



TALLINN UNIVERSITY OF TECHNOLOGY

SCHOOL OF ENGINEERING

Department of Electrical Power Engineering and Mechatronics

# **DEVELOPMENT OF DEVICE FOR NAVIGATING PEOPLE WITH VISUAL IMPAIRMENTS**

## **NÄGEMISHÄIRETEGA INIMESTE NAVIGEERIMISSEADME ARENDAMINE**

MASTER THESIS

Student: Anastasiia Iusupova

Student code: 194819EV

Supervisors: Dmitry Shvarts, research scientist,  
Pavel Kovalenko, associate professor

Tallinn 2020

**AUTHOR'S DECLARATION**

Hereby I declare, that I have written this thesis independently.  
No academic degree has been applied based on this material. All works, major viewpoints, and data of the other authors used in this thesis have been referenced.

"....." ..... 20....

Author: .....  
  
/signature /

Thesis is in accordance with terms and requirements

"....." ..... 20....

Supervisor: .....  
/signature/

Accepted for defense

"....." .....20... .

Chairman of theses defense commission: .....  
/name and signature/

**Non-exclusive License for Publication and Reproduction of Graduation Thesis**

I Iusupova Anastasiia, (date of birth: 5 September, 1996)

1. grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis "Development of device for navigating people with visual impairments",

supervised by Dmitry Shvarts and Pavel Kovalenko,

to be

1.1. reproduced for the purposes of preservation and electronic publication, incl. to be entered in the digital collection of TUT library until expiry of the term of copyright;

1.2. published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of TUT library until expiry of the term of copyright.

2. I am aware that the author also retains the rights specified in clause 1.

3. I confirm that granting the non-exclusive licence does not infringe third persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

\_\_\_\_\_ (signature)

\_\_\_\_\_ (date)

**Department of Electrical Power Engineering and Mechatronics**

**THESIS TASK**

**Student:** Iusupova Anastasiia, 194819EV  
**Study program,** MAHM02/18  
**main specialty:** Mechatronics  
**Supervisor(s):** Dmitry Shvarts, research scientist, TalTech University,  
Pavel Kovalenko, associate professor, ITMO University  
**Consultants:** Mikhail Sachkov, senior lecturer, ITMO University

**Thesis topic:**

(in English) Development of a device for navigating people with visual impairments

(in Estonian) Nägemishäiretega inimeste navigeerimisseadme arendamine

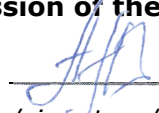
**Thesis main objectives:**

1. Formulate the design and functions of the developed device based on existing solutions and methods;
2. Based on the selected methodology, implement the proposed functions of the device, its design, and algorithm of the work;
3. Test developed methods, construct a prototype of the device and identify further points for development.


**Thesis tasks and time schedule:**

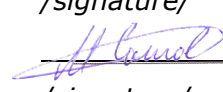
No	Task description	Deadline
1.	Perform background research on previous work	30/11/2019
2.	Formulate methodology and techniques, identify necessary hardware and software	31/12/2019
3.	Prototype and algorithm of the work development	29/02/2020
4.	Prototype testing and optimization	01/04/2020
5.	Conclusion writing, completion of the research	20/05/2020

**Language:** English **Deadline for submission of thesis:** " \_\_\_\_ " \_\_\_\_\_ 2020

**Student:** Anastasiia Iusupova  " \_\_\_\_ " \_\_\_\_\_ 2020  
/signature/

**Supervisor:** Dmitry Shvarts \_\_\_\_\_ " \_\_\_\_ " \_\_\_\_\_ 2020  
/signature/

**Supervisor:** Pavel Kovalenko  " \_\_\_\_ " \_\_\_\_\_ 2020  
/signature/

**Consultant:** Mikhail Sachkov  " \_\_\_\_ " \_\_\_\_\_ 2020  
/signature/

**Head of study program:** Mart Tamre \_\_\_\_\_ " \_\_\_\_ " \_\_\_\_\_ 2020  
/signature/

# CONTENTS

PREFACE .....	7
List of abbreviations and symbols .....	8
INTRODUCTION.....	9
Research motivation .....	10
Research objectives .....	10
Thesis structure .....	11
1.    LITERATURE REVIEW AND BACKGROUND .....	12
1.1.  Literature and existing approaches review .....	12
1.2.  Literature and existing approaches analysis .....	16
2.    RESEARCH METHODOLOGY.....	18
2.1.  Proposing solution .....	18
2.2.  Hardware and software selection .....	19
2.2.1. Hardware selection .....	19
2.2.2. Software selection .....	20
2.3.  Designing concepts .....	21
3.    THE DEVICE DEVELOPING BASED ON THE SELECTED METHODOLOGY.....	23
3.1.  Objects recognition algorithm.....	23
3.1.1 Objects recognition testing analysis.....	28
3.2.  Distance measurement and mapping using ultrasonic sensor .....	35
3.2.1. Distance measurement and mapping analysis .....	44
3.3.  Description of the device operation algorithm .....	46
3.3.1. Operation algorithm testing and analysis .....	49
3.4.  Electrical scheme developing .....	50
3.4.1. Electric scheme testing results .....	54
3.5.  2D and 3D modelling.....	55
3.5.1. 2D and 3D modelling results analysis .....	58
3.6.  Device developing summary.....	59
4.    DISCUSSION.....	60

4.1. Limitations.....	60
4.2. Future improvements .....	61
SUMMARY.....	62
LIST OF REFERENCES .....	63
APPENDICES .....	66
GRAPHICAL MATERIAL .....	75

## **PREFACE**

The topic of the master thesis is proposed by the author. The main goal of this project is to develop a prototype of the device that will help people with visual disabilities in navigation and thereby simplify their lives. The project is the logical completion of a double-degree program in master studies, the research was carried out with the participation of Tallinn Technical University, Tallinn, Estonia and the ITMO University, St. Petersburg, Russia.

This project is a continuation of the bachelor's work of the author and article [1]. However, the master's thesis is devoted not only to the processing of ultrasonic signals but also to work with images, their processing and improving the device's design.

The project was supported by the FASIE and the Grant for university students located in St. Petersburg, graduate students of universities, industry and academic institutions located in St. Petersburg.

The author would like to express his deepest gratitude to supervisors Dmitry Shvarts and Pavel Kovalenko for supporting this topic and also consultant Mikhail Sachkov for helping with research implementation.

A special thanks to my family and friends for moral support.

Keywords: navigation device, image processing, signal processing, people with visual impairments.

## **List of abbreviations and symbols**

CAFFE – Convolutional Architecture for Fast Feature Embedding

GPIO – General-Purpose Input/Output

LMDB – Lightning Memory-Mapped Database

PWM – Pulse Width Modulation

UDS – Ultrasonic Distance Sensor



## **INTRODUCTION**

Everyone has the right to receive knowledge, information about the world and exchange it. A person perceives with the help of vision >70% of the information, but people with visual impairments should fill this gap with the help of hearing, smell and touch. Moving around the city for visually impaired people, given the transport, road conditions, many obstacles, and traffic, is quite problematic. The cane can detect obstacles from no more than a meter, which imposes greater restrictions. Also, these people have psychological features and it should be considered that the device should not distract the user, do not close the auditory canal and it should not allow dangerous situations.

The aim of the research work is the development of a prototype designed to ensure the safe movement of visually impaired people in the urban environment and indoors, analysis of the developed system, writing the algorithm of work. During development, it is required to create not only an effective device but also convenient to use, including by elderly people and children.

The relevance of this work is to develop a device with improved technical and economic parameters in comparison with analogs, improving existing methods. There is no need to develop devices with already achieved characteristics. The price of the developing device is also important.

An expected result of the research is the creation of a prototype device for visually impaired people, which builds a map of the room around the user, determines the type of objects and the distance to obstacles. The obstacle here is the signal obtained by one or more distance sensors and be relevant after post-processing. This signal should correspond to distances in the range of 10-400 cm.

The prototype should define such types of objects: vehicles (bus, truck, bicycle, car), living objects (cat, dog, person), household objects (cup, fork, knife, spoon, chair, dining table, bed, cell phone, microwave, sofa, sink, monitor), and outdoor objects (traffics light, bench). The device is a multi-layer waterproof vest, inside of which electronic components and sensors will be located.

The user must receive information about the obstacles described above, the distance to them, the direction and battery level of the device. Detected objects should be determined with a probability of >90%, the weight of the device (including the vest) should not exceed 3 kg (about 3-5% of the person's weight are the comfortable weight

for frequent wearing without experiencing fatigue, for example, a person weighing 60 kg was chosen); continuous operation of the device should be >3 hours (in big cities the one-way road is capable to take 1 hour and more, so the device should provide at least round trip for user).

## **Research motivation**

According to statistics data [2] [3] [4], there are about 36-39 million blind people and 250-285 million people with visual impairments in the world. By 2020, the number of blind people in the world can increase to 75 million people according to the United Nation data. All these people need support, training, and devices that improve their lives. People with visual impairments face difficulties every day, and there is not always a person nearby who can provide assistance.

Despite the existence of many solutions and devices to help people with visual impairments, they all have advantages and disadvantages, and every device is aimed at performing a specific function.

Given the growing number of people with visual impairments around the world, the development of increasingly advanced assistive devices for them is especially relevant and in demand.

## **Research objectives**

To conduct the research, the following steps must be taken:

- analyzing current solutions and literature overview;
- determination of the concept of the device;
- selection of sensors and electronic components, selecting software;
- development of a system for processing images and data from sensors, distance, and mapping algorithm development;
- algorithm of the work development;
- device design development.

## **Thesis structure**

This section describes the chapters' contents of this thesis.

The introduction provides aims, motivation, and objectives of the research. The problem to be solved is described here.

Chapter 1 contains a current situation in the selected area, existing solutions, working approaches and literature overview. Analyzing the background of the solving problem, it can be possible to justify the methodology of the research.

Chapter 2 describes the methodology of the research, justifies the software and hardware choice. Also, the device's design concepts are being formed in this section.

Chapter 3 comprises an explanation of developing a previously formulated approach. This includes the image recognition algorithm, distance measurement algorithm, electric scheme development, 3D-modelling and analysis of every obtained result.

Chapter 4 contains a discussion of the research, possible limitations, and future improvements formulation.

The summary concludes the work is done and notes which goals of the set were achieved.

# 1. LITERATURE REVIEW AND BACKGROUND

This chapter describes the current situation in the area of navigation devices for people with visual impairments. Literature sources, that reflect the methods used in navigation, object recognition, mapping and measurement using ultrasonic sensors, will be considered here also.

Also, in this chapter, an analysis of existing analogs will be summarized, their advantages and disadvantages will be discussed and their properties that can be improved and used in the developing device below will be noted. Based on the considered literature sources, the research methodology and the basic principles of the developing device will be determined.

## 1.1. Literature and existing approaches review

Literature sources were selected from the following areas: devices for people with visual impairments, recognition of objects on the street and indoors, building a map and measurements using ultrasonic sensors.

- Ultrasonic measurements and mapping:

4 articles on this topic were considered that offer various solutions, including the use of Gray System Theory [5], Uncertainty Calculus [6], preprocessing with Kalman filter measurement technique [7] and Probabilistic Mapping [8]. Article [5] considers the use of the theory of Gray Systems to create environmental maps. The measuring angle of the ultrasonic sensor is  $30^{\circ}$ , and when an object is detected, it is not known in which part of the sensor's workspace the object is located. This option is called uncertainty. The determined algorithm creates an uncertainty model with the corresponding gray values. With updating the map, the previous gray values are compared with the current ones, if they coincide - the weight of this value increases, it is more reliable. The results obtained by this method are quite "blurred", since the map does not have clear boundaries, but is essentially a gradient of values from less to more reliable. The work [7] describes the most easily implemented method of measuring and constructing a map: the values obtained from the ultrasonic sensor are first processed by the threshold method (emissions above a certain threshold value are cut off), then processed by the Kalman filter. These data are transmitted and stored in the memory of the microcontroller, individual discrete values are interconnected in

a line, and thus a map is obtained. This method can be used if the object is not moving very fast. Ilze Andersone in the article [8] represents the implementation of a probabilistic method of constructing maps. In this algorithm, the leveling of uncertainty occurs with the help of positive readings (there is an obstacle) and negative (absence of obstacles). This method gives accurate results and low dispersion of values; however, it requires a long data accumulation time to build a map. The work [6] describes 3 methods of uncertainty calculations - probability theory, fuzzy logic, and fuzzy measures. The authors proved that the probabilistic approach is error-prone and ejection sensitive. The authors assume that fuzzy logic is the best method out of those considered. Fuzzy logic uses fuzzy sets, unlike the probabilistic theory. Fuzzy measures are an improvement on the previous method, which uses not fuzzy sets, but averaging operators.

- Object recognition for navigation people with visual impairments:

Some approaches for object recognition using a camera were considered. The article [9] describes methods for stairs and pedestrian crosswalks recognition. The authors suggest method detecting stairs with 91% accuracy and pedestrian crosswalks with 95% accuracy. The method contains five steps: edge detection of an RGB image, Hough transformation calculation, peaks in the Hough transform matrix calculation, extraction lines, and parallel lines group detection. The work [10] discusses walls, doors, stairs and floor real-time recognition using color-depth data acquired by the Microsoft Kinect sensor. The drawback of this method is the inability to recognize depth data if light conditions are too strong. Authors of paper [11] were implemented the real-time algorithm for stair detection and modeling. This method overcomes the possibility of single-step detection.

Currently, there are many devices that help people with visual impairments. These devices are very diverse: some are designed to help in reading documents and books, some help to identify products in stores, and others contribute to comfortable and safe movement. Among the devices for comfortable movement, there are also many differences: types and place of attachment, method of action, range, etc.

Some examples of devices for comfortable and safe movement:

- Cane based devices:

Cane based devices are located on the white cane and usually use simple ultrasonic distance measurements. [4] [12] [13] [14]

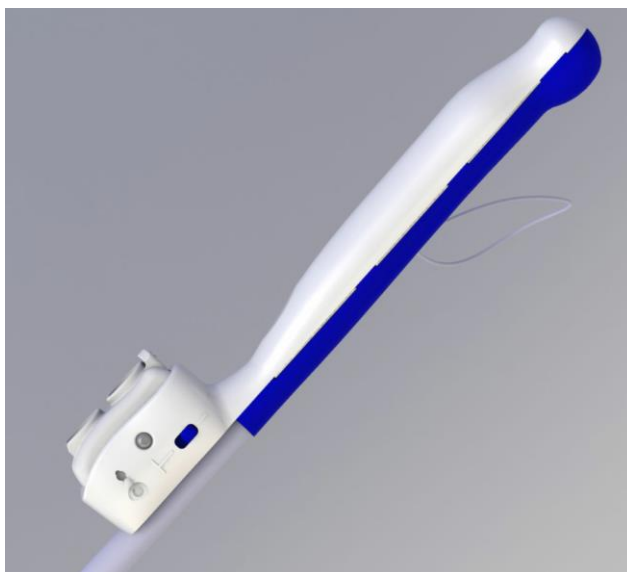


Figure 1 – Cane based navigation device's appearance [13]

Such devices are limited in range (cane length) and the amount of information received.

- Vests:

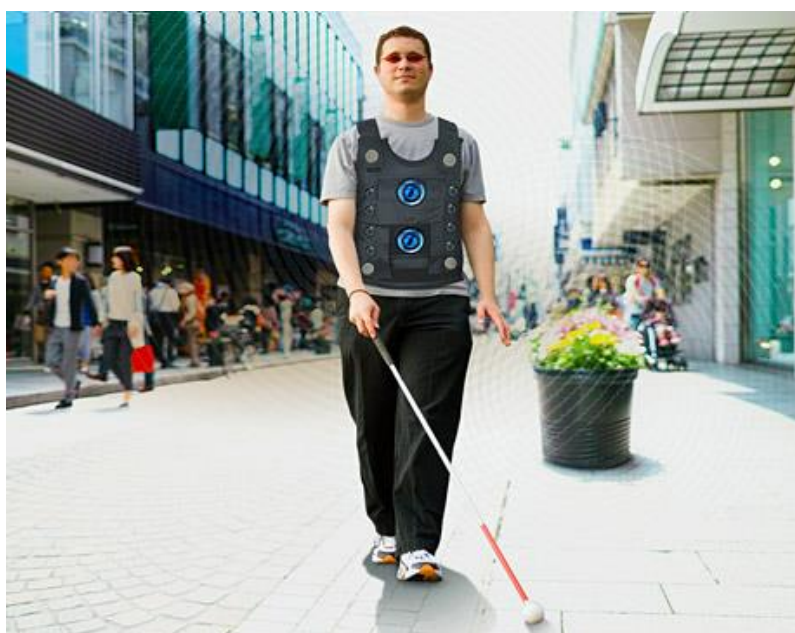


Figure 2 – Vest based device: prototype [4]

Vests based devices are very comfortable to use, it is possible to locate a lot of sensors using body area. However, all existing developments of such devices are now only at the prototype stage. [4] [15] [16] [17]

- Hat-based devices:

There are still other examples of this type of device, but they are all quite similar. The disadvantage of hat-based devices is their size and inconvenience of use. [4] [20]

- Glasses:

Glasses-based devices are also commonly used. Some use miniature cameras to detect objects, others use ultrasonic sensors to determine the distance to obstacles [21]:

- Shoes:

In my opinion, a shoe-based design [4] [22] is the least convenient to use, since the user must either always wear the same pair or fasten it to another pair before going out. In addition, such devices should be especially protected from moisture and dirt.

- Wrist/palm modules:



Figure 3 – wristband device [23]

As in the previous examples, such devices use cameras and ultrasonic sensors. Devices of this type are inconvenient to use together with a cane (a white cane is not only providing navigation, but also it is an identification mark of people with visual impairments), or when used together, both hands will be occupied. [4] [23]

- Chest devices:

These devices are quite similar to vests, but are more compact and have complex mounts, for fixing which you may need the help of a sighted person. [4] [24]

## 1.2. Literature and existing approaches analysis

Using information from the considered sources, the author plans to use some methods for constructing a map and recognition of objects. The easiest way to implement the map building process is preprocessing with the Kalman filter measurement technique. Since this method is quite simple to implement and effective from the point of view of computer calculations, it can be used in microcontrollers using and mobile devices. If possible, a probabilistic method will also be tested.

For the developing device, the definition of objects such as stairs, floors, doors, windows, puddles, and pits is an important property. The methods for recognizing stairs, doors, and floors, considered in literary sources, will become the basis of the algorithm of the device being developed, which must be improved to determine another type of object.

The considered methods have advantages and disadvantages. To evaluate the most suitable method that should be used in the developing device, the author has evaluated existing analogs, considering some of their properties:

- The convenience of use. The most convenient to use can be considered vests and chest-based devices since in this case, the hands remain free. Hat-based, glasses and smart shoes are limited in use in different weather conditions and seasons;
- The weight. Bracelets (<150 gm), glasses (<300 gm) and devices that are attached to a cane (<500 gm) have the least weight. As was mentioned earlier the convenient weight is 3-5% of the user's weight. So, the device's weight should not be more than 3 kg;
- The size. Bracelets (<10 cm<sup>2</sup>) and (<20 cm<sup>2</sup>) glasses have the smallest size;
- Identification of obstacles and/or measuring of distance to them. Some devices can only determine distances to obstacles - cane-based devices, bracelets, shoes. Glasses are mainly designed to recognize text or objects. Vest-based and chest-based can combine several functions;
- Ability to arrange multiple sensors and batteries. For the device to perform several functions, and enough area is required. The greatest functionality can have vests (area for arranging sensors >0.5 m<sup>2</sup>) and chest-based device (>0.2 m<sup>2</sup>);



- The necessity to protect the device from moisture/dust /dirt. All devices must be protected from moisture, but more prone to failure due to environmental conditions are hat-based and smart shoe devices;
- The complexity of mount. Visually impaired people are sometimes unable to cope with complex mounts without someone else's help. So, chest-based and smart shoe devices may have too complicated mounts.

Summarizing all the above items, it is possible to say that vests have the least number of drawbacks - they can be made of waterproof materials, have a large surface for placing sensors, and they can conveniently wear over clothing. However, it is important to control the weight of such a device.

## **2. RESEARCH METHODOLOGY**

This chapter will focus on selection resources for the device's development and explanation of the proposed solution. The first section will explain in detail proposing a solution for prototype implementation. The second and third sections will cover hardware and software selection and analysis. The fourth and fifth sections will account for designing concepts and their analysis. The final section will analyze selected features from previous sections.

### **2.1. Proposing solution**

As was analyzed in the previous chapter, the most appropriate base form for this type of device is the waterproof vest with electronic components inside/on it. The main options of the device are the mass (not more than 3 kg) and the working time (more than 3 hours). These two parameters should be taken into consideration during design and algorithm development. The vest should be layered with the waterproof upper layer to protect electronics inside. The important feature is the mount because it is necessary to provide easy fixation for the user that can only use their hands without seeing the mount.

The device has two main tasks: building the map of the indoor environment and determining the object type that has been mentioned earlier. For building the map device will use a set of ultrasonic sensors and it is necessary to obtain separate points from every sensor, collect them, post-process the data, and plot them on. The sensors are not ideal, every measure has errors and noise. To obtain a correct indoor map it is necessary to process measurements using filters.

For object recognition, the neural networks, classification methods will be applied. In this work, the objects to be recognized are vehicles (bus, truck, bicycle, car), living objects (cat, dog, person), some household objects that might be important for the user (cup, fork, knife, spoon, chair, dining table, bed, cell phone, microwave, sofa, sink, monitor), and outdoor objects (traffics light, bench). After recognition of the obstacle, the device should inform the user about the distance to than obstacle using vibration and/or audio signal. Thereby, for device designing it is necessary to select appropriate sensors, controller and other components with sufficient accuracy and performance, then determine software for algorithm writing, 3D-modelling and testing the developed model.

## 2.2. Hardware and software selection

### 2.2.1. Hardware selection

The hardware part's base of the developing device is the controller. As it is necessary to ensure high performance for object recognition, the controller should be powerful enough. Thereby the Raspberry Pi board computer is the most appropriate choice for the tasks to be solved.

The next step is the camera selection. There's no need to use a camera allowing to make high-quality images since the selected objects haven't sophisticated form and exiguous size.

For building the map ultrasonic sensors are needed. For this task, the HC-SR04 ultrasonic ranging module is quite suitable, because of its simplicity, low power consumption and acceptable accuracy ( $\pm 1-3$  cm).

For the power supply, the Li-Ion accumulator with 5V working voltage will be used, as the instructions for the controller and camera require this level of the input voltage.

To inform the user about obstacles and objects two types of components are needed: vibration motors to send vibration signals and/or audio transmitter to send audio signals.

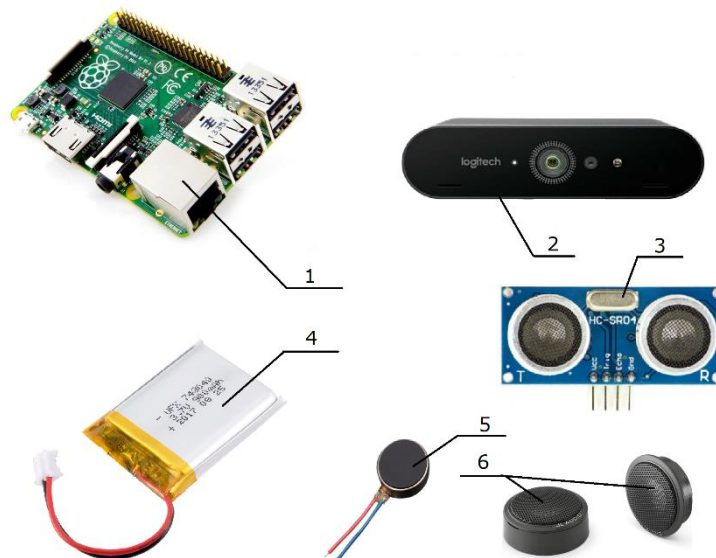


Figure 4 – main electronic components for developing device (1 – Raspberry Pi, 2 – USB camera, 3 – ultrasonic sensor HCSR04, 4 – Li-Ion accumulator, 5 – vibration motor, 6 – audio transmitter)

### **2.2.2. Software selection**

During the development of this project, the software is provided for the development of 3D models of the device, for electronic circuit development, for modeling and control algorithm writing.

There are a lot of programs for 3D modeling, drawing and properties analysis, but 3D models will be produced in "Solidworks 2016" software, as it allows provide the strength and other analyzes, and the author is well acquainted with this program.

For modeling the electrical board and electrical connections the open-source "KiCad EDA" program will be used, as this program has a free license and simple interface for the user with small experience in electrical board design.

And the last necessary type of software is the IDE software for coding in Python because in this program the Raspberry Pi will be used. Here the open-source "Anaconda IDE" will be a good choice since it has a lot of included libraries and packets inside.

## 2.3. Designing concepts

As was described above, the device is the waterproof vest with electric components inside and on it. The design should ensure uniform distribution of the electronics along the body, comfortable wearing, and fastening. It is also important to properly position the camera and ultrasound sensors to obtain adequate data and capture the entire workspace. The workspace here is the space in front, on the right and on the left of the user:

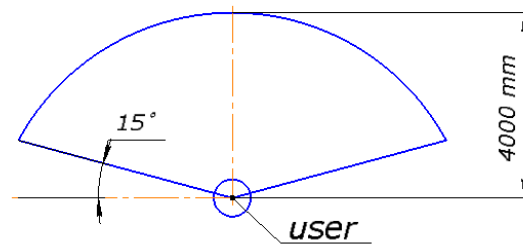


Figure 5 – user's workspace

As mentioned above, the location of the sensors and the camera should ensure the capture of the entire working area, so the proposed scheme is as follows:



Figure 6 – main components proposing location (1,2 – audio and vibration modules, 3 – camera, 4 – Raspberry Pi, 5 – ultrasonic modules, 6 – accumulator)

Also, all elements should be located in silicone cases to protect against shock and liquid, the controller and batteries should be in the inner layer of the vest, and the camera and ultrasonic rangefinders should partially be on the surface.

## **2.4. Selected methodology analysis**

This chapter was focused on the description of proposing solution including hardware and software selection, designing concept definition. As a result of the selected options, the main principles of the algorithm were determined – the mapping using ultrasonic sensors will be implemented using filtering. The object recognition will be realized using neural networks, classification and statistics methods.

The software was chosen according to its usability, author's acquaintance and open source license possibility. The hardware, in turn, was selected according to its accuracy, performance, and cost.

In the section about designing concepts, the main options about the device appearance were described: the workspace, the main electronics' location and the layered structure of the vest.

### **3. THE DEVICE DEVELOPING BASED ON THE SELECTED METHODOLOGY**

This chapter is devoted to developing the device itself. The developing includes the following topics: object recognition algorithm developing and its analysis, the distance measurement and indoor mapping algorithm developing and analysis, the 2D and 3D modelling, the electrical scheme developing and operation time calculation.

Following this chapter, the operational prototype should be realized and the algorithms mentioned above should be tested on it. The prototype should be based on selected previously electrical parts.

#### **3.1. Objects recognition algorithm**

The most productive and efficacious method for object recognition is using deep learning algorithms. Deep learning is a technique for machine learning focused on the examples. Deep learning methods use standard neural network architectures. The difference between simple neural and deep neural networks is the number of hidden layers - neural network (NN) usually contains 2-3 of them, and deep NN, in turn, may have at least 150 hidden layers. [25]

The deep learning models require pre-processing steps, such as data labeling and network architecture description. In this work, the data labeling means the differentiation of the dataset by the specific type manually.

One of the most popular types of deep NN is the convolutional neural networks (CNN). CNN uses 2D convolutional layers that they're well suited for image recognition. [26] The usage of the CNNs allows the elimination of the manual feature extraction - classification of the images. This type of neural network extracts data from the images directly. The essential point here is that relevant features aren't pre-trained, they are learned during the network training. This detail makes deep learning algorithms highly accurate for machine vision and classification of the objects.

Since the Python programming language has been used for this work, the architecture can be set using a CAFFE (Convolutional Architecture for Fast Feature Embedding [27]), deep learning framework which has an interface in Python.

CAFFE works with different types of machine learning, designed for solving the problems of classification and image segmentation. The CAFFE provides convolutional neural networks, RCNN, long-term short-term memory, and fully connected neural networks. At the same time, graphic process systems (GPUs), supported by CUDA architectures and using the CuDNN library from Nvidia, are used to accelerate learning. [27]

The main part of the CAFFE's work are blobs - multidimensional data arrays using in parallel computing that fit on a CPU or GPU. Training in CNN processed as parallel multiprocessor computing of blobs from layer to layer.

The recognition process requires the video stream from the camera and detection of the object type to each frame.

The first step for model training is dataset preparation. The images have been founded in Google Images using simple queries for each necessary object. Then the script in JavaScript was used to collect the URLs for the observed images:

```
var script = document.createElement('script');
script.src = "https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js";
document.getElementsByTagName('head')[0].appendChild(script);
var urls = $(''.rg_di .rg_meta').map(function(){return JSON.parse($(this).text()).ou; });
var textToSave = urls.toArray().join('\n');
var hiddenElement = document.createElement('a');
hiddenElement.href = 'data:attachment/text,' + encodeURIComponent(textToSave);
hiddenElement.target = '_blank';
hiddenElement.download = 'urls.txt';
hiddenElement.click();
```

The file with URLs should be processed using the Python script to download images to the prepared folders. The code for this task can be founded in the Appendix.

The second step is preparing the data for the training: to convert the images to the LMDB (Lightning Memory-Mapped Database) format readable for the Caffe module.

The following bash script has been used for this task:

```
EXAMPLE=~/.scripts/examples/dataset
DATA=~/.scripts/data/dataset
TOOLS=build/tools
```



```

TRAIN_DATA_ROOT=~/.scripts/dataset/train/
VAL_DATA_ROOT=~/.scripts/dataset/train/
RESIZE=true
GLOG_logtostderr=1 $TOOLS/convert_imageset \
$EXAMPLE/dataset_train_lmdb
echo "Creating LMDB values..."
GLOG_logtostderr=1 $TOOLS/convert_imageset \
$DATA/text.txt \
$EXAMPLE/dataset_val_lmdb
echo "Ready"

```

Also, it is necessary to compute the mean image. The purpose of the neural network learning process is to search the global minimum of the cost function. To make this process faster, the correct data preparation is needed. One of the methods for data pre-processing is the data normalization. The data normalization can be implemented by subtracting the mean value to get a new dataset with mean = 0.

The bash script that has been used for mean image computation is the following:

```

EXAMPLE=~/.scripts/examples/dataset
DATA=~/.scripts/data/dataset
TOOLS=build/tools
$TOOLS/compute_image_mean $EXAMPLE/dataset_train_lmdb \
$DATA/dataset_mean.binaryproto

```

The structure and parameters for the neural network are described in the prototxt file.

And the third step is the neural network training that can be implemented using one command: `./build/tools/caffe train --solver=models/dataset_alexnet/solver.prototxt`

As the model is prepared, the script for real-time object recognition can be realized. The Python code uses a model and prototxt file. During the code execution, the video stream from the camera connected to the Raspberry Pi is shown inside the frame. The full code is presented in the Appendix.

The screenshots below show the result of the program execution:



Figure 7 – cars observed by the algorithms

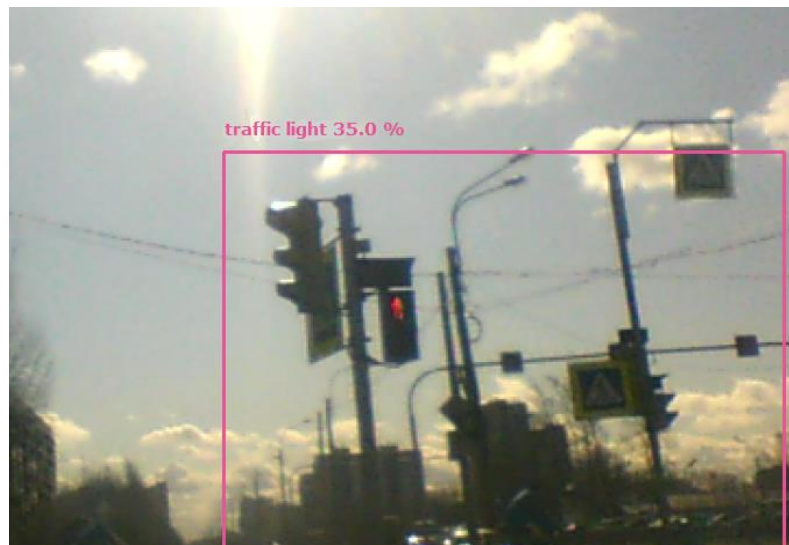


Figure 8 – traffics light is detected



Figure 9 – truck is detected



Figure 10 – person detection has the most level of accuracy (more than 95%)



Figure 11 – sometimes the algorithms shows the wrong results



Figure 12 – the bus is detected correctly

The results of the detecting algorithm will be summarized in the next section.

### 3.1.1 Objects recognition testing analysis

The graph below shows the ratio of losses to accuracy for training and test classifier during the training of the neural network described in the previous section.

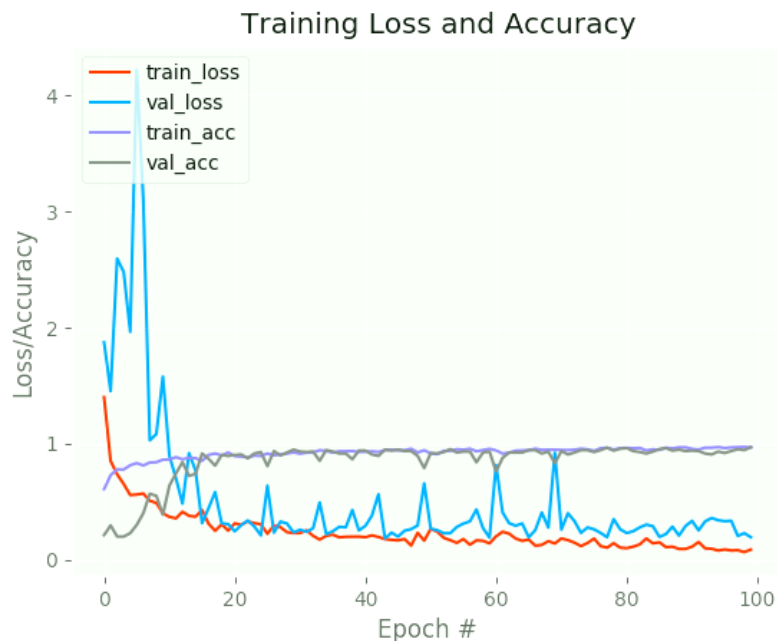


Figure 13 – the plot describing the training loss and accuracy, epochs (corresponding to the time in secs) are along the x axis, the levels (the value 1 equals to 100%) of the training loss and accuracy are shown along the y axis

Training and validation values during the NN learning are shown on the plot above.

The epoch here is the number of cycles, in other words, this value shows how many times NN reads every example to find the pattern on it. The number of epochs for the NN learning is equal to 100.

After the learning of the neural network, the theoretical accuracy, as can be seen from the graph, is 97% for both validation and training sets.

The training of the network was based on the image datasets divided to the training dataset (70%) and validation dataset (30%). Images for datasets were collected in Google Images using a search query that matches the type of object (for example, for the object "person" the request was "person"). The search is conducted according to the relevance of the results, the number of images in the dataset for each type was at least 180.

Before training the network, the following images were discarded:

- not relevant;
- with low resolution (less than 600x800 px);
- damaged;

Network training took about 40 minutes, the result after the first training is shown in the graph above.

The tests in real conditions (outdoor and indoor) for the object recognition algorithm have been implemented. After the tests, the obtained data have been summarized in the table below. The test includes real-time indoor/outdoor observation containing ~30 coincidences for each object type:

Table 1 - analysis of the detection for each object type at the street and at home

Object type	Street	True Positive	False Positive	False Negative	True Negative	Home	True Positive	False Positive	False Negative	True Negative	All
bed	40	33	2	1	4	0	0	0	0	0	40
bench	28	19	2	0	7	0	0	0	0	0	28
bicycle	24	19	1	3	1	0	0	0	0	0	24
bus	38	31	7	0	0	0	0	0	0	0	38
cat	30	25	1	1	3	2	2	0	0	0	32

car	41	40	1	0	0	0	0	0	0	0	41
cell phone	30	12	1	7	2	6	4	3	4	3	36
chair	4	0	4	0	0	33	30	0	1	2	38
cup	0	0	0	0	0	29	18	4	5	2	29
dining table	1	0	1	0	0	29	28	0	1	0	30
dog	14	11	1	1	1	9	6	1	2	0	23
fork	4	0	3	0	0	31	22	6	0	4	35
knife	4	0	2	0	0	22	17	1	5	1	26
microwave	0	0	0	0	0	24	20	2	1	1	24
person	28	27	1	0	0	16	15	1	0	0	44

sink	0	0	0	0	0	22	14	4	1	3	22
sofa	1	0	1	0	0	32	28	1	2	1	33
traffic light	21	16	1	3	1	0	0	0	0	0	21
tv monitor	0	0	0	0	0	30	25	3	0	2	30
Mean	78.319%					80.204%					79.26%

Based on the data from the table, I can say that the algorithm for determining object type recognizing copes with its task in general. On average, it correctly determines the type of object with an accuracy of 79.26%. The average percent of correctly determined objects at the street is 78.319%, at home – 80.204%. The difference might be connected with the number of the observed objects and their size, however, the difference is not large.

The algorithm shows the best results for those objects that have a large size (car, table, sofa) and a large amount of training data (person, chair, bed). The detailed analysis is shown below:

Table 2 – statistical measures of the algorithm performance

Object type	Sensitivity, TPR	Specificity, TNR	PPV	FNR	Accuracy
bed	0.971	0.667	0.943	0.029	0.925
bench	1	0.778	0.905	0	0.929
bicycle	0.864	0.5	0.950	0.136	0.833
bus	1	0	0.816	0	0.816
cat	0.964	0.75	0.964	0.036	0.938
car	1	0	0.976	0	0.976
cell phone	0.593	0.556	0.800	0.407	0.583

chair	0.971	0.5	0.943	0.029	0.921
cup	0.783	0.333	0.818	0.217	0.69
dining table	0.966	0	0.966	0.034	0.933
dog	0.850	0.333	0.895	0.15	0.783
fork	1	0.308	0.710	0	0.743
knife	0.773	0.25	0.850	0.227	0.692
microwave	0.952	0	0.909	0.048	0.87
person	1	0	0.955	0	0.955
sink	0.933	0.429	0.778	0.067	0.773
sofa	0.933	0.333	0.933	0.067	0.879
traffic light	0.842	0.5	0.941	0.158	0.810
tv monitor	1	0.4	0.893	0	0.9
Mean	0.915	0.349	0.892	0.085	0.839

Here, the sensitivity means the proportion of actual positives that are correctly identified as such or true positive rate (TPR) and calculates as:

$$TPR = \frac{TP}{TP + FN}$$

Specificity or true negative rate (TNR) means the proportion of actual negatives that are correctly identified as such and can be calculated as:

$$TNR = \frac{TN}{TN + FP}$$

PPV (positive predictive value) or precision here is proportions of positive and negative results:

$$PPV = \frac{TP}{TP + FP}$$

FNR (false negative rate) shows the probability of falsely detection:

$$FNR = \frac{FN}{FP + TN}$$

And the accuracy shows the closeness of the recognized object to its real type and it's calculate as:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$



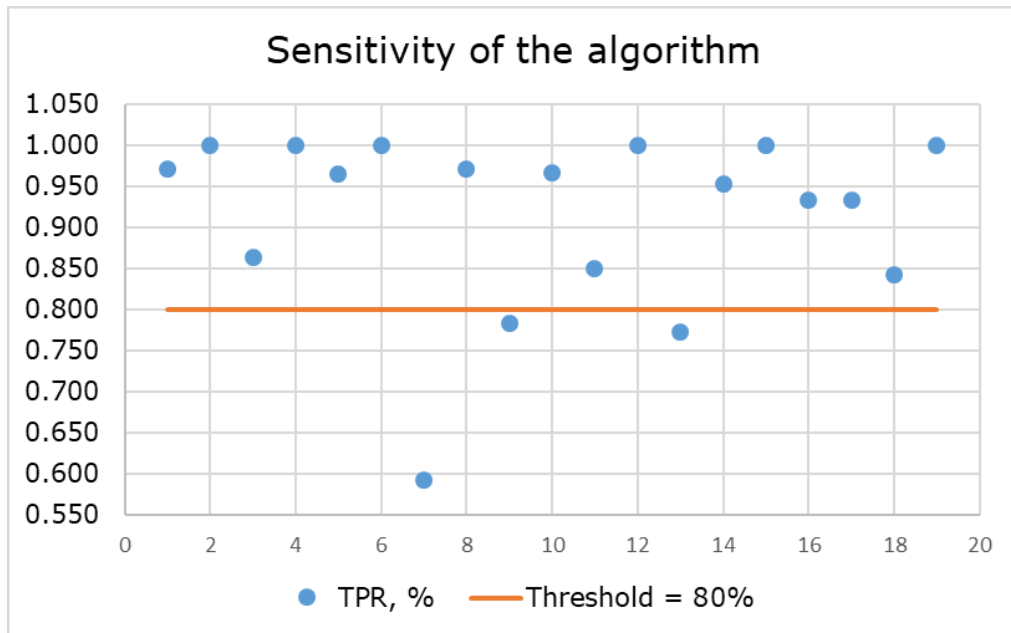


Figure 14 – sensitivity plots demonstrates that most of the determined objects correctly recognized and crossed the threshold 80% value

Mean value of the algorithm sensitivity is 91.5% that is very good result. The plot below shows the specificity of the algorithm:

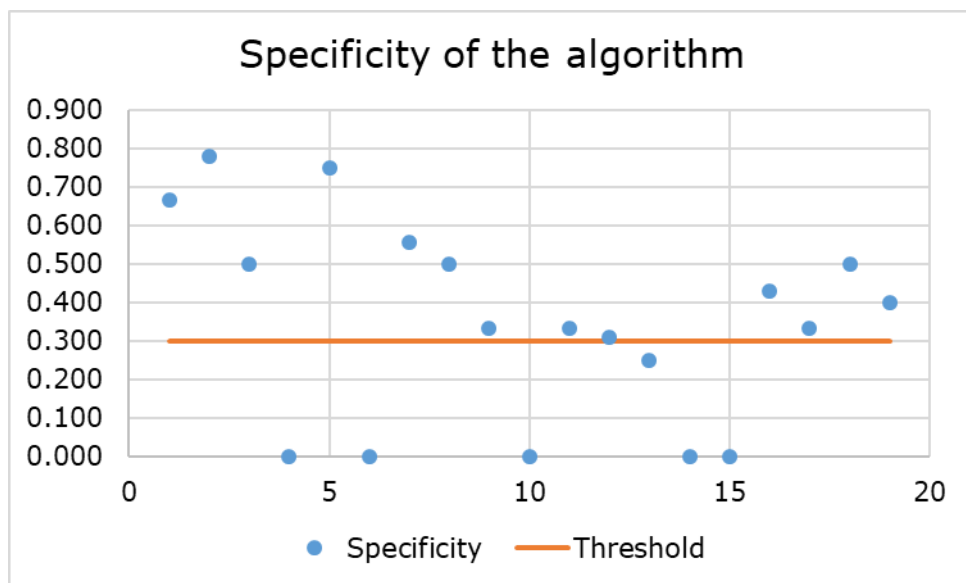


Figure 15 – specificity plots demonstrates the possibility that values have been recognized as negative mostly correctly

Here the threshold is 30%, and most of the values crossed this threshold. According to the plot above, can be concluded that the algorithm should be improved for better definition of the true false values.

However, the mean (83.9 %) and individual accuracy values are quite high that shows the high precision of the algorithm. The accuracy might be increased by improving the algorithm for higher values of the specificity mentioned above.

Comparing theoretical and real accuracy, it's possible to summarize the real accuracy is lower. There's a list of factors causes the accuracy decline:

- training and validation datasets consist of each object separately, i.e. each image has only one object often without bright background. However, if the images in the dataset have excess details, it can decrease the pattern recognition itself;
- in real conditions, the picture may contain several objects of different types and dimensions.

The algorithm has some drawbacks listed above, however, the performance and accuracy is enough for the first prototype. In the future, it is possible to retrain the neural network using more voluminous datasets and varied images inside them.

## 3.2. Distance measurement and mapping using ultrasonic sensor

Determining the distances for objects in rooms is important for people with visual impairments, as this will reduce the risk of injury and allow them to navigate in space easier, especially considering the fact that these people spend much more time indoors than on the street. The following parameters were chosen as the goal for the problem solved in this chapter: find the beginning and the end of the user's path, track the path with an accuracy of 0.5 m, find the correct correspondence to the distance traveled, find the dimensions of the room with an accuracy of 0.5 m, determine the shape of the objects' in the room with an accuracy of 0.5 m.

Ultrasonic sensors emit short, high-frequency sound pulses at regular intervals. These pulses are transmitted in the air at the velocity of sound. If they strike an object, then they are reflected back as echo signals to the sensor. The echo signal can be post-processed to the distance as time-span between emitting the signal and receiving the echo.

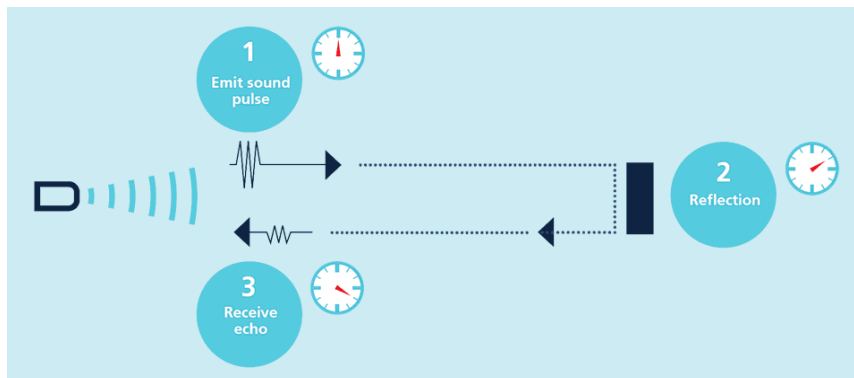


Figure 16 – the description of the ultrasonic sensor working method

An Ultrasonic Distance Sensor (UDS) has 4 pins:

- Vcc is the power pin, requires 5V;
- Trig is transmitting pin (trigger) and it can be triggered with 3.3V;
- Echo is the receiving pin. The output of this pin has 5V;
- Gnd is the ground pin.

Since the Echo pin outputs 5V and the Raspberry can only receive 3.3V maximum, it is necessary to use a potential divider. The resistance depends on the voltages mentioned above:

$$R_2 = \frac{V_{out} * R_1}{V_{inp} - V_{out}}$$

where  $V_{out} = 3.3 \text{ V}$  is the divider output voltage,  $V_{inp} = 5 \text{ V}$  is, respectively, divider input voltage.

If  $R_1$  is assumed as  $1000 \Omega$ , then  $R_2$  will be equal to  $1941, 176 \Omega$ . For more convenience, this value is rounded up:  $R_2 = 2000 \Omega$ .

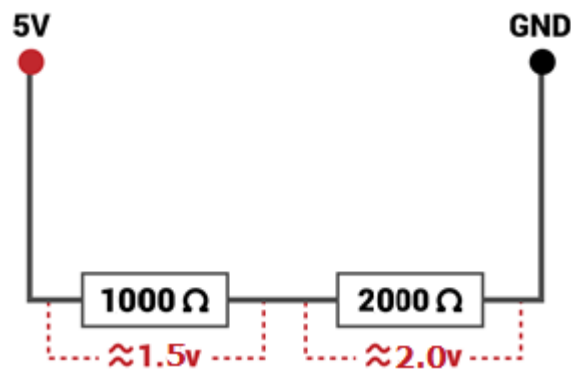


Figure 17 – the voltage divider

Also, the vibration motor was added to provide feedback for the user: the less distance to the object – the more frequent vibrations. During the next stages of developing the device, vibration motor can be replaced by a microphone playing prepared phrases.

Before developing the algorithm, there's a necessity to describe the calculation of the parameters:

1. distance:

$$d = \frac{v * t}{2}$$

where  $v$  – ultrasound speed,  $t$  – time from transmitting signal and receiving it. Since signal passes 2 ways – forward and reflected – the distance is divided by two.

2. vibration intensity:

The vibration motor is controlled by GPIO output using Pulse Width Modulation (PWM). PWM is described in the gpiozero library and has a rate between 0 and 1.

$$v = \frac{(d - \text{minDistance}) \times (\text{minValue} - \text{maxValue})}{(\text{maxDistance} - \text{minDistance})} + \text{maxValue} = \frac{-(d - 0.1)}{(2 - 0.1)} + 1$$

Here minDistance = 10 cm, this is the minimum distance for sensor reaction, maxDistance, accordingly, the maximum distance for reaction, assumed as 2 m. MinValue and MaxValue correspond to the 0 and 1 PWM values.

3. mean ultrasound speed for temperature rate:

$$\sqrt{\frac{\gamma * R * T}{M}}$$

where  $\gamma$  is the adiabatic index of air = 1.4,  $R = 8.3144$  is the universal gas constant,  $T$  is the absolute temperature of the air (K),  $M$  is the molecular mass of air (g / mol) = 28.98. The operating temperature range of the sensor is 30 - 80°C, the speed of sound is calculated for them. The mean speed for the working temperature rate is 343 ms<sup>-1</sup>.

To obtain the distance values, the Python script has been written. The InputDevice and the OutputDevice libraries have been used to control the ultrasonic sensor. Functions in the script use time span for sending and receiving signals according to the sensor's documentation:

- The trigger pin transmits ultrasound for 10  $\mu$ s;
- Then echo pins waits for the reflected signal, its state is active. The pulse time starts recording;
- After the receiving echo, the echo pin becomes inactive, the pulse time interval is obtained;
- Then the sensor "sleeps" for 6 ms.

Then it is necessary to add another 2 sensors and write data to the file. Since the data from the sensors include errors, noise, it is necessary to pre-process them. Firstly, the signal was limited to "bottom" and "top" - negative values and values corresponding to distances greater than 400 cm were discarded. Secondly, the data was processed using

the median filter. The plots below show the data from ultrasonic sensors with and without filtering:

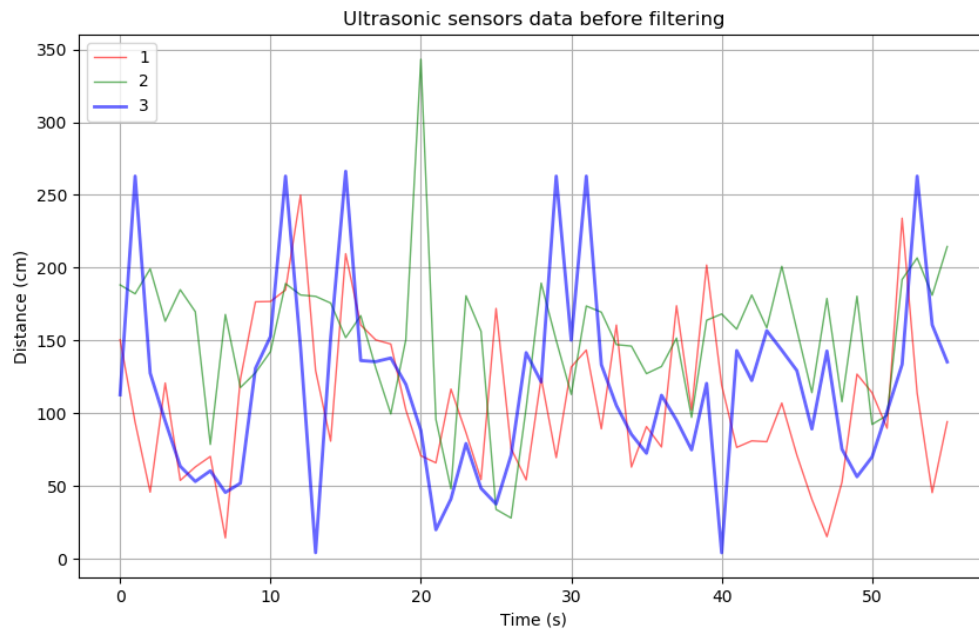


Figure 18 – data from ultrasonic sensors before filtering

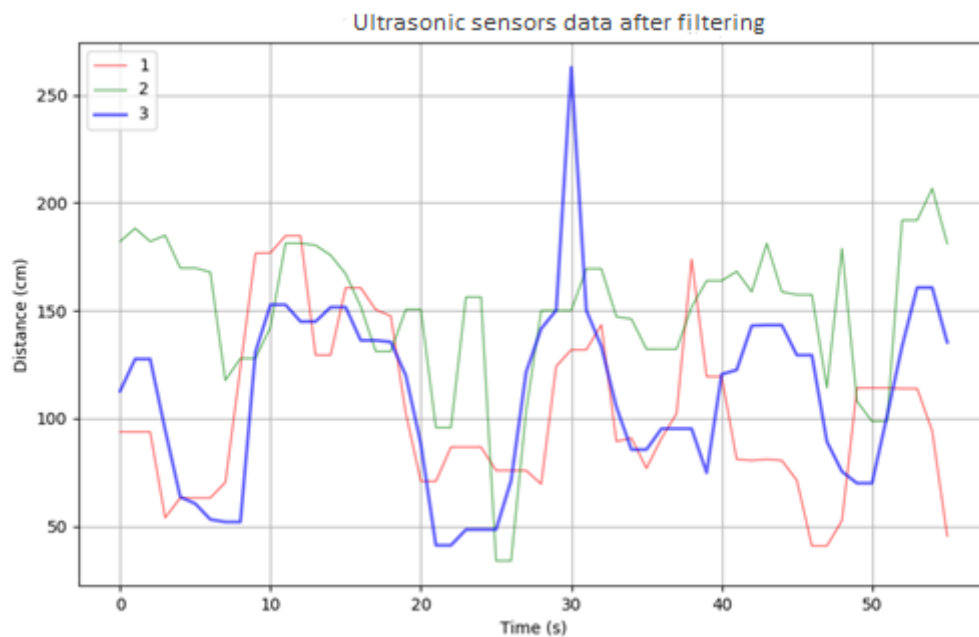


Figure 19 – data from ultrasonic sensors after median filtering with kernel = 3

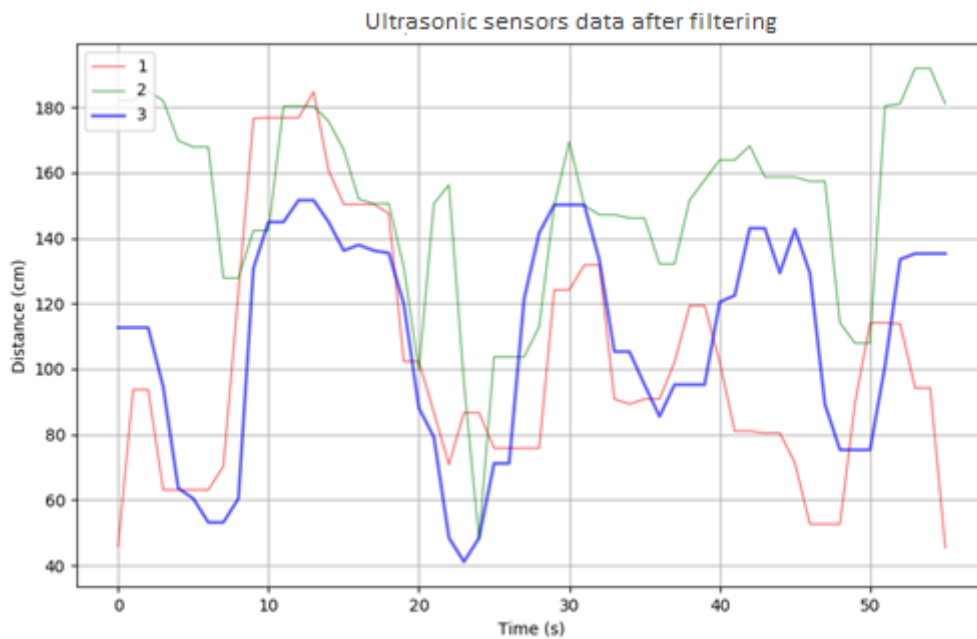


Figure 20 – data from ultrasonic sensors after median filtering with kernel = 5

As can be seen from the plots above, the median filter allows achieving more smooth lines with less number of peaks. The kernel size increasing leads to the more smooth curve.

Also, ultrasonic sensors are located at an angle of 40 degrees, to cover a larger area of space and non-intersection of signals.

The full code in Python can be founded in the Appendix. The photo of the prototype for algorithm testing is described below:

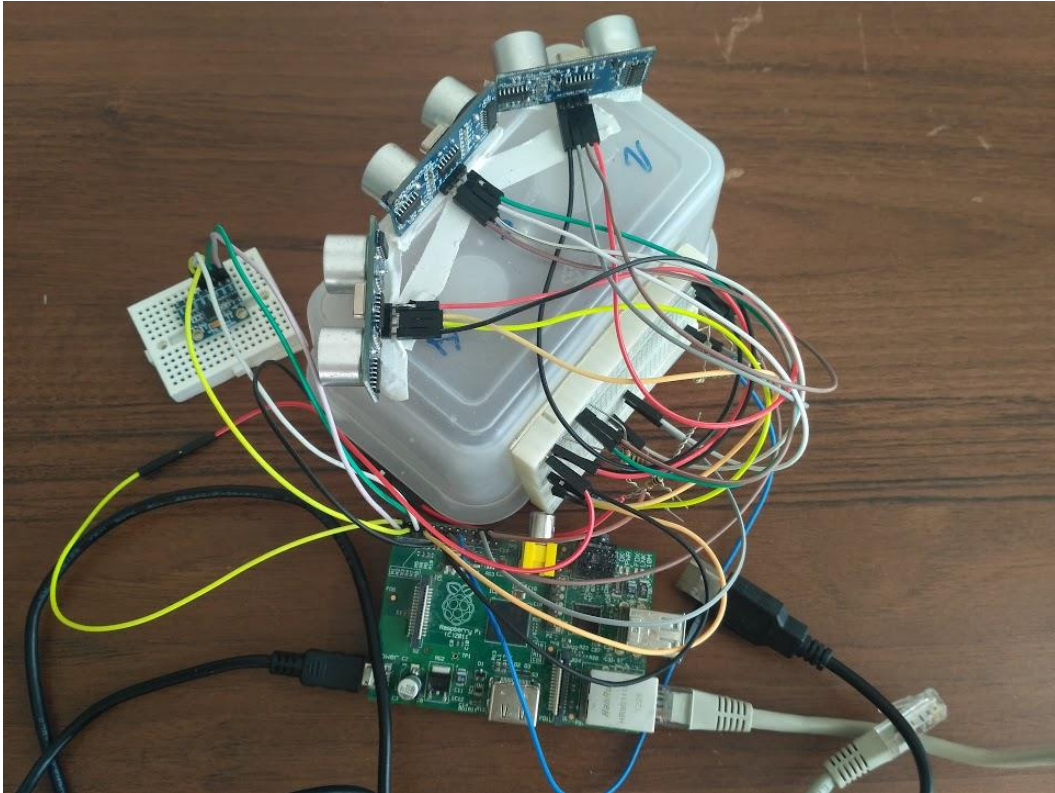


Figure 21 – the prototype used for testing

The important part for mapping is tracking the user's path, due to the necessity to set obstacles according to the user's position. This feature has been realized using the gyroscope/accelerometer sensor GY-521.

To create a map, the following steps are needed:

- read the accelerations from the gyroscope/accelerometer sensor;
- read distances to the obstacles using ultrasonic sensors, process this data;
- write the data both from the gyroscope/accelerometer and ultrasonic sensors to the file;
- read the file and pre-process the angles and acceleration using the Butterworth filter;
- integrate them twice to obtain the coordinates;
- plot the data to obtain a path and a map.



The GY-521 should be connected to the Raspberry Pi board over the I2C interface. To work with the sensor, it is necessary to connect 4 GY-521 pins (VCC, GNS, SCL, SDA) to the Raspberry GPIO pins as it described in the picture above:

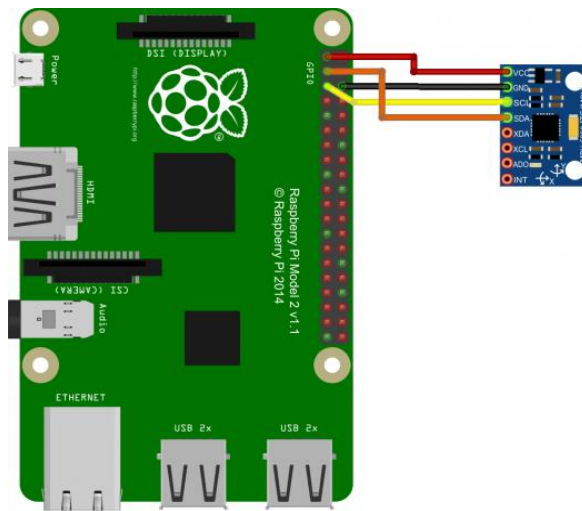


Figure 22 – the gyro accelerometer connection

It is necessary to configure the I2C communication protocol in Raspberry Pi and find the bus address of the connected sensor using the following command in the terminal: `sudo i2cdetect -y 1`.

```
pi@raspberrypi:~$ sudo i2cdetect -y 0
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  68  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
```

Figure 23 – the detection of the register of the GY521 connection

Now the algorithm can be written using this address. The full code for obtaining the angles and accelerations data can be founded in the Appendix.

The next step is obstacle detection. After obtaining data from all sensors it is necessary to write them into file for further processing.

The code for writing the data for sensors into the file is places below:

```
def write_csv(data):
```

with open('/home/pi/Desktop/data2.csv', 'a') as outfile:

```
writer = csv.writer(outfile, delimiter=',')
```

```
writer.writerow(data)
```

```
x = datetime.datetime.now()
```

```
data = [x, acceleration_xout_skaled, acceleration_yout_skaled,  
acceleration_zout_skaled, distance1, distance2, distance3]
```

```
write_csv(data)
```

Then it's necessary to read accelerations from the file and process data using Butterworth filter to remove noise. The code below describes Butterworth filtering:

```
N = 2 # Filter order
```

```
Wn = 0.001 # Cutoff frequency  $0 < Wn < 1$ 
```

```
B, A = signal.butter(N, Wn, output = 'ba')
```

```
xAcc[:] = signal.filtfilt(B, A, xAcc[:])
```

```
yAcc[:] = signal.filtfilt(B, A, yAcc[:])
```

To obtain coordinates, it's necessary to integrate the accelerations twice using `scipy.integrate.cumtrapz` function. This method not accurate, due to the quality of accelerometer data and errors. However, it should be enough to obtain not accurate path to make a general map.

The last step is, respectively, plotting the data using the processed data from the sensors. The plot below shows the obtained map of the room:

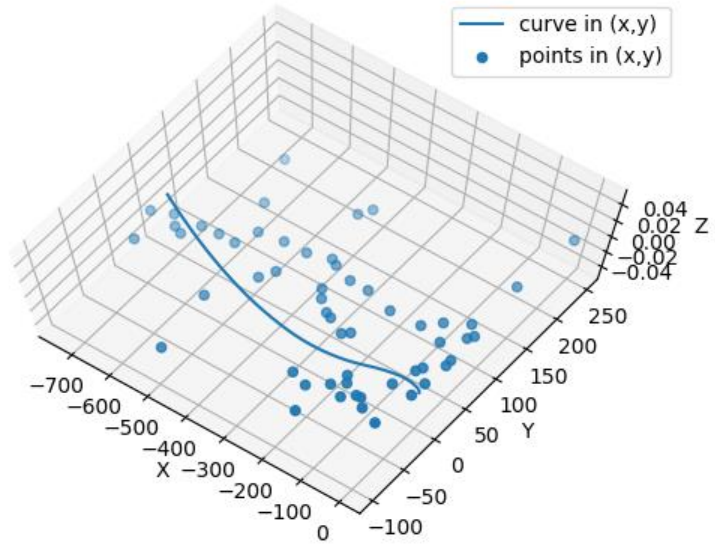


Figure 24 – the path and the room outlines

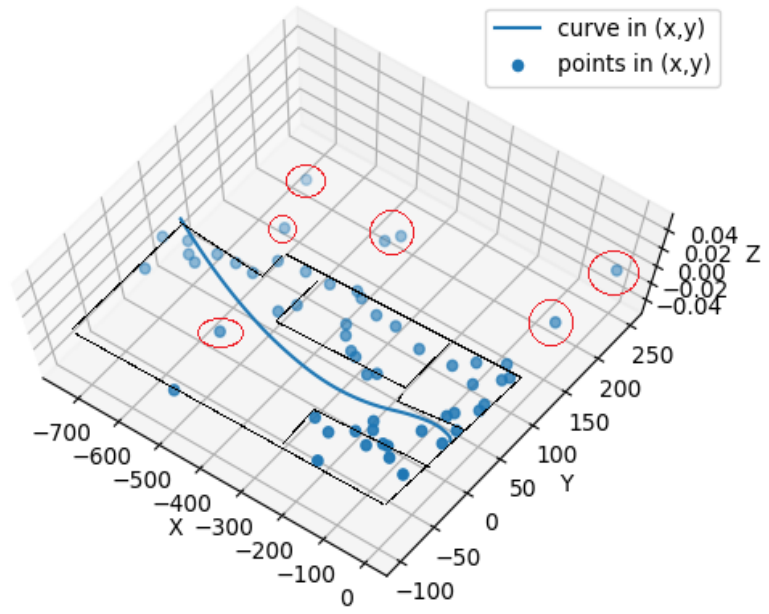


Figure 25 – the outlines of the room and objects in the room are highlighted, the noise values are placed in circles

The constructed map fairly accurately shows the size and shape of the room, although the result is not ideal. It can be noted that the left wall of the room is practically not marked, this was due to the small amount of data (the map was built after one circle). It is not difficult to improve the map by collecting more data.

### 3.2.1. Distance measurement and mapping analysis

The following tasks were set as a goal for the room map:

- Track the path of a person with an accuracy of 0.5 m (turns, beginning, and end of the path, correspondence to the distance travelled);
- Determine the dimensions of the room with an accuracy of 0.5 m;
- Determine the shape of large objects in the room with an accuracy of 0.5 m.

According to the received map of the room described above, it's possible to say that the tasks were completed. The real room dimensions were 3x7 m, the map shows dimensions 2.3x7 m, this is the satisfied result.

As for tracking the path, the tasks set in this direction were also fulfilled: the beginning and end of the path correspond to the actual path, the distance traveled also corresponds to the real one with an accuracy of less than 0.5 m.

The plots below show the accelerations, velocities, and the path obtained by accelerometer:

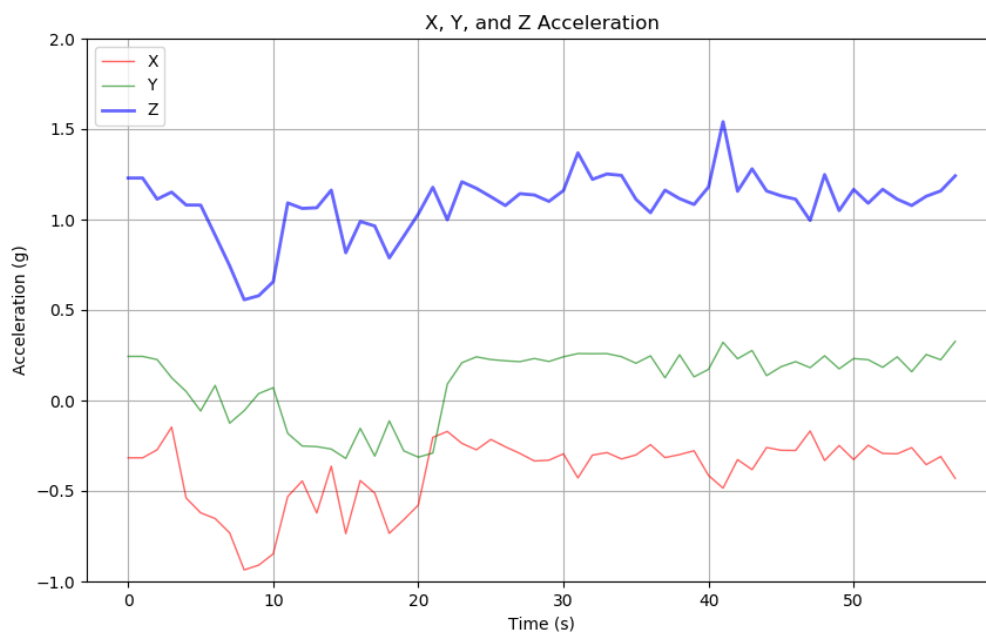


Figure 26 – the accelerations obtained by accelerometer

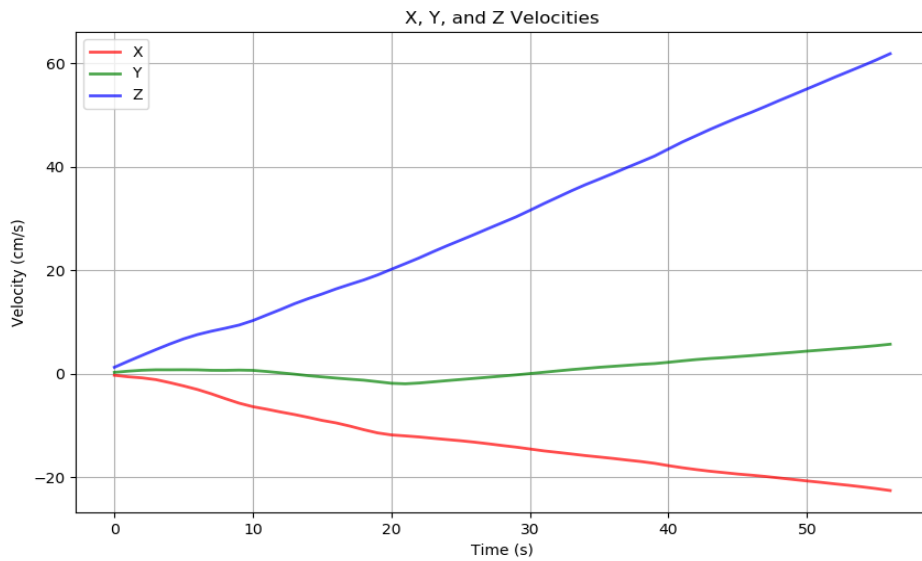


Figure 27 – the velocities calculated from accelerations

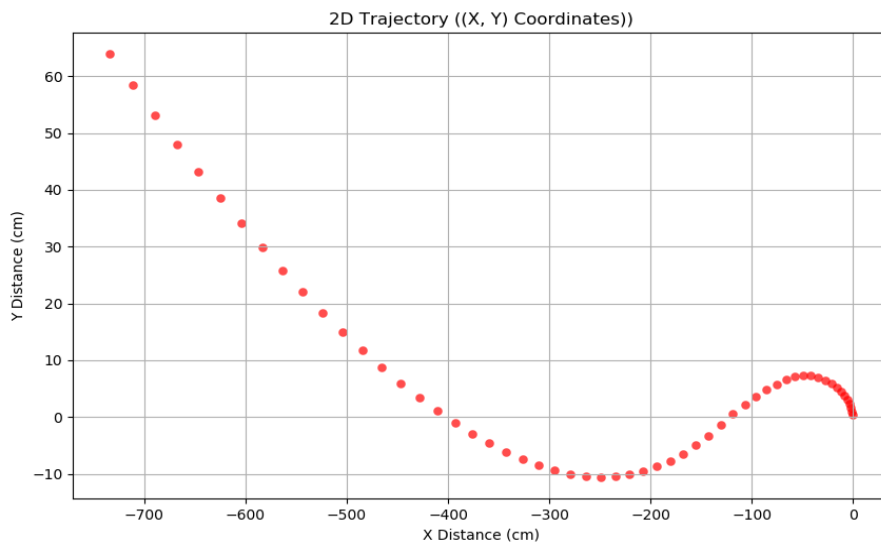


Figure 28 – the position in 2D calculated from the velocities

As a conclusion for mapping and distance measurement, it's possible to say that all tasks were completed. However, the result can be improved by using more data. As mentioned above, this result was obtained after one walk around the room, with an increase in the aisles it's possible to obtain a more detailed map.

### 3.3. Description of the device operation algorithm

The operation algorithm should combine several scripts mentioned above and provide the work of the whole device.

Operation algorithm should solve the following tasks:

- run the script for the device after turning on;
- control the sensors;
- run the script for map designing;
- inform the user about obstacles;
- inform the user about the battery discharging.

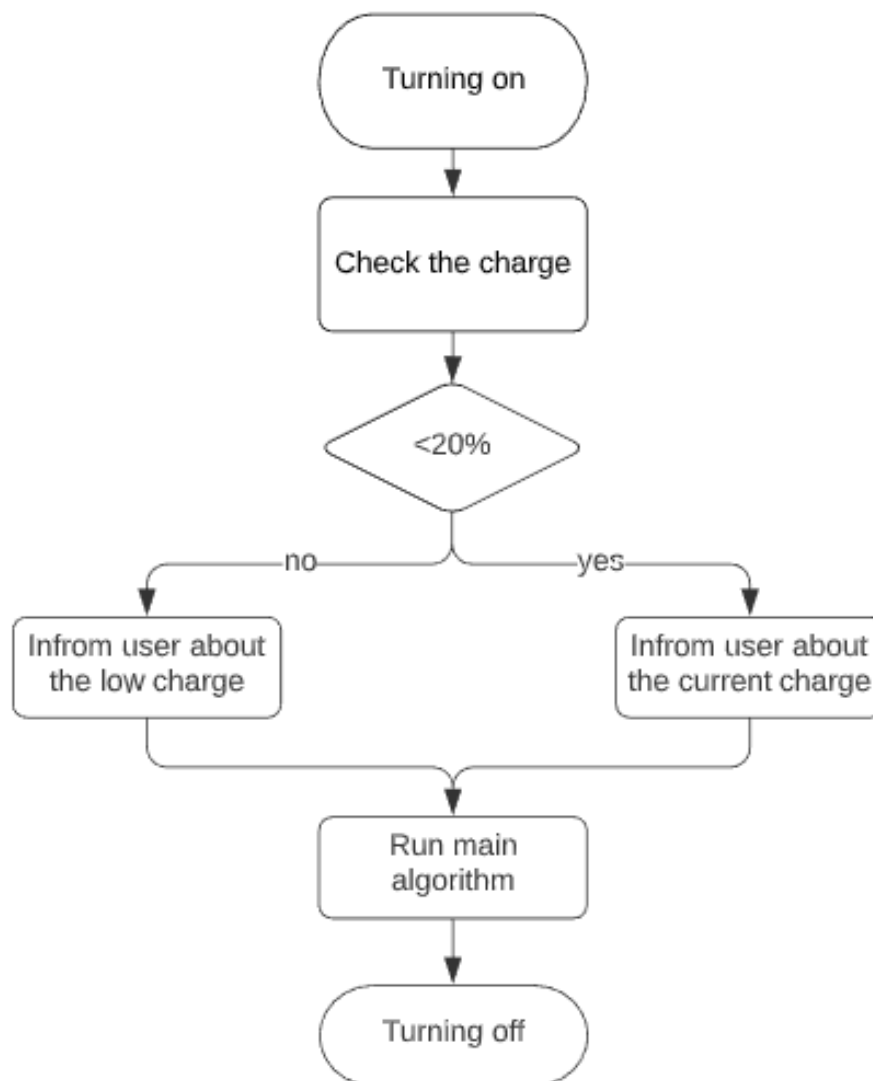


Figure 29 – the operation algorithm description

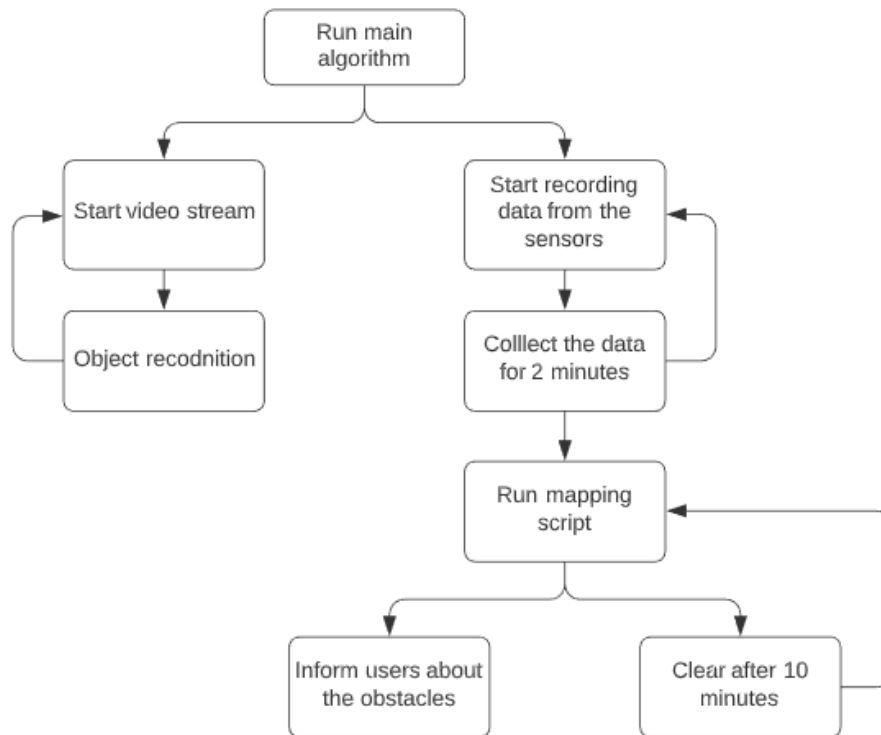


Figure 30 – the main script description

To run scripts after boot it is needed to modify the `.bashrc` file in the `/home/pi/` directory. In this file 2 lines should be added in the end of the file, where `starting.py` is the file with the main operation algorithm:

```

echo Running at boot
sudo python /home/pi/starting.py

```

The `starting.py` code is the following:

```

#!/usr/bin/python
import subprocess
import time
code1 = 'python sensors.py' #code for record data from sensors
code2 = 'python object_detection.py' # code for object detection
code3 = 'python map.py' #code for mapping
data_file = open("/home/pi/Desktop/data.csv", "rw+")
def mapping_buffer(seconds): #run mapping for 10 minutes
start = time.time()
    time.clock()
    elapsed = 0

```

```

while elapsed < seconds:
    elapsed = time.time() - start
    script_3 = subprocess.Popen(code3, stdout=subprocess.PIPE,
shell=True)
def main():
    while True:
script_1 = subprocess.Popen(code1, stdout=subprocess.PIPE, shell=True) #run first
script
script_2 = subprocess.Popen(code2, stdout=subprocess.PIPE, shell=True) #run second
script
time.sleep(60) #wait for data collection
mapping_buffer(600) #run mapping for 10 minutes
    data_file.truncate() #clear file with data to restart the map

```

The code above has been successfully executed, the algorithm runs all necessary scripts.



### **3.3.1. Operation algorithm testing and analysis**

For the working algorithm, the tasks described in the previous section were set. In general, the working algorithm should start after turning on the device and run all other scripts necessary for work.

The most important indicator of a working algorithm is its ability to run in real conditions. The parameter that should be also taken into account is the algorithm performance.

Regarding the ability to work in real conditions, it is possible to say that this purpose has been achieved. The algorithm has been tested indoors (for example, the mapping section describes indoor testing) and outdoors (the section about the object detection shows the results).

However, it should be mentioned that the current scripts have some issues. The object detection part is the most "loading" part since this script uses functions and libraries that are most demanding on the CPU usage. The execution of the main code causes 98-99% CPU utilization. This fact causes the limitations and warnings:

1. There's no possibility to add other features;
2. The high constant load of the CPU causes the board overheating, and subsequently can damage it.

To overcome the issue with high CPU usage, it is possible to split up the solving task between separate boards. For instance, connect an external small-scale Arduino Nano or Raspberry Pi Zero board and execute part of the scripts there (for example, the recording data from the sensors).

Additionally, to prevent overheating, the Raspberry Pi board should be placed in a special case providing ventilation and cooling. The resolution of this task is described in the modelling section.

### 3.4. Electrical scheme developing

The electrical scheme should ensure the correct connection of the selected electronic components. Also, it should describe the power connection.

The requirements set for power source are the following:

- It should provide the necessary voltage level for the components;
- The power source must ensure the operation of the device for at least three hours.

So, the first step before the scheme developing is the calculation of current and voltage consumed by electronic components, and the selection of a battery based on it.

The total power consumption can be calculated as a sum of average consumable power and current:

Table 3 – power characteristics of the module components:

Component	Average consumable power	Average consumable current
Raspberry Pi Model B+	3.5 W	700 mA
Ultrasonic module HC-SR04	0.75 W	15 mA
Vibration motor 1030	0.1914 W	58 mA
Audio module HPM14A	0.005 W	1 mA
USB camera	2 W	500 mA
Gyro accelerometer GY-521	0.013 W	3.9 mA

Average power consumption and current, respectively:

$$W_{\Sigma av} = 3.5 + 0.75 * 3 + 0.1914 + 0.005 + 2 + 0.013 = 7.9594 W$$

$$I_{\Sigma av} = 700 + 15 * 3 + 58 + 1 + 500 + 3.9 = 1307.9 mA$$

To power, the device, lithium-polymer batteries with a built-in protection board and an MCP73833 charge controller with an IM27313 DC-DC converter were selected. Output voltage is 5V (suitable to power the Raspberry Pi board), capacity - 2000 mA/h.

Based on the above data, the device's battery life [28]:

$$T = \frac{U_{acc} * C_{acc} * K * K_{dd} * K_{ac}}{W_{\Sigma av}} = \frac{5 * 2 * 0.8 * 0.85 * 0.8}{7.9594} = 0.683 \text{ (hours)}$$

where  $C_{acc}$  is the capacity of the battery (Ah),  $U_{acc}$  is the supply voltage of the battery (V),  $W_{\Sigma av}$  is the average power consumption (W),  $K$  is the inverter efficiency equal to 80%,  $K_{dd} = 0.85$  is the coefficient depth of discharge,  $K_{ac} = 0.8$  - coefficient of available capacity.

Thus, one battery is not enough to ensure the operation time of the device for 3 hours, it is necessary to use 5 accumulators connected using the parallel method. Then the total capacity of the batteries  $C_{acc} = 10000$  mA/h, the supply voltage of the batteries remains the same. In this case:

$$T = \frac{U_{acc} * C_{acc} * K * K_{dd} * K_{ac}}{W_{\Sigma av}} = \frac{5 * 10 * 0.8 * 0.85 * 0.8}{7.9594} = 3.417 \text{ (hours)}$$

So, it is necessary to use 5 batteries connected in parallel to provide the operation for 3 hours.

The electrical scheme damage in KiCAD software. Some parts of the scheme are described below:

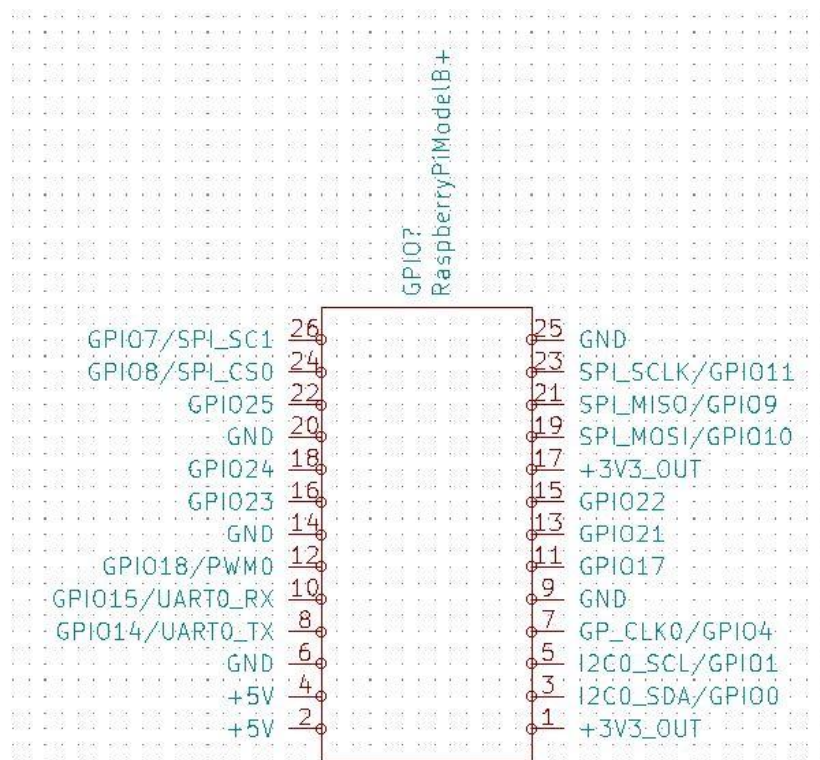


Figure 31 – The Raspberry Pi connecting ports

The Raspberry Pi ports are taken from the board datasheet. As shown in the picture above, Raspberry has outputs for powering the sensors. The sensors can be powered either from Raspberry or common accumulators.

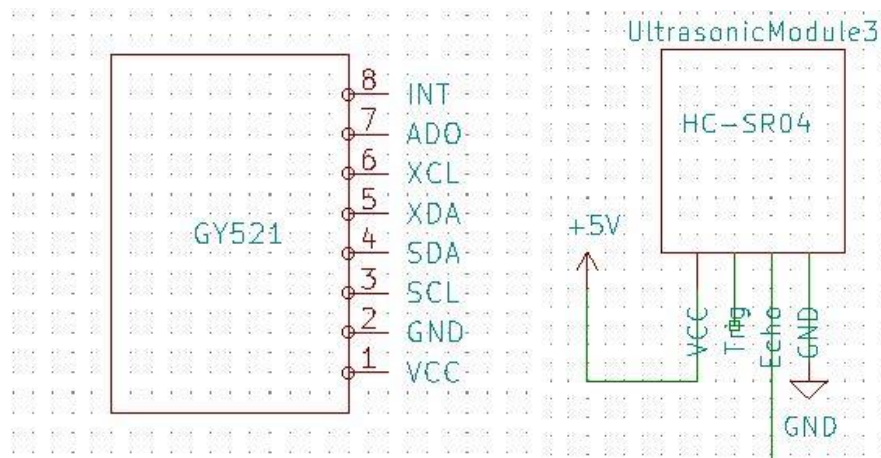


Figure 32 – The GY-521 gyro accelerometer’s and HCSR04 ultrasonic sensor’s ports

The gyro accelerometer has been connected to the 3.3 V power source and I2C ports of the Raspberry Pi board. The connection can be seen in the picture below:

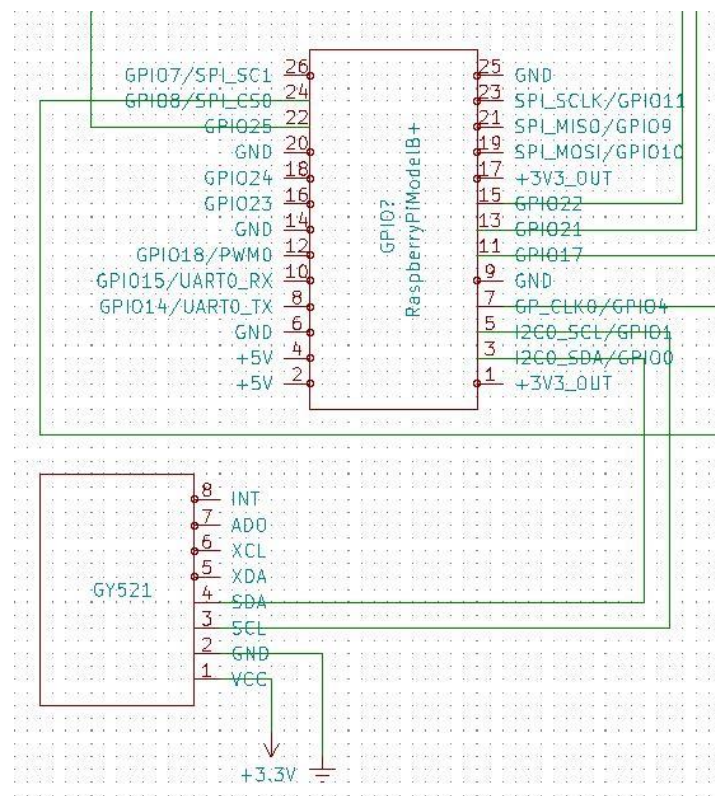


Figure 33 – the connection between Raspberry Pi and GY521

The ultrasonic sensors are connected to the GPIO pins, and as the echo pins output the 5V signal, which can cause the Raspberry Pi damage), it should be connected through the voltage divider:

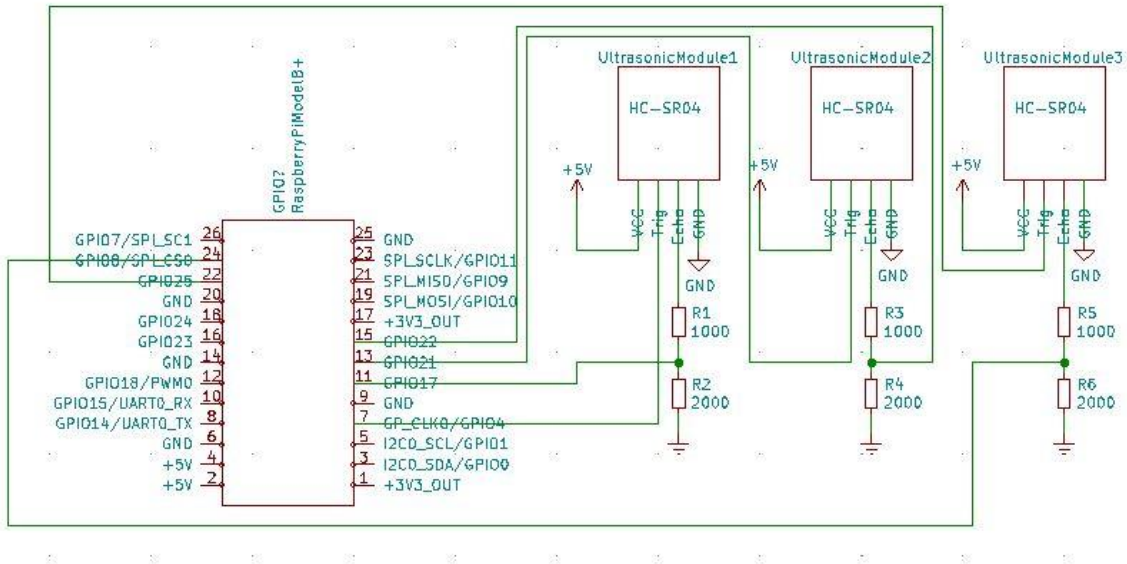


Figure 34 – the connection between Raspberry Pi and HCSR04 sensors

The full scheme can be founded in the Appendix.

### 3.4.1. Electric scheme testing results

The electronic components connected according to the scheme described above fulfill their task. However, as it was noticed in the previous sections the electronic part requires cooling and moisture protection, which will be taken into account in the next section.

The operating time was also checked, the prototype was connected to 5 batteries and left to work, every 5 minutes the battery charge was taken. The graph of the decrease in charge can be seen below:

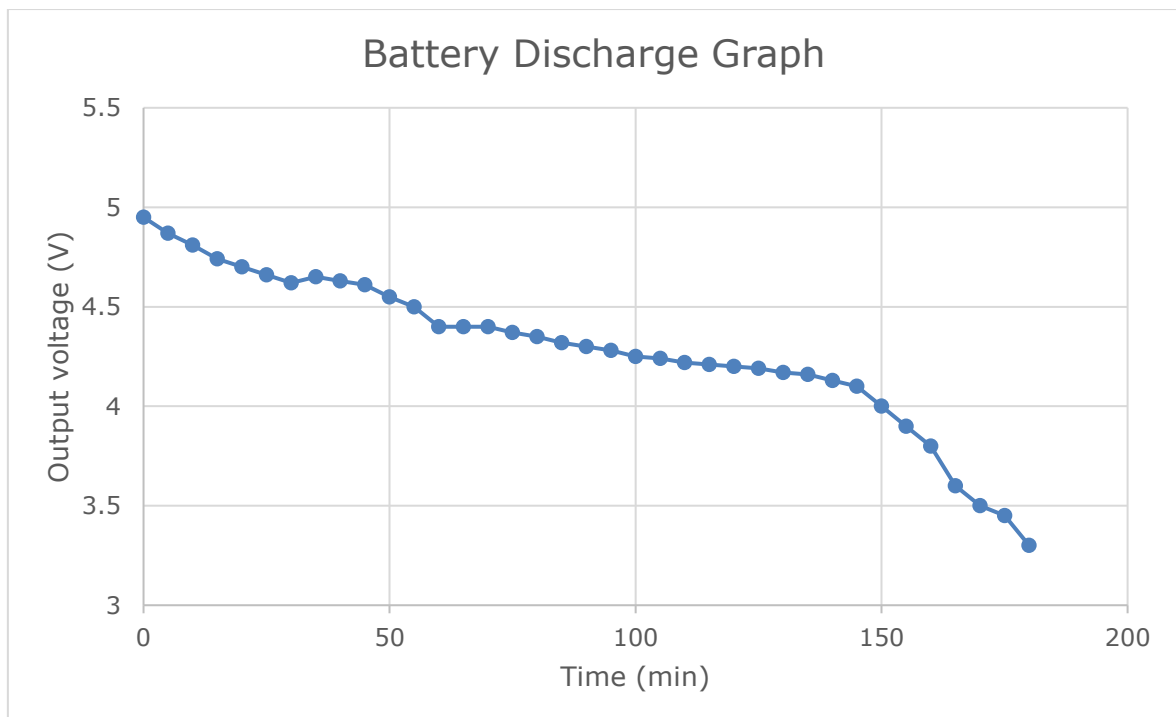


Figure 35 – plot describing the accumulators discharging

As can be seen from this graph, the device can work for 3 hours. However, the theoretical operation time was equal to 3.4 hours, but in practice, the operation time span is 3.1-3.2 hours.

The operation time can be increased by using more number of accumulators or by using the accumulators with more capacity.

### 3.5. 2D and 3D modelling

This section is intended for 2D and 3D modelling of the developing device. As a result of this section the following tasks have been set:

- Developing of the bodies for the Raspberry Pi and ultrasonic sensors;
- The vest should include mountings;
- The placing of the electronic components should be realized correctly, some components should be placed on the vest and some of them inside the vest layers;
- Ventilation and cable management should be taken into account.

Firstly, the body for the Raspberry Pi was developed. The body has holes for the cables and cooling the board. This body should be placed inside the vest. The 3D model can be seen below:

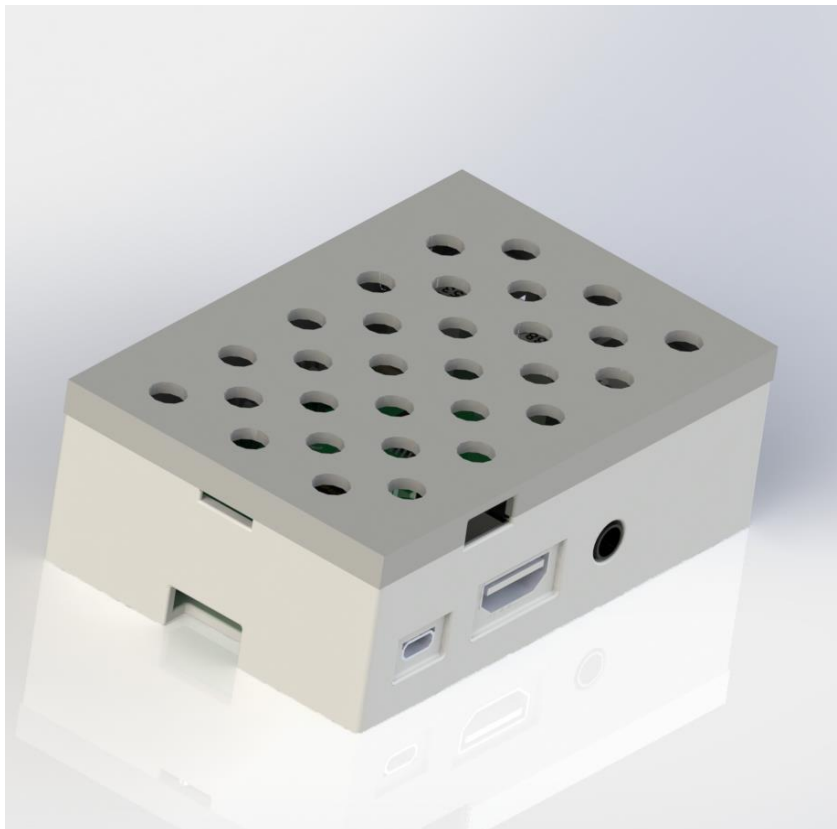


Figure 36 – the body for the Raspberry Pi

The drawing for the body was made in accordance with the requirements of GOST, which is required by the terms of the grant "UMNIK", under which this work is carried out. The drawing can be viewed in the Appendix.

Secondly, the mountings and the body were modelled for ultrasonic sensors. These sensors should be placed at an angle of 40 degrees. Sensors should be located on the vest. The 3D model is as follows:

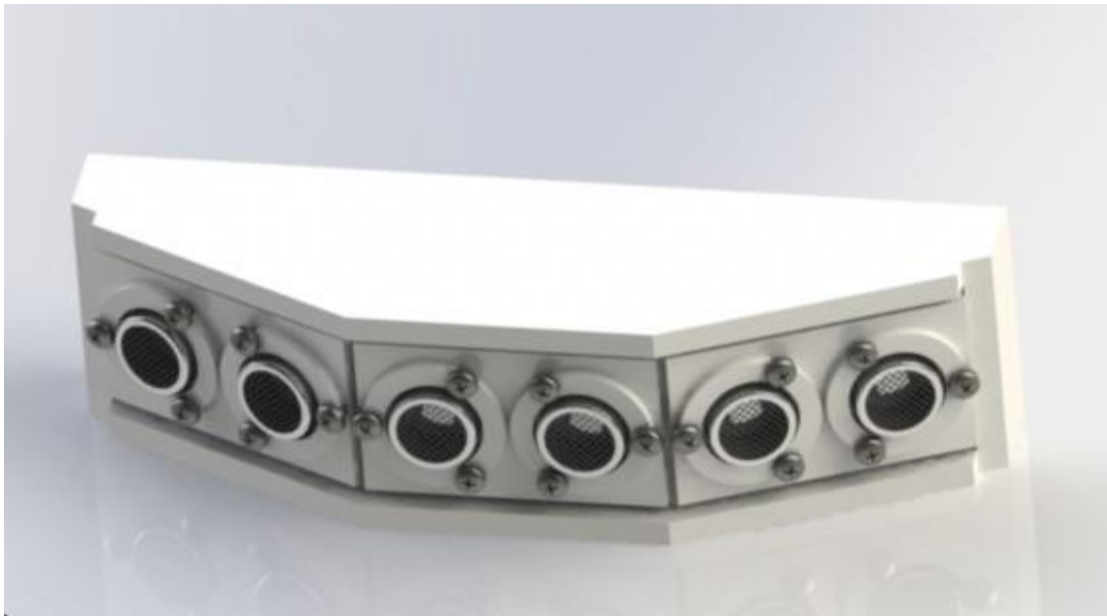


Figure 37 – ultrasonic sensors in the body

The drawing can be found in the Appendix.



The last part is the assembling of the components. The assembling looks as follows:



Figure 38 – the prototype of the device

This vest has mounts on the sides so that it is possible to place the camera and sensors on the front in the center. The camera and ultrasonic sensors are located in the front.

The holes under the camera has been implemented for better Raspberry Pi board cooling. The accumulators are placed in the back, the audio modules and vibration motors are located near the shoulders for better receptivity.

The wires can be easily integrated into the internal layers of the vest. The bodies can be printed using a standard 3D printer.

Also, the weight of the vest is important, as it was defined in the Proposing solution section it should be no more than 3 kg. The vest used as a base for the assembling is

made from neoprene and has weight 2 kg. With all electronics, the weight of the vest shown above is about 2.5-2.6 kg, that less than a set maximum.

### **3.5.1. 2D and 3D modelling results analysis**

To model the device, several important tasks were posed: the vest should have convenient fastenings that any user can handle; vest weight should be less than 3 kg; all electronic components must be placed so that their correct performance is ensured, and at the same time they should be protected from moisture and the heat excessing must be conducted.

All of the above factors were taken into account in the simulated prototype shown in the previous section.

Also, as noted above, the body elements can be printed on a 3D printer to simplify the manufacture of the prototype or made by casting for a complete device.

The basis, type, material, and fastening of lifejackets were used. Such vests are usually made of lightweight materials (neoprene), have convenient fastenings, and do not constrain movement. For the prototype, you can use a life jacket, converted for the required task.

Thereby, it's possible to say that all the tasks for prototype modeling were completed.

### **3.6. Device developing summary**

This chapter was devoted to the development and testing of a prototype device. For development, electronic components and programs were selected.

As was analyzed in the previous chapter, the most appropriate base form for this type of device is the waterproof vest with electronic components inside/on it. As a base, the type and material of the life vest was used, which has convenient fastenings, lightweight and is comfortable to wear. The developed model shows that taking into account all electronic components, the weight of the vest is less than 3 kg.

Also, the device must be able to work for at least three hours. For this, a circuit for connecting electronic components was developed and on their basis, the theoretical operating time and the number of batteries capable of providing it were calculated. According to theoretical calculations, 5 batteries should ensure the operation of all electronics for the planned time, which was confirmed experimentally.

Also in the chapter, algorithms were developed and tested for recognizing a number of objects in real-time and obtaining a room map by determining the distance to objects using ultrasonic sensors and tracking the user's path using an accelerometer. Object recognition is performed in real-time using a camera connected to the Raspberry Pi, the algorithm uses data from a pre-trained neural network.

3 ultrasonic sensors located at an angle of  $40^\circ$  determine obstacles in front, to the right and to the left of the user and cover a distance of 400 mm in front of him. At the same time, the accelerometer tracks movement and turns when walking. Then these data are processed and based on them, a room map is constructed with the outlines of objects inside the room. The size of the room is determined with an accuracy of less than 0.5 m.

Thus, it is possible to conclude that all the tasks were completed.

## 4. DISCUSSION

This chapter is intended for discussion about project results. The limitations that the author encountered while performing the work will be considered. Further steps that can improve current results and possible new features will also be described.

### 4.1. Limitations

There were few limitations encountered during the implementation of the project. They are listed below:

- The CPU of the Raspberry Pi. As noted earlier, the Raspberry Pi CPU is 98-99% loaded, so you need to use an additional controller to separate tasks between them and reduce the load on one board;
- Operation time. Operating time is very limited, and with its increase, the number of batteries should increase also, which affects the weight of the device and ease of use;
- Real-time work. If the object type recognition algorithm is capable of working in real-time, then the map building algorithm needs separate scripts for reading data from sensors and a separate algorithm for building a map since data from sensors must be pre-processed;
- Object type. The type of recognizable objects is limited by the parameters of a trained neural network. In order to expand the list of objects, the neural network must be retrained;
- Object recognition. The quality of object recognition depends on the quality and quantity of data used to train the neural network. As can be seen from the results obtained during the testing of the algorithm, some types of objects have a high level of recognition errors (> 20%). This error level can be reduced by increasing the data in the dataset for the required type of object;
- Path tracking. The accuracy level of tracking the user's path is limited by the accuracy of one sensor, the data from which include noise and measurement errors.

## 4.2. Future improvements

As noted above, the developed prototype has several limitations. Some of them may be allowed in the future. There are also some functions that were not implemented in this work, but might be added in the future.

- Separation of tasks between multiple controllers;
- Adding the ability to notify the user both using the speaker and using headphones;
- Increasing the number of accelerometers for more accurate data on the movement of the user;
- Ability to build maps outside the premises, save them and export to tactile screens;
- Ability to use ready-made databases containing environmental information (for example, the user can use Google Maps and information from them to notify the user about objects nearby).

## **SUMMARY**

This work was devoted to the development of a device that facilitates the qualitative movement of people with visual impairments.

In the first chapter, an analysis of existing solutions was carried out, the advantages and disadvantages of modern developments were noted. The goal of the study and its objectives. The concept of the device is determined and the relevance of development at the present time is confirmed.

Based on a certain concept, components were selected for the implementation of the module. An algorithm for recognizing given types of objects has been developed. An algorithm has been developed that reads indicators from ultrasonic sensors and an accelerometer, as well as an algorithm for constructing a room map.

According to certain criteria, the layout of the device is selected. The operation time for the device is about 3 hours.

Summing up the work, it's possible to say that it fully complied with the requirements of the technical specifications.

## LIST OF REFERENCES

- [1] Sachkov M., Yusupova A., "Filtration algorithms of ultrasound sensor signals for obstacle detection devices," *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, vol. 19, no. 2, pp. 326-332, 2019.
- [2] Fricke, TR, Tahhan N, Resnikoff S, Papas E, Burnett A, Suit MH, Naduvilath T, Naidoo K, "Global Prevalence of Presbyopia and Vision Impairment from Uncorrected Presbyopia: Systematic Review, Meta-analysis, and Modelling," *Ophthalmology*, 9 May 2018.
- [3] Bourne RRA, Flaxman SR, Braithwaite T, Cicinelli MV, Das A, Jonas JB, et al., "Magnitude, temporal trends, and projections of the global prevalence of blindness and distance and near vision impairment: a systematic review and meta-analysis," *Lancet Glob Health*, 5 September 2017.
- [4] E. K. Elmannai W., "Sensor-Based Assistive Devices for Visually-Impaired. Review," *Sensors*, 10 March 2017.
- [5] L. J. Z. D. W. L. W. X. Pei Xuming, "IEEE International Conference on Mechatronics and Automation," in *A Robot Ultrasonic Mapping Method based on the Gray System Theory*, 2010.
- [6] O. G. U. G. Gambino F., "A comparison of three uncertainty calculus techniques for ultrasonic map building," *Proceedings of SPIE - The International Society for Optical Engineering*, January 2015.
- [7] F. H. Tunai P. Marques, "IEEE Green Energy and Smart Systems Conference (IGESSC)," in *Autonomous robot for mapping using ultrasonic sensors*, 2017.
- [8] I. Andersone, "Probabilistic Mapping with Ultrasonic Distance Sensors," *Procedia Computer Science*, no. 104, p. 362 – 368, 2017.
- [9] Y. T. Shuihua Wang, "IEEE International Conference on Bioinformatics and Biomedicine Workshops (BIBMW)," in *Detecting stairs and pedestrian crosswalks for the blind by RGBD camera*, 2012.
- [10] T.-L. L. N. V. Huy-Hieu Pham, "Real-Time Obstacle Detection System in Indoor Environment for the Visually Impaired Using Microsoft Kinect Sensor," *Journal of Sensors*, pp. 1-13, November 2016.
- [11] L.-N. G. G. J. Perez-Yus A., "European Conference on Computer Vision," in *Detection and Modelling of Staircases Using a Wearable Depth Sensor*, 2015.
- [12] "Obstacle Sensing Stick for Visually Impaired Persons," 26 October 2015. [Online]. Available: [http://digital-wizard.net/pic\\_projects/smart\\_stick\\_for\\_visually\\_impaired](http://digital-wizard.net/pic_projects/smart_stick_for_visually_impaired).

- [13] Saaid, M.F. et. al., "Smart cane with range notification for blind people," *IEEE International Conference on Automation Control and Intelligent Systems (I2CACIS)*, pp. 225-229, 2016.
- [14] Manikanta, K., et. al., "Implementation and Design of Smart Blind Stick for Obstacle Detection and Navigation System," *International Journal of Engineering science and Computing*, August 2018.
- [15] L. Shin Byeong-Seok and Cheol-Su, "Obstacle Detection and Avoidance System for Visually Impaired People," *Haptic and Audio Interaction Design*, pp. 78-85, 2007.
- [16] T. Lewis, "Vibrating Clothes Could Help Blind People Navigate," January 2014. [Online]. Available: : <https://www.livescience.com/46236-vibrating-clothes-help-blind-navigate.html>.
- [17] D. Eagleman, "VEST: A Sensory Substitution Neuroscience Project [Online]. Available," March 2016. [Online]. Available: [https://www.kickstarter.com/projects/324375300/vest-a-sensory-substitution-neuroscience-project?ref=category\\_featured](https://www.kickstarter.com/projects/324375300/vest-a-sensory-substitution-neuroscience-project?ref=category_featured).
- [18] K. Rosmino, "Replacing white sticks with electronic devices: new technologies for the visually impaired," 26 July 2019. [Online]. Available: <https://www.euronews.com/2019/07/22/replacing-white-sticks-with-electronic-devices-new-technologies-for-the-visually-impaired>.
- [19] Brilhault, A.; Kammoun, S.; Gutierrez, O.; Truillet, P.; Jouffrais, C, "In Proceedings of the 4th IFIP International Conference on New Technologies, Mobility and Security (NTMS)," in *Fusion of artificial vision and GPS to improve blind pedestrian positioning*, Paris, 2011.
- [20] C. Liuyuan, "SensCap is a device that guides the visually impaired around obstacles," August 2009. [Online]. Available: [https://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2011/zb34\\_lc465/zb34\\_lc465/index.html](https://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2011/zb34_lc465/zb34_lc465/index.html).
- [21] M. McBride, "Sonar Glasses Obstacle Detection Device for Blind and Visually Impaired," 13 July 2016. [Online]. Available: <https://www.disabled-world.com/assistivedevices/visual/sonar-glasses.php>.
- [22] L. Hardesty, "Avoiding stumbles, from spacewalks to sidewalks," 22 July 2016. [Online]. Available: <http://news.mit.edu/2016/vibrating-footwear-astronauts-visually-impaired-0722>.
- [23] A. P. J. Petsiuk, "Low-Cost Open Source Ultrasound-Sensing Based Navigational Support for the Visually Impaired," *Sensors*, July 2019.



- [24] L. Hardesty, "Wearable system helps visually impaired users navigate," 31 May 2017. [Online]. Available: <http://news.mit.edu/2017/wearable-visually-impaired-users-navigate-0531>.
- [25] M. Nielsen, "Neural Networks and Deep Learning", 12 September 2018. [Online]. Available: <http://static.latexstudio.net/article/2018/0912/neuralnetworksanddeeplearning.pdf>.
- [26] "What Is Deep Learning?", December 2018. [Online]. Available: <https://www.mathworks.com/discovery/deep-learning.html>
- [27] Jia Yangqing, et. al., "Caffe: Convolutional architecture for fast feature embedding", *Proceedings of the 22nd ACM international conference on Multimedia*, March 2014.
- [28] Kyung-Wook N. "Vibration Pattern for the Implementation of Haptic Joystick", 6th International Conference, Proceedings, Part I, 2013.

## APPENDICES

1. JavaScript code for collecting the URLs for dataset images:

```
var script = document.createElement('script');

script.src = "https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js";

document.getElementsByTagName('head')[0].appendChild(script);

var urls = $(''.rg_di .rg_meta').map(function() { return JSON.parse($(this).text()).ou;
});

var textToSave = urls.toArray().join('\n');

var hiddenElement = document.createElement('a');

hiddenElement.href = 'data:attachment/text,' + encodeURIComponent(textToSave);

hiddenElement.target = '_blank';

hiddenElement.download = 'urls.txt';

hiddenElement.click();
```

2. Python script for the dataset images downloading:

```
from imutils import paths
import argparse
import requests
import cv2
import os

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-u", "--urls", required=True,
                help="path to file containing image URLs")
ap.add_argument("-o", "--output", required=True,
                help="path to output directory of images")
args = vars(ap.parse_args())
```

```

# collect the list of URLs from the file, then find the number of images should be
#downloaded
rows = open(args["urls"]).read().strip().split("\n")
total = 0

# loop
for url in rows:
    try:
        # try to download the image
        r = requests.get(url, timeout=60)

        # save the image to disk
        p = os.path.sep.join([args["output"], "{}.jpg".format(
            str(total).zfill(8))])
        f = open(p, "wb")
        f.write(r.content)
        f.close()

        print("[INFO] downloaded: {}".format(p))
        total += 1

    # if something went wrong
    except:
        print("[INFO] error downloading {}...skipping".format(p))

# loop over the image paths we just downloaded
for imagePath in paths.list_images(args["output"]):
    # initialize if the image should be deleted or not
    delete = False

    # try to load the image
    try:
        image = cv2.imread(imagePath)

        # if the image can be opened
        if image is None:
            print("None")
            delete = True

```

```

except:
    print("Except")
    delete = True

# check to see if the image should be deleted
if delete:
    print("[INFO] deleting {}".format(imagePath))
    os.remove(imagePath)

```

### 3. Python script for real-time object recognition:

```

# import the necessary packages
from imutils.video import VideoStream
from imutils.video import FPS
import numpy as np
import argparse
import imutils
import time
import cv2

# construct the argument parse and parse the arguments
arg = argparse.ArgumentParser()
arg.add_argument("-p", "--prototxt", required=True,
    help=" prototxt")
arg.add_argument("-m", "--model", required=True,
    help=" caffe")
args = vars(ap.parse_args())

# initialize the list objects should be detected
CLASSES = ["bus", "truck", "bicycle", "bench", "cup",
    "fork", "knife", "car", "cat", "chair", "spoon", "diningtable",
    "dog", "bed", "cell phone", "person", "microwave", "traffic light",
    "sofa", "sink", "tvmonitor"]
COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))

# load the model
print("[INFO] loading model...")

```

```

net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])

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

# loop over the frames from the video stream
while True:
    # grab the frame from the threaded video stream and resize it
    # to have a maximum width of 400 pixels
    frame = vs.read()
    frame = imutils.resize(frame, width=600)
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),
                                  0.007843, (300, 300), 127.5)

    # pass the frame over the network and obtain the detections
    net.setInput(blob)
    detections = net.forward()

    # loop over the detections
    for i in np.arange(0, detections.shape[2]):
        # extract the confidence (i.e., probability) associated with
        # the prediction
        confidence = detections[0, 0, i, 2]

        # show the output frame
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF

        # if the `q` key was pressed, break from the loop
        if key == ord("q"):
            break

# do a cleanup

```

```
cv2.destroyAllWindows()
vs.stop()
```

#### 4. Script for UDS:

```
#!/usr/bin/python

from gpiozero import InputDevice, OutputDevice, PWMOutputDevice
import time
from time import sleep
import numpy as np
import csv
import math
import datetime
from multiprocessing import Process

# Ports
trig1 = OutputDevice(4)
trig2 = OutputDevice(21)
trig3 = OutputDevice(25)

echo1 = InputDevice(17)
echo2 = InputDevice(22)
echo3 = InputDevice(8)

motor = PWMOutputDevice(14)

#calculate the time span between sending and obtaining impulse for 3 sensors
def get_pulse_time_3():
    trig3.on()
    sleep(0.00001)
    trig3.off()
    timeout = 0

#start recording when signal is sent, stop recording when signal is received
try:
    while (echo3.is_active) == False:
        pulse_start3 = time.time()
```

```

        while (echo3.is_active) and (timeout < 50) == True:
            timeout += 1
            pulse_end3 = time.time()

        timeout = 0
        sleep(0.06)
    #is something went wrong return the minimum time span
    return pulse_end3 - pulse_start3
except:
    return 0.02

def get_pulse_time_2():
    trig2.on()
    sleep(0.00001)
    trig2.off()
    timeout = 0

    try:
        while (echo2.is_active) == False:
            pulse_start2 = time.time()

        while (echo2.is_active) and (timeout < 50) == True:
            timeout += 1
            pulse_end2 = time.time()

        timeout = 0
        sleep(0.06)

        return pulse_end2 - pulse_start2
    except:
        return 0.02

def get_pulse_time_1():
    trig1.on()
    sleep(0.00001)
    trig1.off()
    timeout = 0

```

```

try:
    while (echo1.is_active) == False:
        pulse_start1 = time.time()

    while (echo1.is_active) and (timeout < 50) == True:
        timeout += 1
        pulse_end1 = time.time()

    timeout = 0
    sleep(0.06)

    return pulse_end1 - pulse_start1
except:
    return 0.02

#the distance calculation using a time span and speed of air, the distance is calculated
#in cm
def calculate_distance(duration):
    distance = duration * 17160.5 #to cm
    distance = round(distance, 2)

    #discard wrong results
    if distance > 400:
        distance = 400

    if distance < 0:
        distance = 0

    return distance

# write to file
def write_csv(data):
    with open('/home/pi/Desktop/data.csv', 'a') as outfile:
        writer = csv.writer(outfile, delimiter=',')
        writer.writerow(data)

# Main cycle

```



```
while (True):
```

```
    distance1 = calculate_distance(get_pulse_time_1())
    distance2 = calculate_distance(get_pulse_time_2())
    distance3 = calculate_distance(get_pulse_time_3())
    print("Dist1: ", distance1)
    print("Dist2: ", distance2)
    print("Dist3: ", distance3)
    print

    x = datetime.datetime.now()
    #write distances to the file using a sensors' angles)
    data = [x, distance1*cos(-0,698132), distance2, distance3*cos(0,698132)]
    write_csv(data)
```

5. Python script for gyro accelerometer:

```
#!/usr/bin/python

import time
import numpy as np
import csv
import smbus
import math
import datetime

# Register
power_mgmt_1 = 0x6b
power_mgmt_2 = 0x6c

#read the data
def read_byte(reg):
    return bus.read_byte_data(address, reg)

#read high and low 16-bits data
def read_word(reg):
    h = bus.read_byte_data(address, reg)
    l = bus.read_byte_data(address, reg+1)
```

```

    #concatenate them
    value = (h << 8) + l
    return value

def read_word_2c(reg):
    val = read_word(reg)
    #get a sign
    if (val >= 0x8000):
        return -((65535 - val) + 1)

    else:
        return val

def dist(a,b):
    return math.sqrt((a*a)+(b*b))

def get_y_rotation(x,y,z):
    radians = math.atan2(x, dist(y,z))
    return -math.degrees(radians)

def get_x_rotation(x,y,z):
    radians = math.atan2(y, dist(x,z))
    return math.degrees(radians)

def get_z_rotation(x,y,z):
    radians = math.atan2(z, dist(x,y))
    return math.degrees(radians)

bus = smbus.SMBus(0) # bus = smbus.SMBus(0) for Raspberry Pi version B
address = 0x68      # via i2cdetect

# write to file
def write_csv(data):
    with open('/home/pi/Desktop/data2.csv', 'a') as outfile:
        writer = csv.writer(outfile)
        writer.writerow(data)

# Main cycle

```

```

while (True):
    gyro_xout = read_word_2c(0x43)
    gyro_yout = read_word_2c(0x45)
    gyro_zout = read_word_2c(0x47)

    acceleration_xout = read_word_2c(0x3b)
    acceleration_yout = read_word_2c(0x3d)
    gyro_zout = read_word_2c(0x47)

    acceleration_xout = read_word_2c(0x3b)
    acceleration_yout = read_word_2c(0x3d)
    acceleration_zout = read_word_2c(0x3f)

    acceleration_xout_scaled = acceleration_xout / 16384.0
    acceleration_yout_scaled = acceleration_yout / 16384.0
    acceleration_zout_scaled = acceleration_zout / 16384.0

    x = datetime.datetime.now()
    data = [x, acceleration_xout_scaled, acceleration_yout_scaled,
acceleration_zout_scaled]
    write_csv(data)

```

6. Python script for plot and process the values and mapping:

```

import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as it
import scipy.signal as signal
from mpl_toolkits import mplot3d
from mpl_toolkits.mplot3d import Axes3D
from datetime import datetime
from array import array

# File under analysis.
filename = '/home/pi/Desktop/data.csv'

```

```

# Data preparation
allData = pd.read_csv(filename)
xAcc = allData.iloc[:, 1].values
yAcc = allData.iloc[:, 2].values
zAcc = allData.iloc[:, 3].values
Dist_1 = allData.iloc[:, 4].values
Dist_2 = allData.iloc[:, 5].values
Dist_3 = allData.iloc[:, 6].values

time = allData.iloc[:, 0].values
#set an epochs for easier calculations below
epoch = np.zeros(len(xAcc))
for i in range(len(xAcc)):
    if i == 0:
        epoch[0] = 0.000
    else:
        epoch[i] = epoch[i-1] + 1
    xAcc[i] = float(xAcc[i])
    yAcc[i] = float(yAcc[i])
    zAcc[i] = float(zAcc[i])
    Dist_all[i] = float(Dist_all[i])

epoch1 = np.zeros(len(Dist_1))
for i in range(len(Dist_1)):
    if i == 0:
        epoch1[0] = 0.000
    else:
        epoch1[i] = epoch1[i-1] + 1
    Dist_1[i] = float(Dist_1[i])
    Dist_2[i] = float(Dist_2[i])
    Dist_2[i] = float(Dist_2[i])

# ultrasonic data before filtering
plt.figure(figsize = (10, 6))
plt.plot(epoch1, Dist_1, label = '1', color = 'r', linewidth = 1, alpha = 0.6)
plt.plot(epoch1, Dist_2, label = '2', color = 'g', linewidth = 1, alpha = 0.6)
plt.plot(epoch1, Dist_3, label = '3', color = 'b', linewidth = 2, alpha = 0.6)
plt.title('Ultrasonic sensors data before filtering')

```

```
plt.legend(loc = 'upper left')
plt.xlabel('Time (s)')
plt.ylabel('Distance (cm)')
plt.xlim()
plt.ylim()
plt.grid()
plt.show()
```

```
#filtering data using median filter with kernel = 5
```

```
Dist_1 = signal.medfilt(Dist_1, 5)
```

```
Dist_2 = signal.medfilt(Dist_2, 5)
```

```
Dist_3 = signal.medfilt(Dist_3, 5)
```

```
# ultrasonic data after filtering
```

```
plt.figure(figsize = (10, 6))
```

```
plt.plot(epoch1, Dist_1, label = '1', color = 'r', linewidth = 1, alpha = 0.6)
```

```
plt.plot(epoch1, Dist_2, label = '2', color = 'g', linewidth = 1, alpha = 0.6)
```

```
plt.plot(epoch1, Dist_3, label = '3', color = 'b', linewidth = 2, alpha = 0.6)
```

```
plt.title('Ultrasonic sensors data before filtering')
```

```
plt.legend(loc = 'upper left')
```

```
plt.xlabel('Time (s)')
```

```
plt.ylabel('Distance (cm)')
```

```
plt.xlim()
```

```
plt.ylim()
```

```
plt.grid()
```

```
plt.show()
```

```
# Plotting X Acceleration
```

```
plt.figure(figsize = (10, 6))
```

```
plt.plot(epoch, xAcc, label = 'X', color = 'r', linewidth = 1)
```

```
plt.title('X Acceleration')
```

```
plt.ylabel('Acceleration (g)')
```

```
plt.xlabel('Time (s)')
```

```
plt.xlim()
```

```
plt.ylim(-1, 1)
```

```
plt.grid()
```

```
plt.show()
```

```

# Plotting Y Acceleration
plt.figure(figsize = (10, 6))
plt.plot(epoch, yAcc, color = 'g', linewidth = 1)
plt.title('Y Acceleration')
plt.ylabel('Acceleration (g)')
plt.xlabel('Time (s)')
plt.xlim()
plt.ylim(-1, 1)
plt.grid()
plt.show()

# Plotting Z Acceleration
plt.figure(figsize = (10, 6))
plt.plot(epoch, zAcc, color = 'b', linewidth = 1)
plt.title('Z Acceleration')
plt.ylabel('Acceleration (g)')
plt.xlabel('Time (s)')
plt.xlim()
plt.ylim(-1, 2)
plt.grid()
plt.show()

# Plotting all accelerations
plt.figure(figsize = (10, 6))
plt.plot(epoch, xAcc, label = 'X', color = 'r', linewidth = 1, alpha = 0.6)
plt.plot(epoch, yAcc, label = 'Y', color = 'g', linewidth = 1, alpha = 0.6)
plt.plot(epoch, zAcc, label = 'Z', color = 'b', linewidth = 2, alpha = 0.6)
plt.title('X, Y, and Z Acceleration')
plt.legend(loc = 'upper left')
plt.xlabel('Time (s)')
plt.ylabel('Acceleration (g)')
plt.xlim()
plt.ylim(-1, 2)
plt.grid()
plt.show()

# Butterworth Filtering X and Y Acceleration
N = 2 # Filter order

```

```

Wn = 0.9 # Cutoff frequency 0 < Wn < 1
B, A = signal.butter(N, Wn, output = 'ba')
xAcc = signal.filtfilt(B, A, xAcc)
yAcc = signal.filtfilt(B, A, yAcc)
plt.figure(figsize = (10, 6))
plt.plot(epoch, xAcc, linewidth = 1, color = 'r')
plt.plot(epoch, yAcc, linewidth = 1, color = 'g')
plt.title('X and Y Acceleration Denoised')
plt.xlabel('Time (s)')
plt.ylabel('Acceleration (g)')
plt.xlim()
plt.ylim()
plt.grid()
plt.show()
plt.show()

# First integration: generating velocity
xVel = it.cumtrapz(xAcc, epoch)
yVel = it.cumtrapz(yAcc, epoch)
zVel = it.cumtrapz(zAcc, epoch)

# Plotting the velocities
plt.figure(figsize = (10, 6))
epoch = np.delete(epoch, -1)
plt.plot(epoch, xVel, linewidth = 2, alpha = 0.7, label = 'X', color = 'r')
plt.plot(epoch, yVel, linewidth = 2, alpha = 0.7, label = 'Y', color = 'g')
plt.plot(epoch, zVel, linewidth = 2, alpha = 0.7, label = 'Z', color = 'b')
plt.title('X, Y, and Z Velocities')
plt.xlabel('Time (s)')
plt.ylabel('Velocity (cm/s)')
plt.legend(loc = 'upper left')
plt.grid()
plt.show()

# Second integration: generating path
xDis = it.cumtrapz(xVel, epoch)
yDis = it.cumtrapz(yVel, epoch)
zDis = it.cumtrapz(zVel, epoch)

```

```

# 3D Trajectory Plotting
plt.figure(num = None, figsize=(10, 8), dpi=80, facecolor = 'w', edgecolor='b')
ax = plt.axes(projection = '3d')
ax.plot3D(xDis, yDis, zDis, 'red', label = 'Trajectory', linewidth = 2)
ax.set_xlabel('X DISTANCE [cm]', fontsize = 12)
ax.set_ylabel('Y DISTANCE [cm]', fontsize = 12)
ax.set_zlabel('Z DISTANCE [cm]', fontsize = 12)
ax.set_xlim3d()
ax.set_ylim3d()
ax.set_zlim3d()
plt.legend(loc = 'upper left')
plt.title('Location Trajectory (Accelerometer)')
plt.show()

```

```

fig = plt.figure()
ax = fig.gca(projection='3d')
# Plot a sin curve using the x and y axes.
x = xDis
y = yDis
ax.plot(x, y, zs=0, zdir='z', label='curve in (x,y)')
#ax.plot3D(xDis, yDis, zDis, 'red', label = 'Trajectory', linewidth = 2)
# By using zdir='z', the y value of these points is fixed to the zs value 0
# and the (x,y) points are plotted on the x and y axes.
ax.scatter(x, Dist_all-100, zs=0, zdir='z', label='points in (x,y)')
# Make legend, set axes limits and labels
ax.legend()
ax.set_xlim()
ax.set_ylim()
ax.set_zlim()
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
# Customize the view angle so it's easier to see that the scatter points lie
# on the plane y=0
ax.view_init(elev=20., azimuth=-35)
plt.show()

```

```

# 2D Trajectory Plotting

```



```

plt.figure(figsize = (10, 6))
plt.scatter(xDis, yDis, linewidth = 0.1, alpha = 0.7, label = 'XY Coord.', color = 'r')
plt.title('2D Trajectory ((X, Y) Coordinates)')
plt.xlabel('X Distance (cm)')
plt.ylabel('Y Distance (cm)')
plt.xlim()
plt.ylim()
plt.grid()
plt.show()

```

## 7. Python starting script

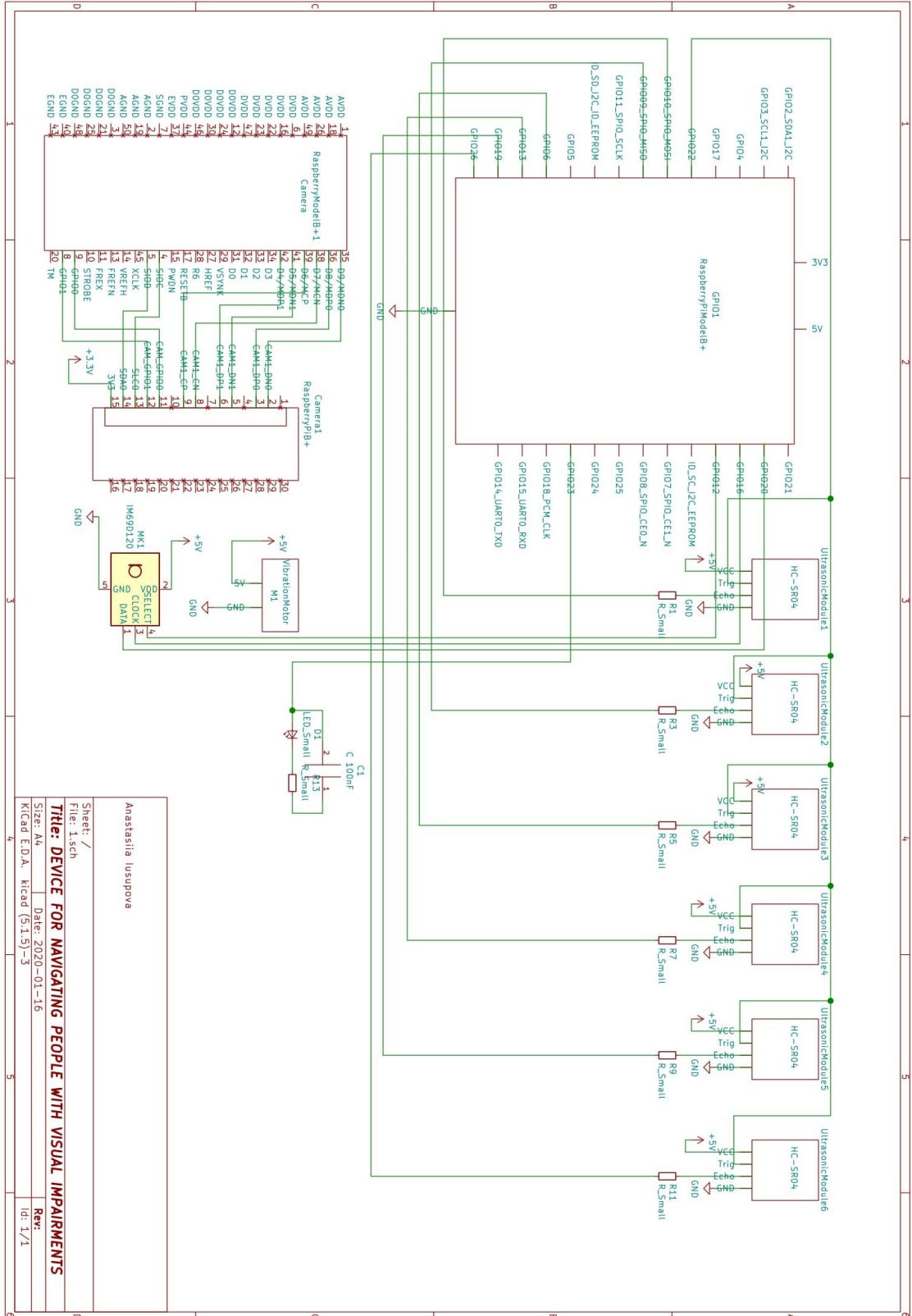
```

#!/usr/bin/python
import subprocess
import time
code1 = 'python sensors.py' #code for record data from sensors
code2 = 'python object_detection.py' # code for object detection
code3 = 'python map.py' #code for mapping
data_file = open("/home/pi/Desktop/data.csv", "rw+")
def mapping_buffer(seconds): #run mapping for 10 minutes
start = time.time()
    time.clock()
    elapsed = 0
    while elapsed < seconds:
        elapsed = time.time() - start
        script_3 = subprocess.Popen(code3, stdout=subprocess.PIPE,
            shell=True)
def main():
    while True:
script_1 = subprocess.Popen(code1, stdout=subprocess.PIPE, shell=True) #run first
script
script_2 = subprocess.Popen(code2, stdout=subprocess.PIPE, shell=True) #run second
script
time.sleep(60) #wait for data collection
mapping_buffer(600) #run mapping for 10 minutes
    data_file.truncate() #clear file with data to restart the map

```

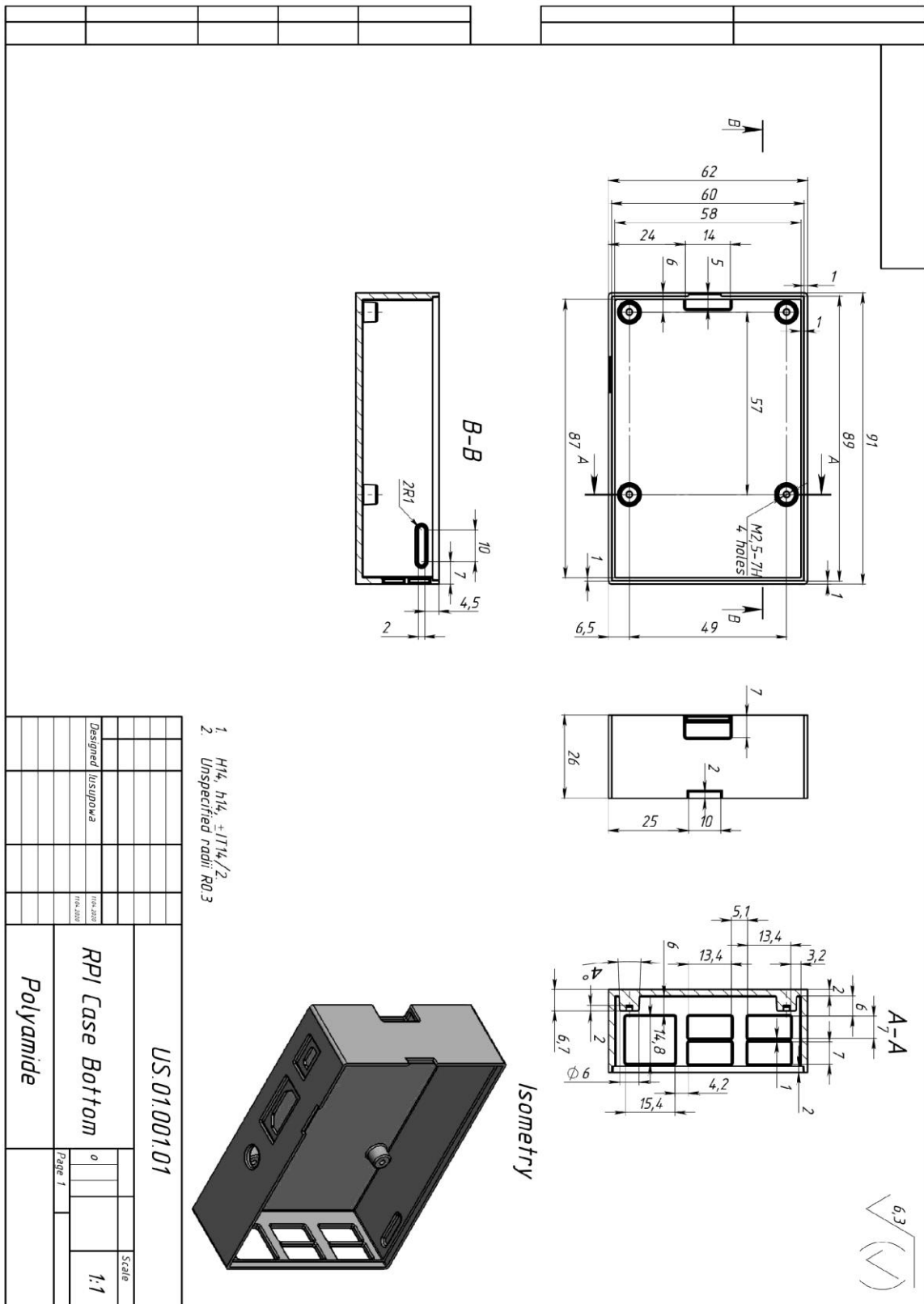
# GRAPHICAL MATERIAL

The electronics connection scheme:

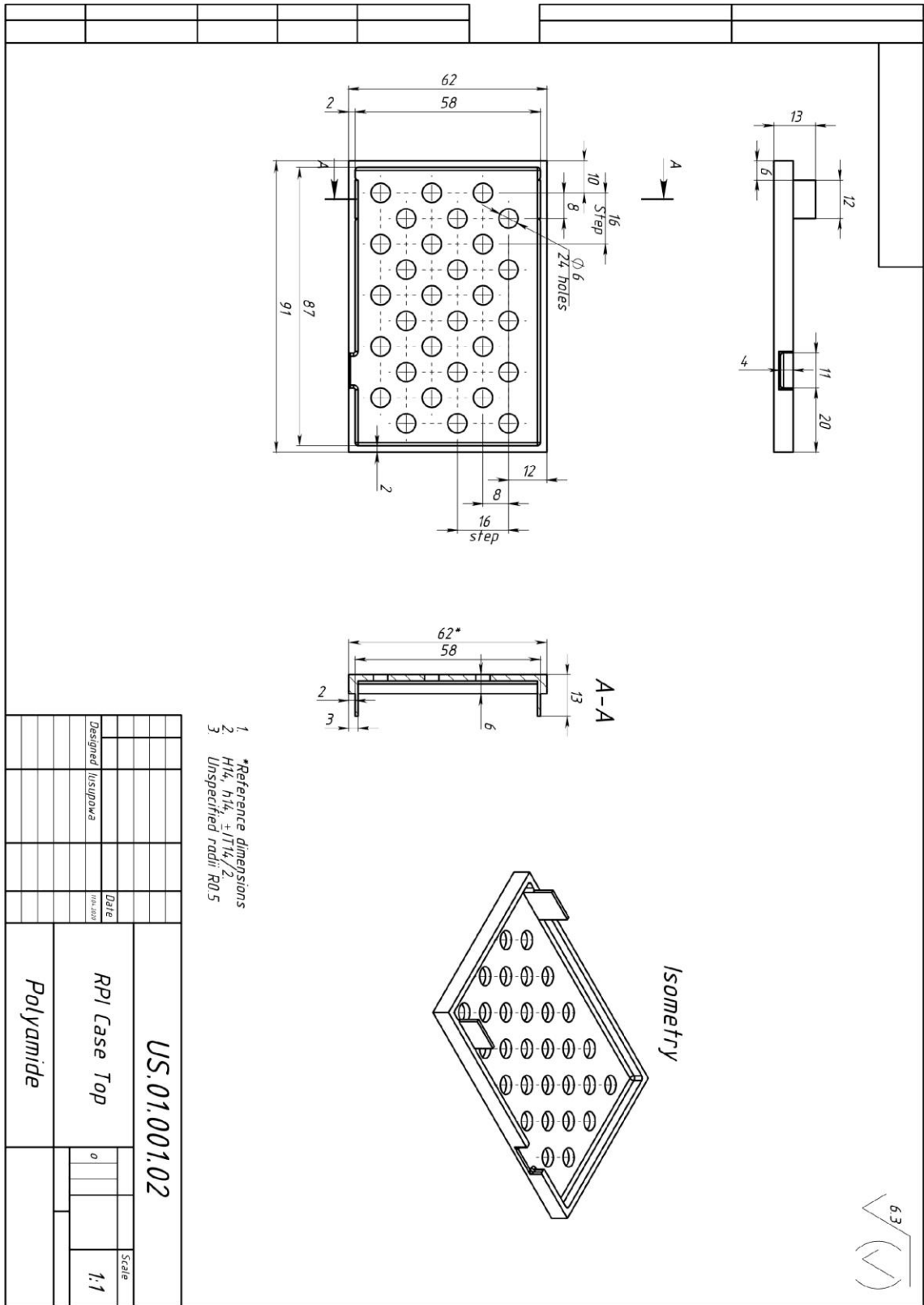


Sheet: /	File: 1.sch
Size: A4	Date: 2020-01-16
Kicad E.D.A. Kicad (5.1.5)-3	Rev: 1/1
Anastasia Iusupova	
<b>TIRE: DEVICE FOR NAVIGATING PEOPLE WITH VISUAL IMPAIRMENTS</b>	

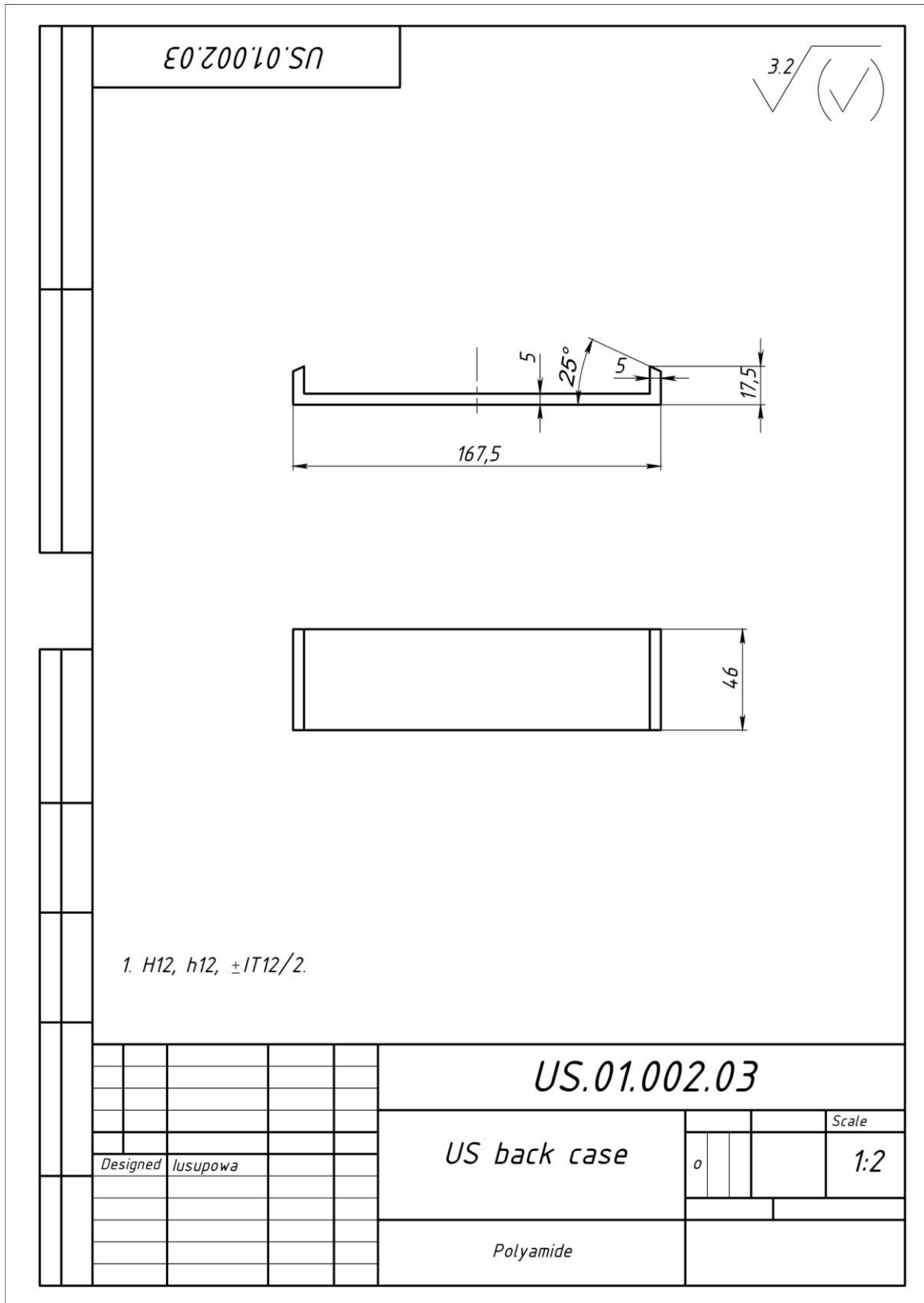
The drawing of the Raspberry Pi body, the bottom part:



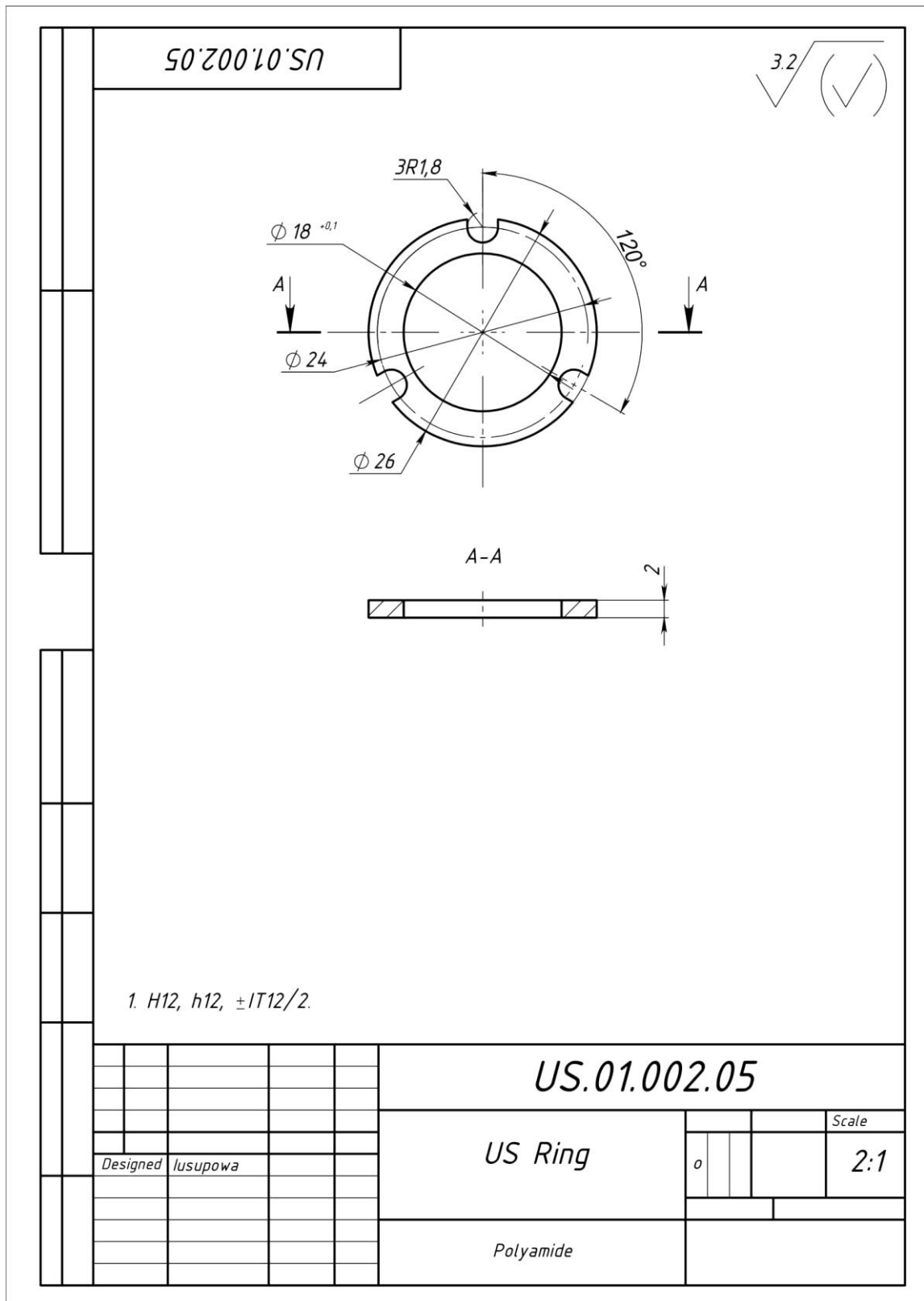
The drawing of the Raspberry Pi body, the top part:



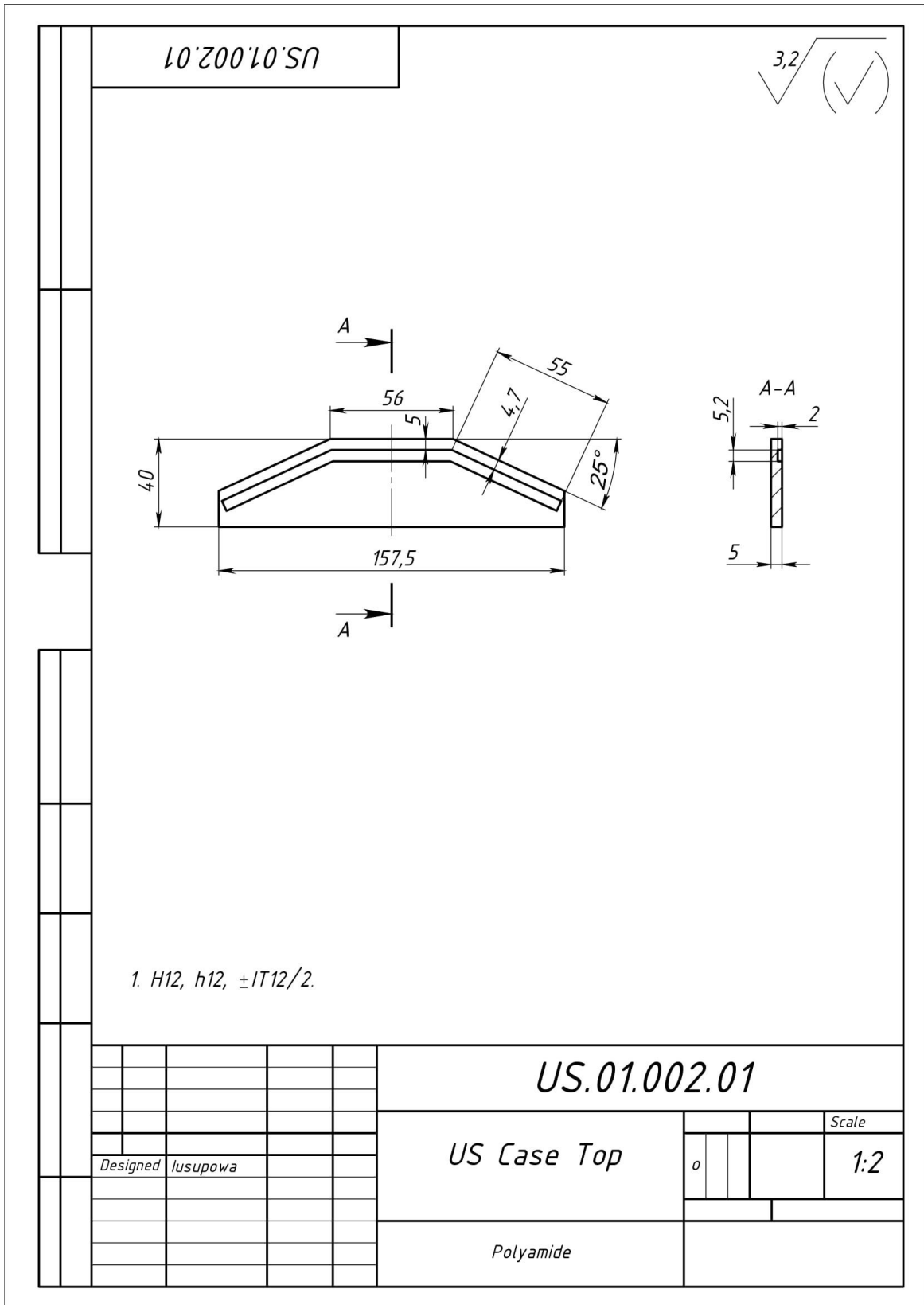
The drawing of the back case for the ultrasonic body:



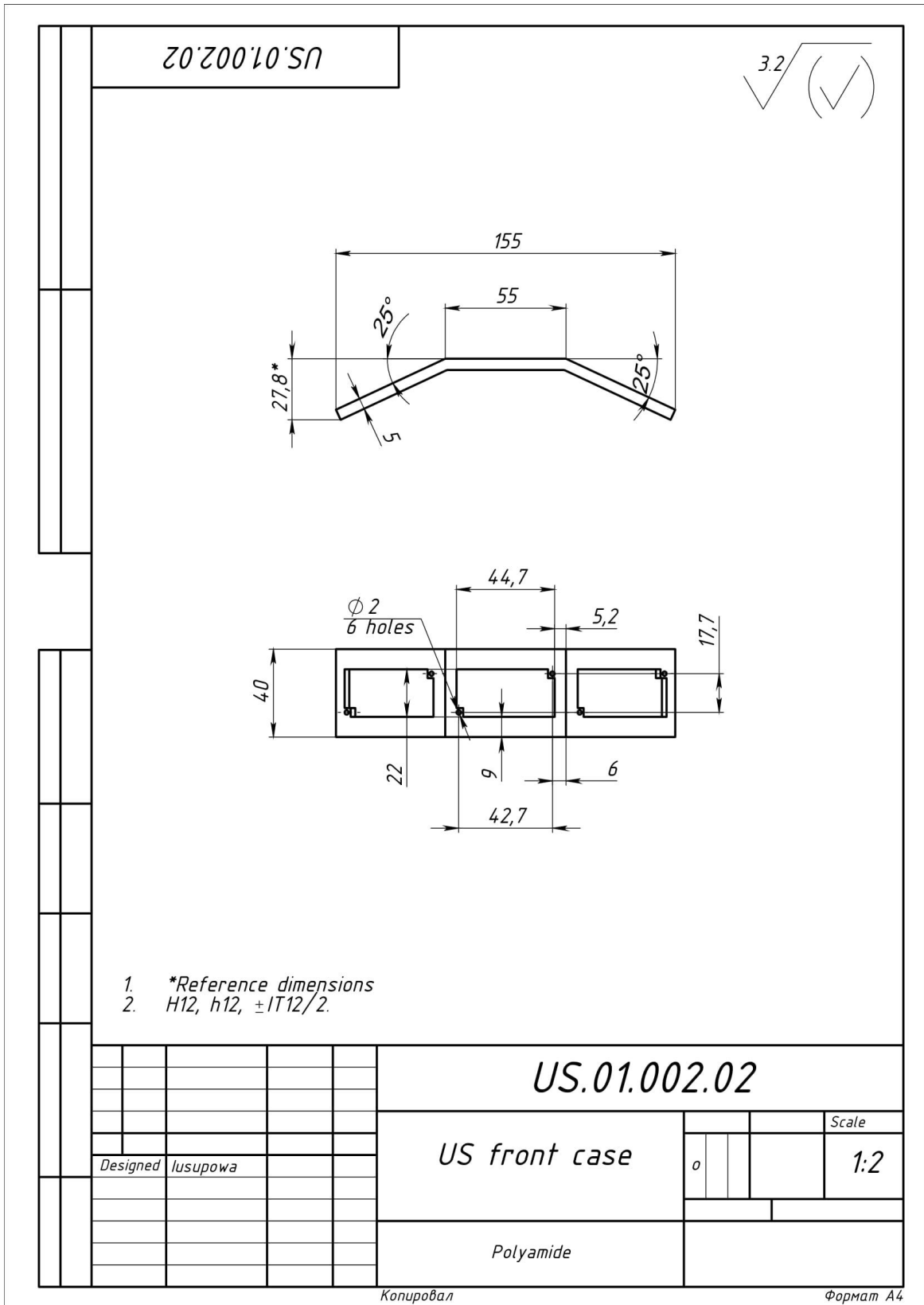
The drawing of the protecting ring for the ultrasonic body:



The drawing of the top case for the ultrasonic body:

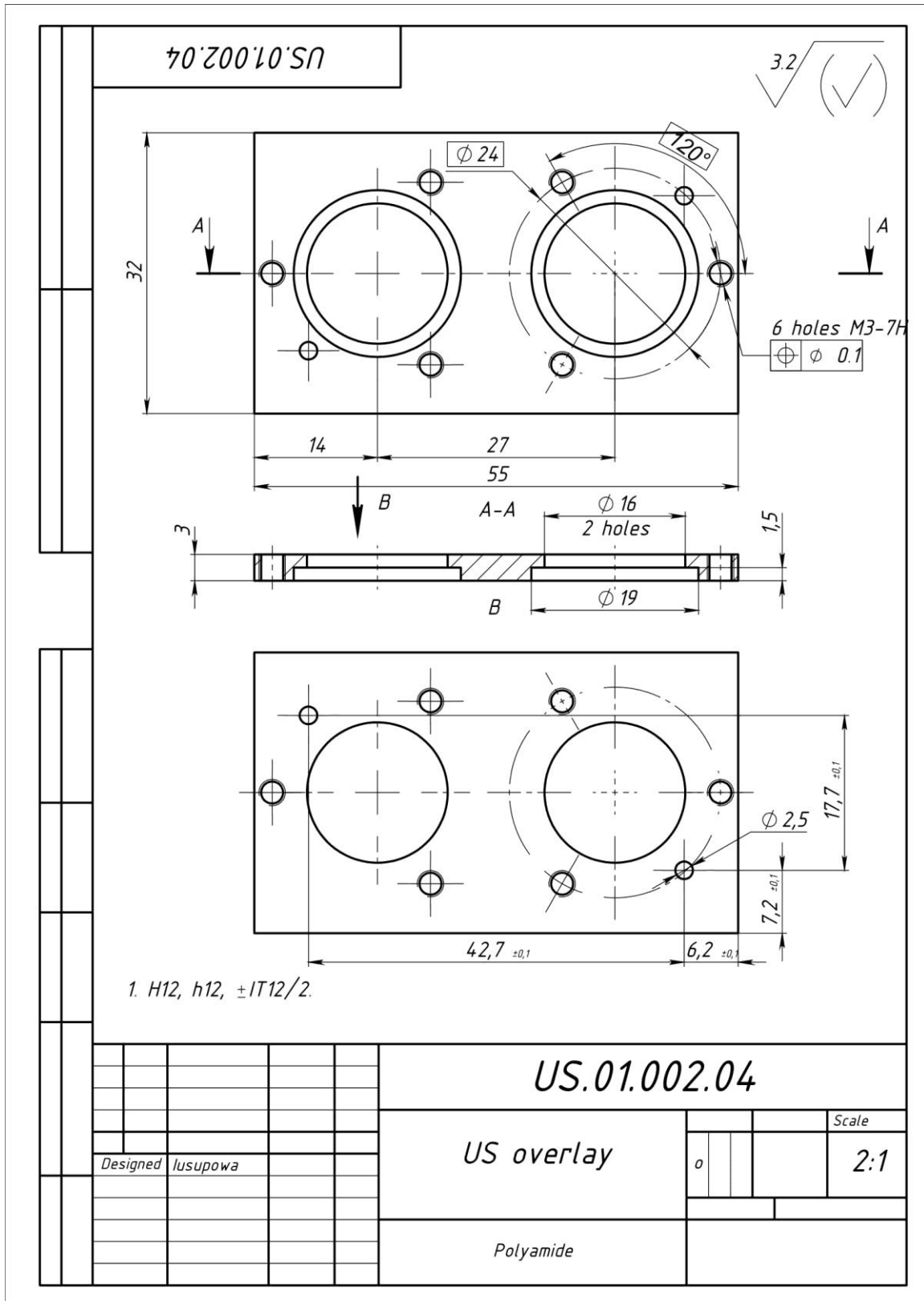


The drawing of the front case for the ultrasonic body:

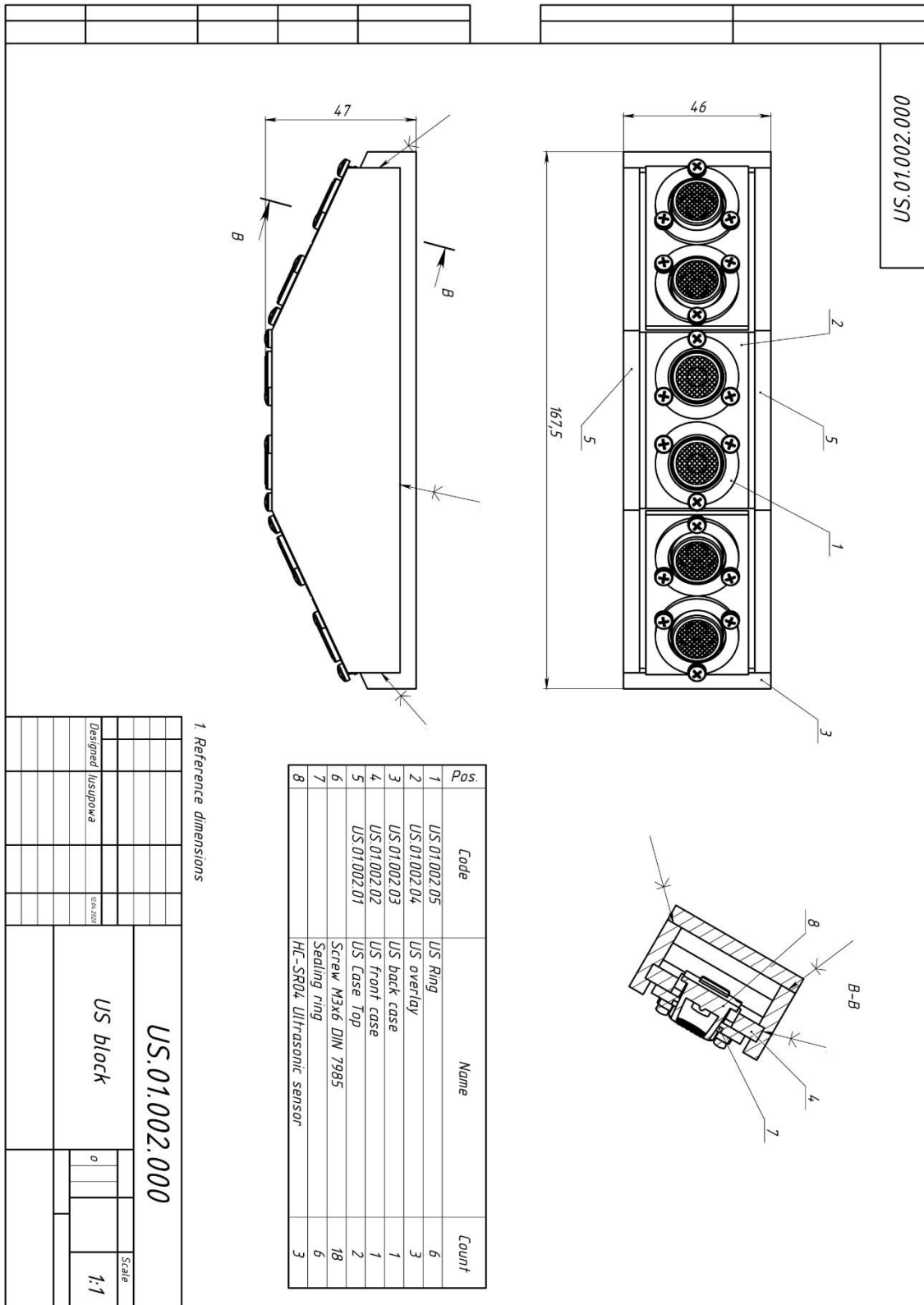




The drawing of the overlay case for the ultrasonic body:



The drawing of the assembled case for the ultrasonic body:



The rendering of the vest prototype, front:



The rendering of the vest prototype, back:

