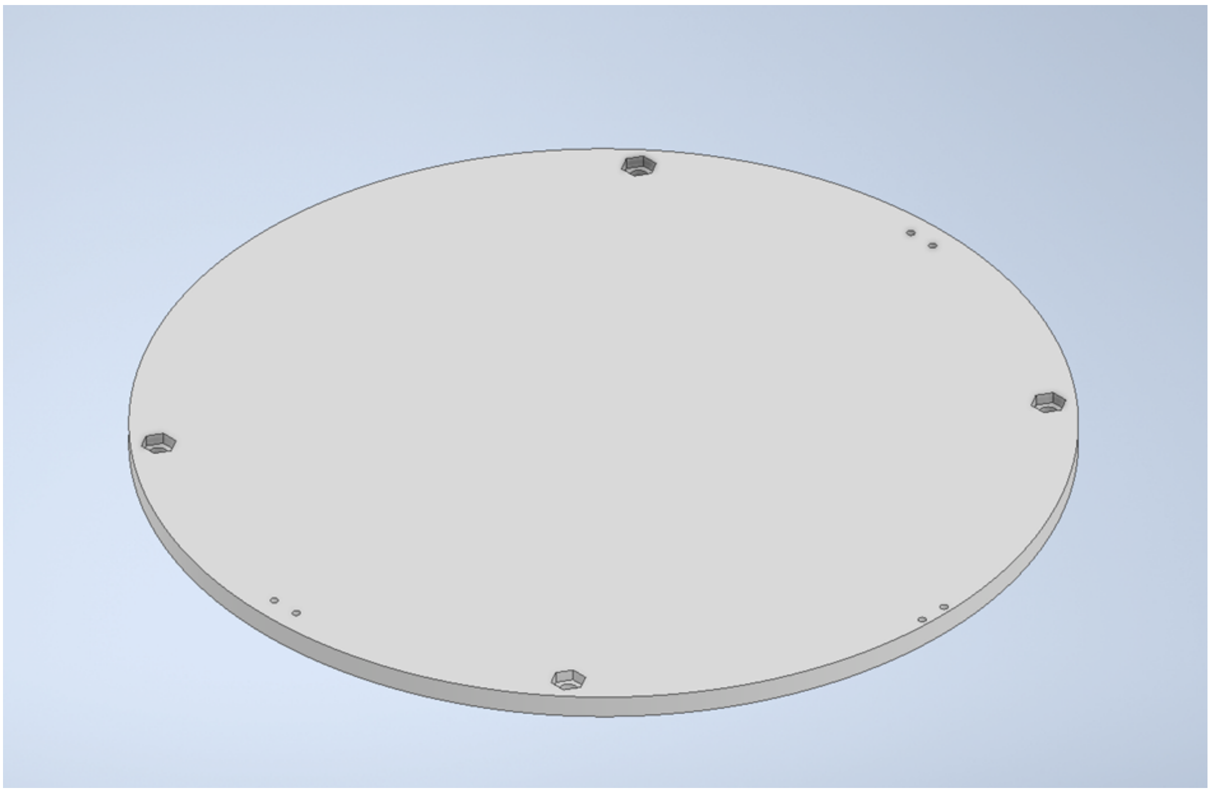


Fig.11 Middle Plate

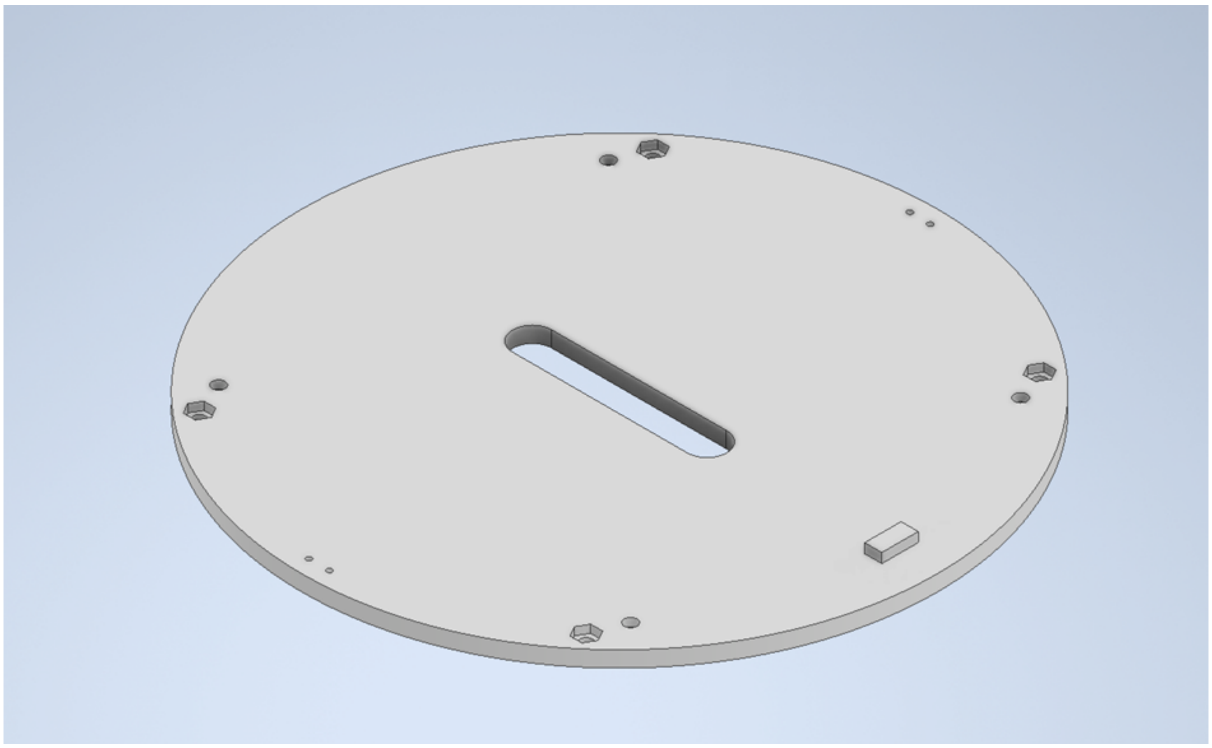


Fig.12 Top Plate

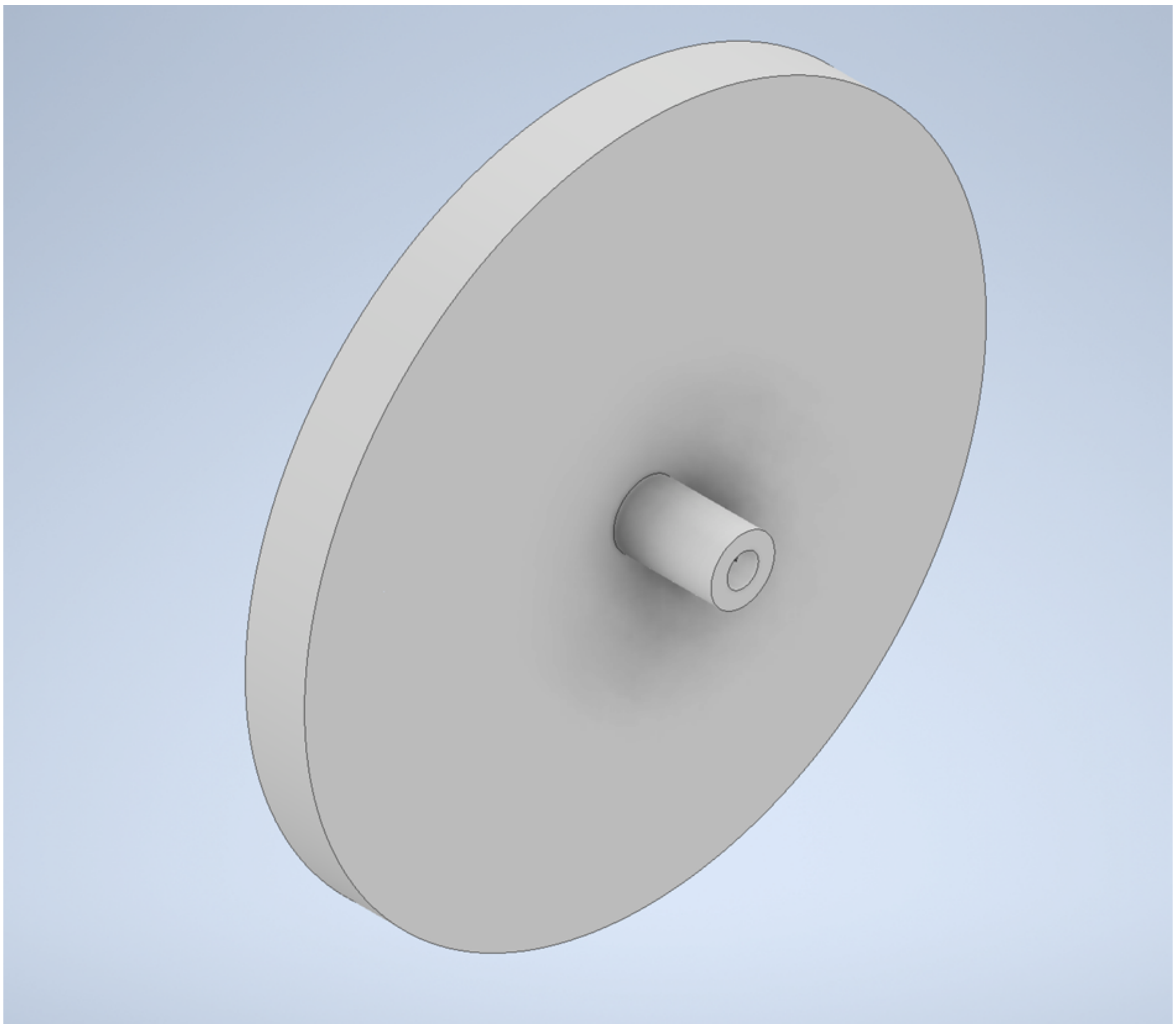


Fig.13 Wheel

Strength analysis:

All strength calculation reports are attached with the report containing all constraints, loads, reaction forces and all necessary detail.

We analyse the four main components of our project. The top plate with its assumed load and support. Then we propagate the reaction force as load to the middle plate with its own load and supposed support. Again we take that reaction as load for the base plate with its surface load and support at castor wheel and coupler bearings. Finally we calculate the strength of the coupling element with the final reaction forces. Below we present our results in reverse order to the order of calculations.

Wheel-base coupling:

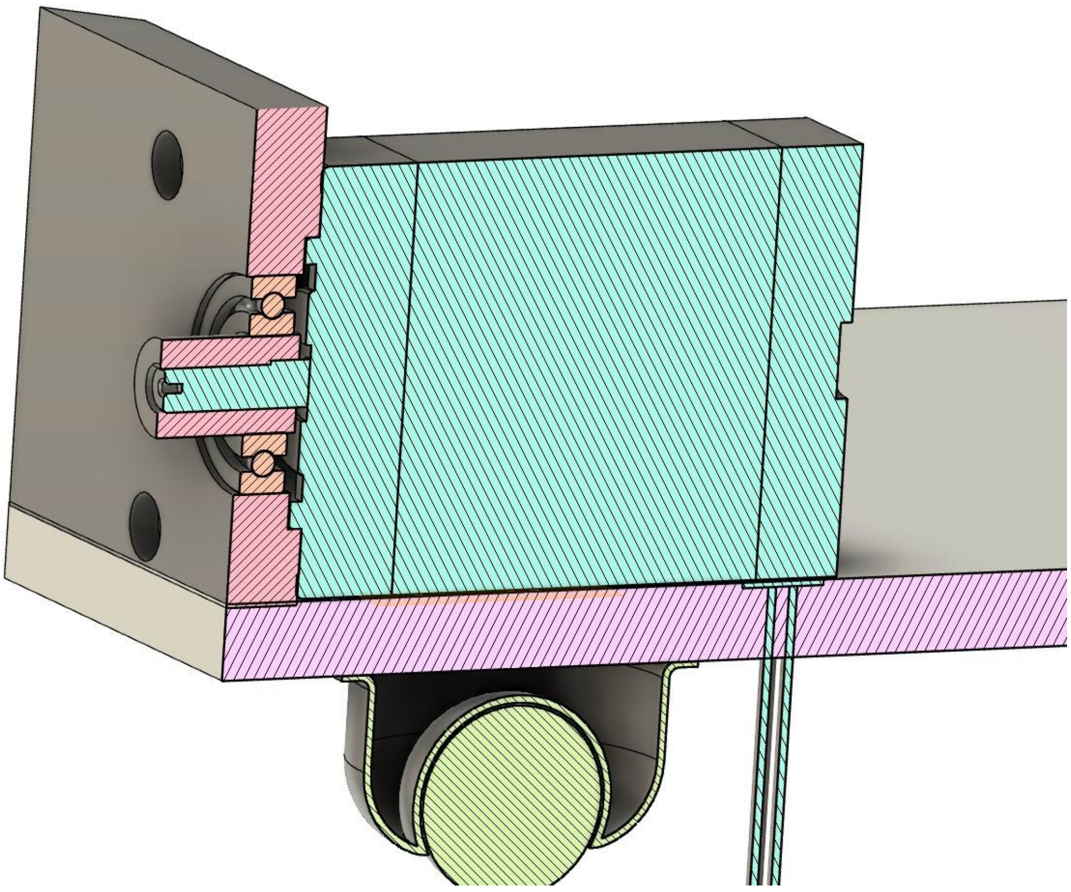


Fig.14 Base plate - motor bracket assembly section

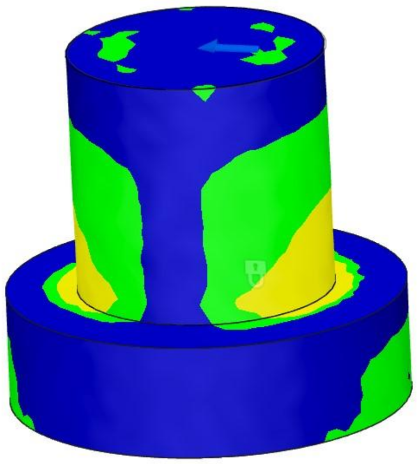


Fig.15 Coupling element simulation

In the final product the wheel has an integrated keyed coupler that connects to the motor shaft and is secured in the motor bracket by a bearing.

For the purpose of wheel base coupling a new 3d model had to be made to integrate the beating, motor shaft and wheel coupler.
The simulation assumes a uniform ABS plastic and calculations show all strength requirements are met. The fact that in reality the coupler has a metal core greatly increases the strength of the model and is considered as an additional safety factor.

Constraint: bering
Force: 20kg applied on the wheel sheer line

We chose the 20kg load due to the reaction forces on that bering from the motor bracket

Base:

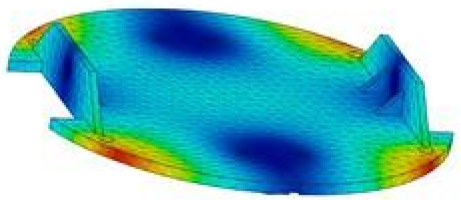


Fig.16 Base plate simulation
For the simulation of the base we chose the hole of the bearing and caster wheels as constraints and for load we applied a uniform 20kg load on the top surface and an additional 20kg load on the four supports for higher levels.
The simulation concluded a high safety factor.

Middle plate:

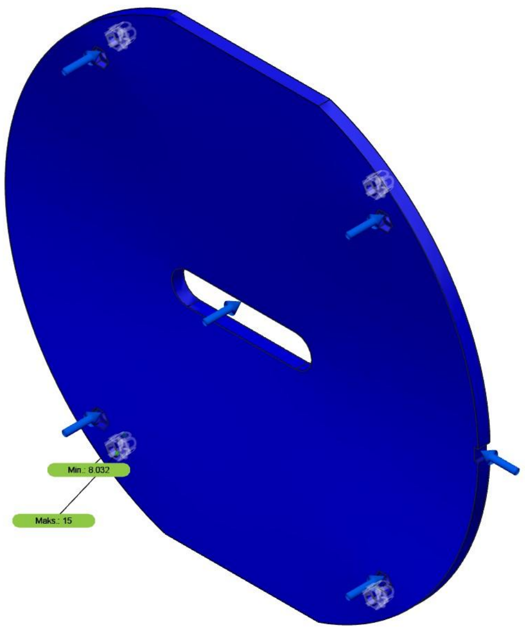


Fig.17 Middle plate simulation
Load: 10kg from the plate above, 10kg from uniform load on plate surface, 10kg from simulated inertia from robot acceleration.

Constrained in plate support holes.
Material: ABS

Top plate:

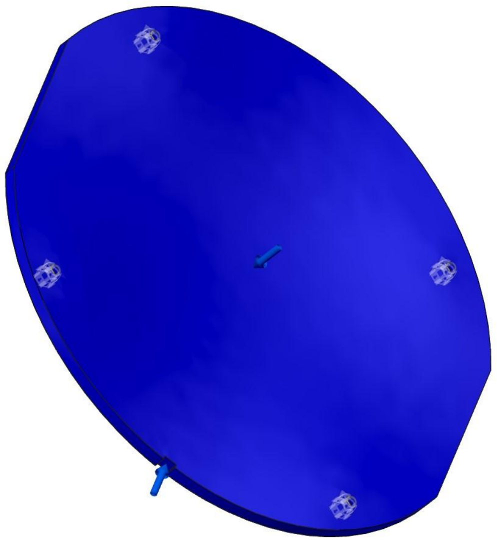


Fig.18 Top plate simulation
Load: 10 kg uniform distribution and 10 kg inertia force
Constrained on support holes
Material: ABS

Full list of components:

Name	**Model**	**Quantity:**
Base Plate	Own design	1
Middle Plate	Own design	1
Top Plate	Own design	1
Ball bearing	60355K505_Ball Bearing	2
Wheel	Own design	2

Sleeve	Spacer sleeve steel-d1-M8 L-160mm SW-13 PlacoTec	8
Sleeve bolt	AS 1427 H - Metric M6 x 20	16
Caster wheel	5674K1	2
Caster wheel bolt	AS 1427 Z - Metric M5 x 20	4
Caster wheel nut	AS 1112 (2) - Metric M5 Type 5	4
Stepper motor	Nema 23 stepper motor	2
Stepper motor bolt	AS 1427 H - Metric M5 x 20	8
Stepper motor nut	AS 1112 - Metric M5	8
Sensor bracket	Ultrasonic sensor silicon bracket	5
Sensor bracket bolt	AS 1427 H - Metric M3 x 20	10
Sensor bracket nut	AS 1112 (4) - Metric M3	10
18650 Li-Ion battery	Samsung INR18650-35E	8
5V Step down converter	Pololu D24V50F5	1

Stepper Motor Controller	Amis 30543	2
Stepper Motor	JK57HS56-2804	2
Ultrasonic Sensor	DFRobot SEN0307	5
Camera	Raspberry pi HD camera module	1
Raspberry Pi 4b	N/A	1

Table.1 Component list

Electronic Design:

Battery Pack:

The power unit is composed of two 3,6V modules with a maximal amperage of 32A. Each module is composed of four 18650 Li-Ion batteries connected in parallel. The two modules are joined together in series to finalise the power unit of 7.2V and over 230W max output power.

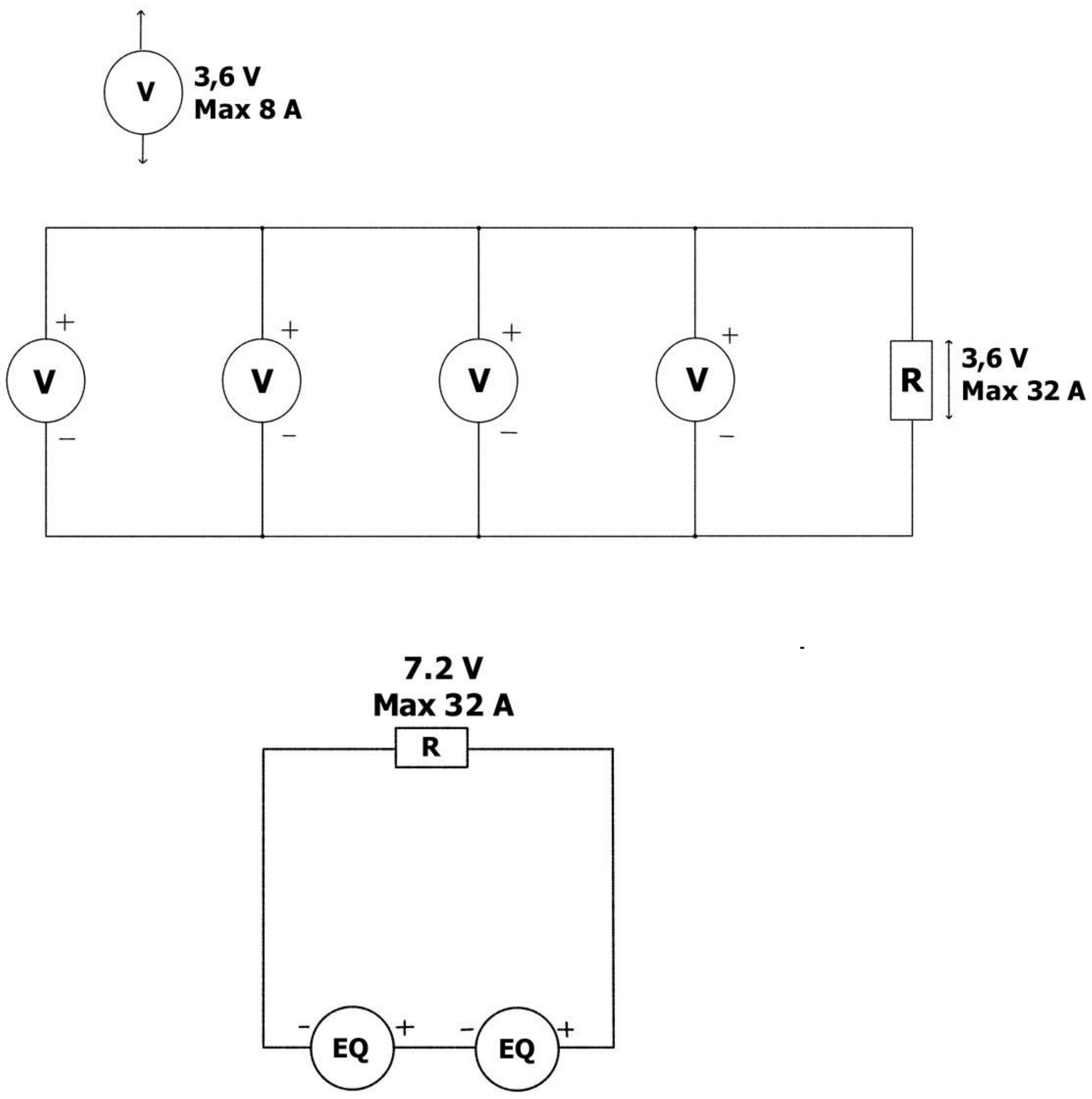


Fig.18 Power Unit scheme.

General electric scheme:

The 7,2V power unit directly powers the stepper motor controller, therefore also the two stepper motors,and feeds into a 5V step down transformer. The 5V circuit powers the Raspberry Pi 4b, logical components of motor controllers and ultrasonic sensors (not shown on the scheme below).

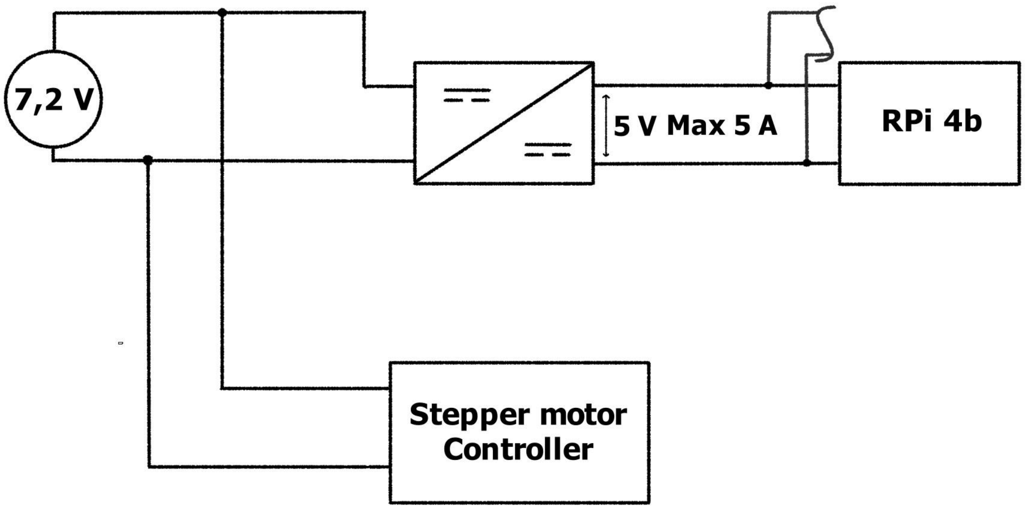


Fig.19 general electric scheme

Electronic components connections scheme:

On the fig. below is the scheme of all necessary connections between sensors, motor controllers, motors and power circuits.
Notice that not all ultrasonic sensors are present on this scheme for illustrative purposes. The other sensors can be connected in the same pattern as the rest. Neither the power requirements, nor the limited number of GPIO pins would prevent one to attach another three sensors or as much as needed.

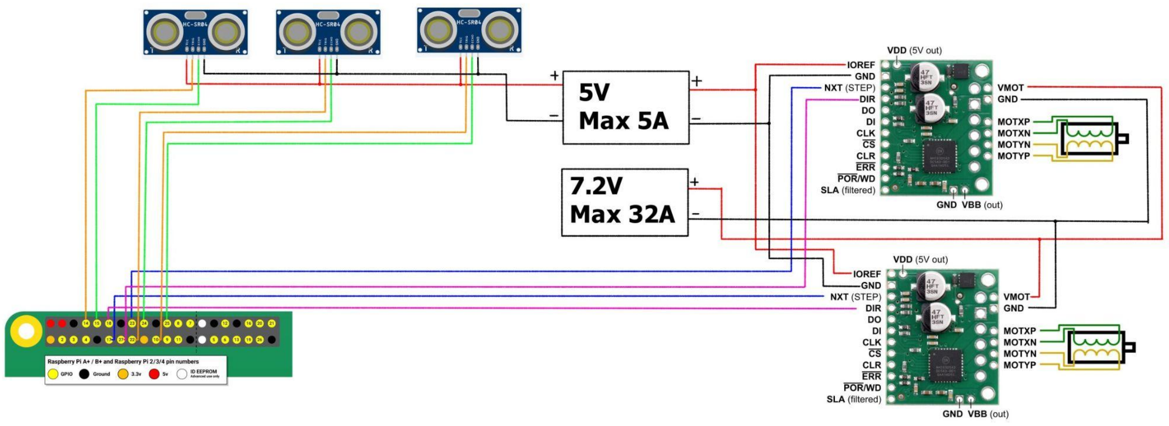
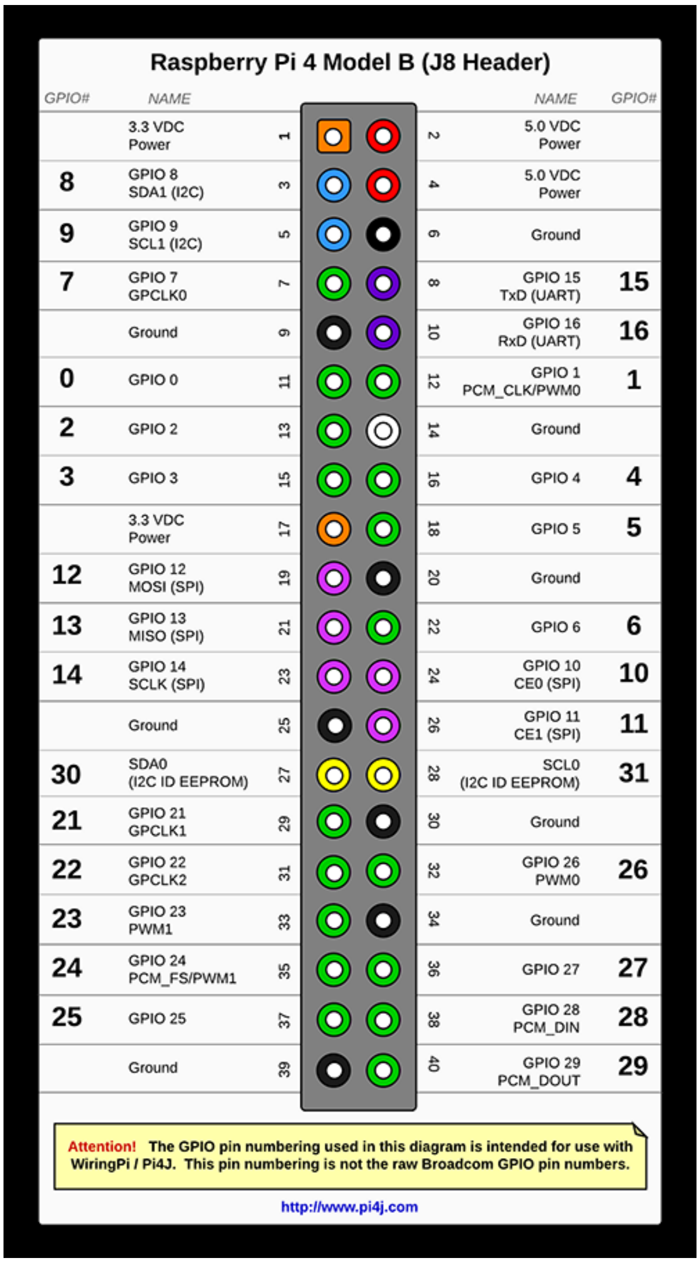


Fig.20 Electronic components connections scheme

For reference:



Raspberry Pi 4 Model B (J8 Header)

GPIO#	NAME			NAME	GPIO#
	3.3 VDC Power	1	2	5.0 VDC Power	
8	GPIO 8 SDA1 (I2C)	3	4	5.0 VDC Power	
9	GPIO 9 SCL1 (I2C)	5	6	Ground	
7	GPIO 7 GPCLK0	7	8	GPIO 15 TxD (UART)	15
	Ground	9	10	GPIO 16 RxD (UART)	16
0	GPIO 0	11	12	GPIO 1 PCM_CLK/PWM0	1
2	GPIO 2	13	14	Ground	
3	GPIO 3	15	16	GPIO 4	4
	3.3 VDC Power	17	18	GPIO 5	5
12	GPIO 12 MOSI (SPI)	19	20	Ground	
13	GPIO 13 MISO (SPI)	21	22	GPIO 6	6
14	GPIO 14 SCLK (SPI)	23	24	GPIO 10 CE0 (SPI)	10
	Ground	25	26	GPIO 11 CE1 (SPI)	11
30	SDA0 (I2C ID EEPROM)	27	28	SCL0 (I2C ID EEPROM)	31
21	GPIO 21 GPCLK1	29	30	Ground	
22	GPIO 22 GPCLK2	31	32	GPIO 26 PWM0	26
23	GPIO 23 PWM1	33	34	Ground	
24	GPIO 24 PCM_FS/PWM1	35	36	GPIO 27	27
25	GPIO 25	37	38	GPIO 28 PCM_DIN	28
	Ground	39	40	GPIO 29 PCM_DOUT	29

Attention! The GPIO pin numbering used in this diagram is intended for use with WiringPi / Pi4J. This pin numbering is not the raw Broadcom GPIO pin numbers.

http://www.pi4j.com

Fig.21 Raspberry Pi 4B GPIO pinout.

Power draw and duration of operation calculations:

In general $Battery\ Life\ (Bl) = \frac{Battery\ Capacity\ (Bc)}{Load\ (Il)}$ units [h]=[mAh]/[mA] eq.1

$Bc = 3500mAh \times 4 = 14Ah$

Load:

1. Rpi 4b max draw 6,4W
2. Ultrasonic sensors 0.1W per sensor ~ less than 1W for all sensors
3. Stepper motor 13.44W~ 14W per motor

Together:
Step down converter efficiency in documentation is over 90%

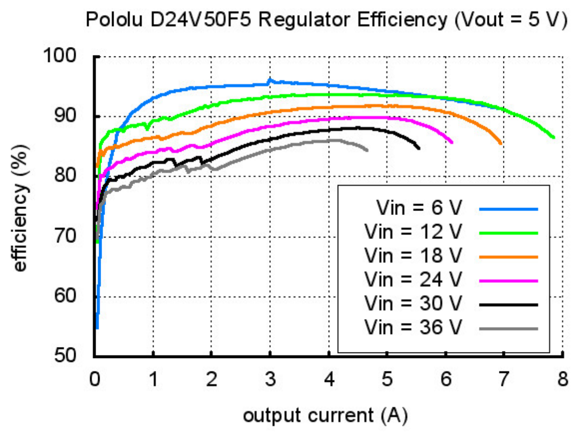


Fig.22 Efficiency chart

We assume overall efficiency of the whole electric circuit to be E = 85%
Total Power draw:
$(Rpi(6,4W)\ +\ 6\times sensor(0.1W)\ +\ 2\times motor(14W)\ =\ 41,18W \sim 42W$ eq.2
$Load\ (Il\ )\ =\ Power\ Draw\ (42W)/Unit\ Voltage(7,2V)\ =\ 7A$ eq.3

Now substituting into eq.1 we get a Battery life of min 2h of full power draw.

Software Design:

Attached to this report is the flowchart of all functions explaining which function depends on which function, library or data file.

Also attached is the whole code required to present a proof of concept on any pc with python3 and required libraries. Only thing required to change in code for operation on any PC is to change the path of the camera in RobotJourney. The code has been written with clarity and modularity in mind. The naming of functions, variables and code comments hopefully describe the functioning of the code unambiguously.

1. Principle of operation

The goal of the project is to build a system for controlling the food delivery robot operating in an indoor restaurant. The user chooses from GUI the starting point and the endpoint of the robot path. The robot navigates thanks to the lines on the floor of the restaurant and QR codes on the intersection of each line. Robot is equipped with a camera, and thanks to image processing functions described below and OpenCV library it is able to detect the line as well as to recognize the qr code and obtain necessary information from it. The software uses a predetermined map of the restaurant.

2. SimpleGuiMod:

It is least complicated mode out of these three. It opens the window in which we can insert the starting point and end point. GUIi has been created thanks to PySimpleGUI library.

3. HowToGetThere:

In this module we have the function which takes as arguments previously returned by the SimpleGuiMod. Then it opens the file with a map of all paths. Map is the list of all paths possible to take by the robot. One path is the list of intersections which is a tuple which has two values: intersection ID and angle of turn that needs to be performed on that intersection.

Based on the parameters from the user it chooses which path to use. As proof of concept we chose to have one base and 3 tables and one intersection. Robot can travel from the base to each table and from each table to the base. For now it cannot travel between the tables but it is only a matter of expanding the map and HowToGetThere library of possible routes. Logic will stay the same.

4. RobotJourney:
This is the most complicated part. Up to this point the purpose of the modules was just getting input data that will be used by the RobotJourney mode. Thanks to that mode our robot can move along the chosen path and interact with the environment around it.
Here we have the RobotJourney function which takes as argument path chosen previously.

At the beginning we define code frequency which tells us how often our code refreshes. Then we define the time after which the robot will initiate the findLine function if the line has been lost. It is one second in this case. Thanks to that our robot won't stop moving if it loses a path for less than a second which naturally happens due to glitches. Then we have camera settings. In the next part, code unpacked the path object into two lists. One for the qr code IDs (intersections and path ends) and one for the angles of turn. The actual process of controlling the robot starts in a while loop. Now we start from getting the image from the camera. After that we gather information needed for steering the robot. Cx and Cy are the coordinates of the middle of the object (line detected). In order to get the object we first convert the image from the camera to grayscale and then we use a series of thresholding and blurring operations to extract the line. Cx is the parameter that is important for steering the robot. For controlling two wheeled robots following the line only x-axes is relevant.

Additionally we use the CheckForCollison function. It uses data gathered from the ultrasonic sensor. If the external object is too close to the sensor, the function returns true. If collision equals true the robot stops for half a second and we overwrite coordinates Cx and Cy as 0. In that case the robot thinks that he did not detect the line. It causes the robot to stop for 15 seconds and then it starts looking for the line again. The same part of code is used if the robot actually loses the line.
What if our robot found the line and there was no collision. We set the speed of the robot. Then based on the cx parameter and speed parameter whileOnline function steers the robot according to the following rule: „If the centre of the object is to the right, turn right, if the centre of the object is to the left, turn left, if the centre of the object is in the middle continue the journey". This is the line following part.

Then we have part responsible for navigating by the qr codes. At first we are looking for the qr code using the function QR_Read. This function allows us to find the orientation of the QR code in space (we draw 3 orientational points of the qr code and calculate the angle) and to read the information included in this qr code. If the ID from encountered code matches the ID of expected IDs, this ID is saved in the list of previously encountered intersections. It is important because it eliminates the problem with reading the same ID twice due to buffering errors. When the robot sees the proper ID for the first time it saves it in a list, thanks to that if it sees the same ID two Times in the row it ignores the second ID and does not stop in the same place doing the same instruction all over again. If the code ID matches expected ID and it is not the same code as previously and the robot saw that the amount of intersections it has encountered is the number of intersections it was supposed to encounter it means that it is at the end of the path and needs to stop.

Full resolution figure attached with rapport.
Orange bubbles: python libraries needed.
Blue rectangles: functions developed by us and attached with this report.
Purple rhomboid: Map data structure describing the restaurant layout.

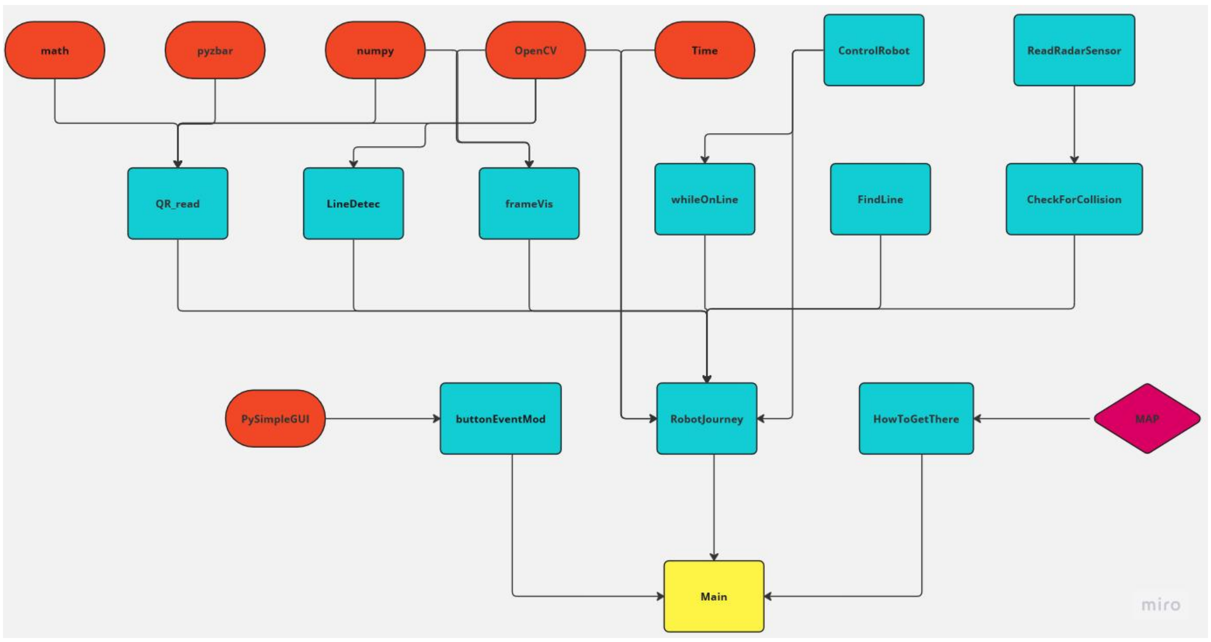


Fig.23 Functions dependencies

Demonstration:
https://drive.google.com/file/d/18mu3qNN8VbPbW8ydXH--WKKcO7tEQdbm/view?usp=sharing

Sources:

Fig.1
https://www.edikson.com/product-page/keenon-dinerbot-t5-mistrz-dostawy?gclid=EAIaIQobChMIkoWa0eyQgAMV9AWiAx2PAgEjEAQYASABEgKHUPD_BwE

Fig. 2
https://www.procobot.com/produkt/bellabot/

Fig. 3
https://archiwum.allegro.pl/oferta/robot-kelner-4s-sister-xii-i8418928502.html

Fig. 21
https://pi4j.com/1.4/pins/rpi-4b.html

Fig. 22
https://botland.com.pl/przetwornice-step-down/2754-d24v50f5-przetwornica-step-down-5v-5a-pololu-2851-5903351242752.html

Attachments:

With this report are attached:

- All code required for functioning of the robot
- All CAD models and drawings required for manufacturing
- Strength analysis reports for all crucial project parts
- Morphological analysis chart