TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

AIBIKE YESSIMBEKOVA

Master of Science Thesis

"Autonomous Navigation of Mobile Robots: Marker-based Localization System and On-line Path"

Examiner:
Prof. Jose L. Martinez Lastra
Examiner and topic approved on 3th
of January 2018

ABSTRACT

AIBIKE YESSIMBEKOVA:
Tampere University of technology
Master of Science Thesis 60 pages, 11 Appendix pages
September 2018
Master's Degree in Automation Engineering,
Major: Factory Automation and Industrial Informatics
Examiner: Professor Jose L. Martinez Lastra

Keywords: autonomous navigation, intelligent wheelchair, indoor navigation, human-computer interaction

Traditional wheelchairs are controlled mainly by joystick, which is not suitable solution with major disabilities. Current thesis aiming to create a human-machine interface and create a software, which performs indoor autonomous navigation of the commercial wheelchair RoboEye, developed at the Measurements Instrumentations Robotic Laboratory at the University of Trento in collaboration with Robosense and Xtrensa,. RoboEye is an intelligent wheelchair that aims to support people by providing independence and autonomy of movement, affected by serious mobility problems from impairing pathologies (for example ALS – amyotrophic lateral sclerosis).

This thesis is divided into two main parts – human machine interface creation plus integration of existing services into developed solution, and performing possible solution how given wheelchair can navigate manually utilizing eye-tracking technologies, TOF cameras, odometric localization and Aruco markers.

Developed interface supports manual, semi-autonomous and autonomous navigation. In addition to that following user experience specific for eye-tracking devices and people with major disabilities. Application delevoped on Unity 3D software using C# script following state-machine approach with multiple scenes and components.

In the current master thesis, suggested solution satisfies user's need to navigate hands-free, as less tiring as possible. Moreover, user can choose the destination point from defined in advance points of interests and reach it with no further input needed. User interface is intuitive and clear for experienced and unexperienced users. The user can choose UI's icons image, scale and font size. Software performs in a state machine module, which is tested among users using test cases. Path planning routine is solved using Dijkstra approach and proved to be efficient.

PREFACE

This Master of Science Thesis is made at the Laboratory of Measurements Instrumentation and Robotics in the University of Trento as a part of the exchange studies. This work is not a usual thesis, since it is a continuation of an existing project, which aims to improve the life of people with limited mobility. From the beginning of my studies, I always wanted to create a product that can be beneficial for people. I was looking for a right topic for a while and today I am happy that I was a part of this important project.

The process of working on this Master thesis was challenging not only from the technical point of view but also from also from the personal side. Even though I had a great opportunity to explore a new country, culture, language, working environment, sometimes I found it very difficult to adapt, since Italian lifestyle is completely different not only from my home country life but from also Finnish one. Unfortunately, not every step of the thesis work was rewarding. Often I spent a lot of time to find a great solution, but instead, I kept finding thousands of ways that did not work. However, when the solution was ready I felt a great happiness and satisfaction, so I came back to Finland relieved. I was lucky to have a support from right people during the project completion; I want to express my gratitude to all of them.

First, I want to thank Professor Mariolino de Cecco for giving me an opportunity to be a part of RoboEye challenge. Also thank you all the MIRO team members, special thanks to Luca Maule for being patient and supportive every time I needed.

I want also to thank Professor Jose L. Martinez Lastra for examining this thesis.

I would like to thank my friends who helped me to stay motivated and productive. The great gratitude to those who tested my solution and gave the honest feedback.

Last but not the least, thanks to my family who always supports me.

In Tampere, Finland, 2018

Aibike Yessimbekova

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

TUT	Tampere University of Technology
ALS	Amyotrophic Lateral Sclerosis
HMI	Human Machine Interface
UI	User Interface
AR	Augmented Reality
EOG	Electrooculography
EMG	Electromyography
LIDAR	Light Identification Detection and Ranging
PLC	Programmable Logic Controller
ROS	Robot Operating System
RGB-D	a combination of a RGB (Red, Green, Blue) image and its corresponding Depth image
TOF	Time of flight
CAD	Computer aided design
JSON	JavaScript Object Notation
ID	Identification number
PWM	Pulse-Width Modulation
APP	Application Software
LED	Light-emitting Diode
LUX	International System of Units of illuminance
RFID	Radio-Frequency identification
GPS	Global Positioning System
FPS	Frame-per-second
POI	Point of Interest
IC	Integrated circuit
DLL	Data Link Layer
OS	Operating System

LIST OF SYMBOLS

Symbol	*SI unit*	*Description*
T	[s]	the sampling time
V	[m/s]	speed
Θ	[˚]	orientation in Cartesian space
X, Y	[m]	position of a wheelchair in Cartesian space
R_R and R_L	[m]	Right and left wheels radius
n_{Rk} and n_{Lk}	[-]	the number of counts from the right and left encoders respectively between two subsequent step
n_o	[-]	the number of counts of the encoder
x_k, y_k	[m]	the estimated position
δ_k	[m]	the estimated attitude
y_{NAZ}	[m]	*y* normalized at zero is set to be *0.25*
y_P	[m]	the actual position of eye gaze on a monitor
X_{NAZ}	[m]	*x* normalized at zero, is set to *0.15*
xP	[m]	the actual position of the eye on the screen
W	[m]	a width of the screen
$V\ lat\ M$	[m/s]	the maximum values of the lateral speed.

. LIST OF EQUATIONS

1. INTRODUCTION

1.1 Motivation and Justification

Development of manually powered and electrical wheelchairs for people suffering from the paralyzed condition is not wide. Standard wheelchairs usage assumes that a person still can move their hands; however, that excludes paralyzed people from an end-user group. Various diseases or accidents influencing the nervous system also might cause paralysis which is divided into three groups: local, global and specific. Unfortunately, most of the paralysis diseases are constant, but sometimes this condition is temporary. The most-well known victim of Amyotrophic Lateral Sclerosis which cases paralysis was a scientist Stephen W. Hawking. He was using a wheelchair on a constant base. The most limiting disabilities are those that prevent a person to move independently. People who are not able to use their arms and legs encounter severe problems and absolutely depend on others. This creates a need to have a device that can ensure independent living for disabled people. It is obvious that the primary solution is to have an electric wheelchair that can be controlled mechanically by any mobile part of the body under their own command.

Today, many solutions of wheelchair are controlled by joysticks using chin or mechanical devices attached to the headrest. However, those solutions are lacking the user comfort because these people have to move their chin or head at all times in order to navigate. Additionally, chin movements are limited and cannot ensure the desired level of control. Also using chin to control a joystick is cosmetically unpleasant for the user. Other chairs can be controlled by user's puffing or sipping a plastic tube. This method has several disadvantages such as increased difficulty in adjusting control plus sanitation problems. All the above-mentioned solutions case disturbance for the users and are found to be very tiring. Moreover, it is hard for a user to master control.

Existing wheelchairs are used only for disabled and elderly people that can partially move their limbs. In contrast, there are cases when people are affected by serious diseases causing the paralyzed condition and the only agile parts are eyes. Therefore, nowadays available for sale wheelchairs controlled by the joystick are not useful for people suffering from ALS or Parkinson diseases due to very limited mobility. Wheelchairs suitable to be used for paralyzed people can be divided in the following manner:

1. Biosignal-based, thus EOG and EMG might transfer bio signals received from the brain into a control signal that can be used to control an electric wheelchair. For example, when one of the above-mentioned methods perform analysis on the user's eye movement by directly connecting recognition device's electrodes on the eye. On the other hand, brain signals and muscle signals can be used to control the wheelchair, if the output signal is converted accordingly into the control signal.
2. Voice-based. In this case, a wheelchair is controlled by means of speech recognition systems, user interface, etc. While controlled by voice, the patient can pronounce command to the wheelchair to move to the desired position, for example, left, right, forward, etc.
3. Vision-based. The user intents are captured and transformed into a control signal with help of a camera. Some of those are head gestures, horizontal eye gaze, and blinking. Also, this cameras are widely used to detect obstacles, create a map used for autonomous navigation and provide an image of the surrounding environment is required.

Extensive research [1] has shown that wheelchairs controlled by voice, brain control system and vision can fulfill the needs of people with abovementioned diseases. Voice controlled systems do not work well when used in a noisy environment, the voice command might be confused or not detected. In brain-controlled devices, using EEG signals it is easy to navigate, but setting up the system might be too demanding and somehow inconvenient for the patient. Even though these systems save a significant amount of energy and require less external manpower, it might still be very difficult for a paralyzed person to use it.

To address these issues many wheelchairs have been upgraded with equipment such as cameras, sensors, equipment and technologies such as image processing, simultaneous navigation, and localization, moving these wheelchairs into a category of a "smart" wheelchairs. Semi-autonomous, autonomous navigation, mapping, and obstacle avoidance make these chairs "intelligent" and extremely comfortable for the user. In addition to that, vision-based control does not require direct contact, e.g. electrodes attached to the body of the user. However, with the advanced technical capabilities arises cost of maintaining and creating such wheelchairs. In addition to that, these systems are often not reusable, so thus changing from one wheelchair to another might be very costly. Another problem is a human-machine interface, which is not user-friendly. Most of the smart wheelchairs are used mainly for the research purposes, lacking commercialization.

One of the important issues of any commercial product is its price. Nowadays, most robotics applications use highly precise and very expensive sensors in order to provide the

desired output, for example, LIDAR-based lasers. Not every person with limited abilities can afford an expensive wheelchair. Moreover, often it is not sufficient to have an ambiguous technology in order to satisfy everyday needs. Therefore, it is important to find a right balance between cost and performance. Another challenge, the algorithms required to run a mobile-wheelchair are demanding, which require a personal computer with great performance characteristics. Moreover, physical interface is poor on the most of wheelchairs, and there is no standard in communication protocols required by wheelchair's input devices and various modules. This problem can be solved by utilizing common robotic frameworks. Last but not the least, currently wheelchairs are not accepted much by society and clinic. It remains immature technology, which is hardly desired and appreciated by disabled people.

1.2 Problem statement

The number of wheelchair users increases dramatically, thus creating a comfortable and up-to-date robotic wheelchair is important. The desired wheelchair should be able to plan a path fast and efficiently, while the user should feel calm and comfortable. The wheelchair should be intractable with the user, thus semi-autonomy is preferred rather than being completely autonomous. Moreover, the mobile robot should have the ability to perceive data not only at the very beginning of a trip but also during its navigation, thus efficient user interface is crucial.

In addition to that, the user interface is required to be easy-customizable in order to satisfy the needs of each user. If it is impossible to adapt to the specific user the system will fail to satisfy basic needs of the patient. It is well-known fact that assistance robot shall improve the user's ease to move independently yet be able to work in pair with the user. It is not required to have a high-level path planning for its navigation but rather to keep the patient safe and comfortable. A system is required to be more than a navigation system. It is important to create a system able to assist the user in navigation providing safety and ease.

This master thesis is a part of a commercial RoboEye project [2]. RoboEye aims to support people affected by mobility problems that range from very impairing pathologies (like ALS, amyotrophic lateral sclerosis) to old age. In this context, mobile robotic can play a key role to improve autonomy and lifestyle of the patients. The focus of this project is the restore of users' mobility using novel technology based, in this case, on the gaze. At this point the navigation of the wheelchair can be performed using different methods:

• Direct navigation by the user;

• Semi-Autonomous navigation.

In order to form a testbed an electric wheelchair is used. The two original motors are connected with an industrial PLC that performs the low-level control of the device (odometric localization, path following, safety). A Windows PC, connected with the PLC, runs the high-level application that consists of the HMI, the communication with external devices (Kinect, Eye tracker, and monitor) and the path planning. However, the user interface of the developed prototype needs to be improved. The functionality needs to be extended offering competitive value to the start-up company owning this product.

The thesis was performed as a part of an exchange program between the Tampere University of Technology and The University of Trento, Italy. Given thesis was performed as a part of a teamwork project, which is not a standalone research topic but it is a part of a large continuous activity started more than one year ago.

1.3 Objectives

The aim of the master thesis is to develop an application designed to move a wheelchair from one point to another inside a civil environment. The main goal is the development of an autonomous navigation software in order to give to the patient a novel control technique. This software has to be integrated into the main application of the current project. With the autonomous navigation, the user can select a target point, where a person wants to arrive, and then the wheelchair has to reach the goal without any other additional input. This type of navigation aims to be as little tiring as possible.

The autonomous navigation application research is divided into three major objectives:

• Development of a graphical interface easy to use for people affected by mobility problems. In particular, the user has to be able to select the target position from a map of the environment displayed on the monitor.

• Localization of the wheelchair inside the environment using some visual tags mounted in key points (doors, tables, etc.). This part will start with two algorithms developed in the lab during past works.

• Planning of the path to reach the selected target. The environment has to be represented with a graph base schema and then used to find the path. The collaboration with patients allows evaluating all the parts in order to reach better the needs of the user.

As explained in the introduction it is clear that in order to complete above-mentioned objectives it is important to answer following questions:

- What would be the most suitable solution to provide accurate mapping of an indoor environment? Moreover, this solution should not be source demanding and expensive. To be short, it should be simple, inexpensive and be suitable for autonomous navigation in the flat of a patient.
- How to localize a wheelchair inside environment? The localization of the wheelchair inside the environment is necessary to allow a mobile robot to move autonomously. Therefore, the localization should be effective and cheap.
- Which algorithm would be the best option to perform the shortest path planning from the position localized by the wheelchair to the desired position chosen by the user?
- In which way a human-machine interface should be developed in order to solve above-mentioned problems. The UI should give a user opportunity to navigate in manual, semi-autonomous and autonomous modes, to change settings and to see the map of the environment when is necessary. Moreover, software should include previous prototype's functionalities (for example Kinect data acquisition) and should work on a given configuration (hardware configuration should not be changed). It is important to choose how disabled people should control a wheelchair.

1.4 Outline

The thesis is composed of five themed chapters, including Introduction, Existing solutions, Methods and Algorithm, An Overview of RoboEye project and Results of the thesis.

The main issues addressed in this paper are: a) Human machine interface development b) Path planning issues c) Wheelchair solutions for disabled people

Chapter two begins by laying out the theoretical dimensions of the research and looks at how some researches were trying to create an intelligent wheelchair aiming to ensure manual, autonomous, semi-autonomous navigation of the wheelchair. It suggests some tricks on methods and algorithms of a software development on Unity 3D.

The third chapter is concerned with methodology used for this study. In order to achieve set objectives the author researched path planning methods, localization approaches, and human machine interface development with high user experience and software development strategies. The chosen approach is discussed in this chapter. In other words, theoretical approach is presented in this chapter.

The fourth section presents the finding of the research. Here presented results of the simulation of path planning algorithm, test cases results tested by two groups of users and user's satisfaction survey results.

Chapter 5 concludes whether problem is solved or not and analyses the results of work done and focuses on limitations of the study, in addition to that proposing future improvements for further research.

Appendix contains block diagrams for the each state of the state machine based software and main software architecture scheme.

2. LITERATURE AND BEST PRACTICES REVIEW

Research of the intelligent wheelchairs does not have a long history. The first serious discussions and analyses of smart wheelchair emerged during the 1986, where these devices were controlled and navigated by means of vision. David L. Jaffe [3] suggested using a totally non-contacted electrically powered wheelchair to provide mobility to disabled people, addressing problems of collision avoidance and tracking along a straight path. In this manner the term "smart" electric wheelchair appeared. This prototype and research have established that set goals are achievable. The given model uses Polaroid Ultrasonic Sensor Technology in order to achieve mentioned goals. Polaroid sensors are utilized to detect subject-to-camera distance needed to focus. It is required to triangulate the user's head position on the wheelchair. [3].

Next, during the next 40 years of extensive researchers' work, many various intelligent wheelchairs from different countries appeared. For example, Wheelesley of Massachusetts Institute of Technology [4], wheelchair named SIAMO developed in Spain [5] and numerous work from China such as head movements controlled wheelchair platform of Shenzhen Institute of Advanced Technology, Chinese Academy of Science, etc. Therefore, there are many different solutions for the stated problem. However, the developed solutions have their pros and cons being expensive, too complex and not reusable in general. The subsections below describe some outstanding solutions to various wheelchair problems, which combination might be an ultimate solution to problems discussed in chapter 1 of the given thesis.

2.1 Autonomous Exploration and Navigation

Recently, numerous researchers have attempted to design an intelligent wheelchair ensuring low cost and high reusability yet realizing navigation and autonomous exploration. For example, on 10th International Symposium on Computational Intelligence and Design, which is held on 2017, a low cost highly reusable wheelchair was presented.

Suggested solution runs on ROS – open source framework, which provide point-to-point connection, thus network is capable to connect to each process in the system. Indoor environmental data is collected using low cost (compared to laser sensors) RGB-D camera. The motors of wheels are controlled with help of Arduino microcontroller. The navigation is performed using Pulse Width Modulation (PWM) algorithms, calculating this value with respect to the path planning routine, further the value is sent to the microcontroller. Finally, wheelchair motors controlled according to received PWM, meanwhile the patient

can control wheelchair distantly, see information from depth perception camera, take actions with created indoor map, such as open and save using APP running on Android operating system.

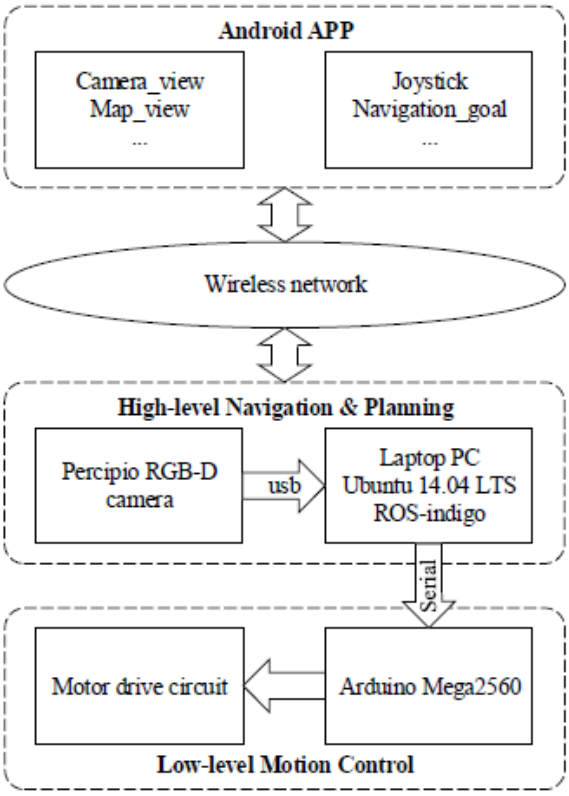


Figure 1 System structure diagram

Figure 1 depicts a system structure diagram of the developed wheelchair. The wheelchair works as follows: to begin a process, the wheelchair gathers environmental information, meanwhile edge-based autonomous navigation technique determines navigation target. During that process, Gmapping algorithm creates an indoor map according to the gathered data from the camera, within movement the map is updated continuously. The map is fully created once autonomous exploration and map building are completed.

Further, A* algorithm [6] uses generated map as an input in order to perform global path planning. DW methods are used to perform obstacle avoidance routine with help of Point Cloud Images. Combined results are sent to ROS, which is generating the shortest path to the target with no obstacles on a way. In this wheelchair's configuration, environmental map is grid-based, the main disadvantage of which is in high resolution. Thus, the cost of finding path is high; therefore, timeliness should be reduced as much as possible. The reason why global path planning is performed with use of A* is because this algorithms has strong timeliness and able to find the shortest path, which is highly needed when grid based map is used.

In order to ensure human-computer interaction, a user interface is performed on ROS that provides a package for creating ROS programs on Android. Rosjava uses standard Java programming language and Java's libraries. Moreover, it is easy to quickly invoke ROS Nodes, Topics and Services. In addition to that, Android_core is implemented inside ROS to help developers to create programs designed for the Android devices. Laboratory tests indicated that the wheelchair is able to create a partial map of the laboratory, and more preciseness can be achieved by expanding the search area. The wheelchair can effectively avoid obstacles and calculate the shortest path to reach a target. In case of a deadlock, the wheelchair runs exception-handling mechanisms and perform rotation in order to find a way to exit from this state. Created HMI is giving an opportunity for a user to navigate using virtual joystick and a user can obtain an indoor map and obstacle scanning information on a screen. To be short, user-friendly interface gives an opportunity to interact with user and provides satisfaction of the basic needs of the user [7].

However, researchers have not treated importance of the wheelchair's functionalities for patient in much detail. For example, the HMI is user-friendly, yet too basic and lacking user experience. Overall functionality is low, does not include semi-autonomous navigation opportunities and does not give a user freedom to adjust interface according to the user's needs. Research on the subject has been mostly restricted to be used for paralyzed people. Although extensive research has been carried out, this model does not have much competitive value on a market, thus cannot be used to set as a commercial product.

2.2 Autonomous Navigation Using Intermediate Targets

The previous section showed how authors tried to solve navigation problems inside uneven and unknown environment. In the section that follows, it will be argued that people suffering from severe diseases mostly spend their time indoor. Therefore, an intelligent wheelchair should be mostly adapted to be used in flat or hospital. A significant analysis and discussion on the subject was presented on International Conference on Advanced and Electric Technologies where an autonomous wheelchair navigation in indoor environment using Fuzzy logic and intermediate targets were presented [8]. The goal of the research was to provide a wheelchair with an autonomous navigation ability inside uneven and unknown indoor environment by exploiting artificial intelligence technique – Fuzzy logic. Authors claim that fuzzy logic is a method that is good to be used in cases when the system require many efforts to be modeled and human expert knowledge is available. This type of controller should be used when developers willing to imitate the human reasoning and simulate human behavior.

It is well known fact that in order to navigate from one room to another it is crucial to take into consideration the position, size of doors and of course, obstacles on its way.

Therefore, the path between initial point and a target is complex and uneven, thus authors propose using intermediate targets in order to facilitate moving to the desired position. This approach preventing the mobile robot from being trapped in front of the obstacle and roving with no direction. For this purpose, researchers analyzed kinematic model of a unicycle type of mobile robot, which is equipped with two rotating and two driving wheels. Note, this model allows to navigate a robot to any direction by orienting wheels of the robot. This configuration can be characterized by the position, *x* and *y* coordination and the orientation θ in a Cartesian space. Researches assuming that there is no slipping during rolling motion, the system performs movement on the horizontal ground and the wheel ground contact is a point, the kinematic model is created as follows:

$$\frac{dX}{dt} = \frac{V_L+V_R}{2}\cos\theta \tag{1}$$

$$\frac{dY}{dt} = \frac{V_L+V_R}{2}\sin\theta \tag{2}$$

$$\frac{d\theta}{dt} = \frac{V_R-V_L}{2} \tag{3}$$

While discrete form of the abovementioned model is (T – is the sampling time):

$$X_{k+1} = X_k + T\frac{V_{Rk}+V_{Lk}}{2}\cos\theta_k \tag{4}$$

$$Y_{k+1} = Y_k + T\frac{V_{Rk}+V_{Lk}}{2}\sin\theta_k \tag{5}$$

$$\theta_{k+1} = \theta_k + T\frac{V_{Rk}-V_{Lk}}{L} \tag{6}$$

Above-mentioned equations are used to simulate a robot model in MATLAB, exploiting fuzzy logic on the discrete form. One of the most important feature to be implemented in order to ensure adequate autonomous navigation is an ability of a robot to sense surrounding environment. Researchers in order to fulfill this requirement mounted appropriate ultrasonic sensors, which are able to detect distance from the wheelchair to the walls, obstacles and the desired position. This sensor is able to provide information, so the distance can be easily calculated. As it can be seen from Figure 2, the proposed configuration is equipped with three ultrasonic sensors in order to calculate the distance, so the robot might obtain the info regarding left, right, front distances, which are inputs for the fuzzy logic controller. Thus, outputs are the speeds of the wheels of the robot – in such a way the wheelchair is avoiding obstacles.

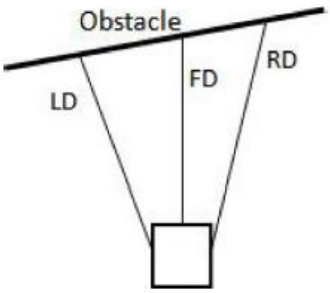


Figure 2 The distances measured by the ultrasonic sensor

In a comprehensive study of intermediate target usage, it was proved that the navigation without intermediate targets is extremely time consuming, the trajectory is not the shortest which is not efficient. However, the implementation of the abovementioned approaches, such as pointing the doors as an intermediate target, brings impressive results. Robot does not move directly to the desired position but chooses the nearest intermediate target in order to achieve desired position. Afterwards, the robot moves toward the chosen target, once reached this target is considered the starting point, the robot once again chooses a new the most relevant intermediate target and algorithm repeats, until the target is reached. Results demonstrated that the robot moves along the shortest path with appropriate execution time. Fuzzy logic ensures robot's autonomy and obstacle avoidance routine.

2.3 Wheelchair controlled by eye-only

So far, this chapter has shown how interaction problems can be solved for people with limited mobility, however, these solutions cannot entirely address problems of paralyzed people. Kohei Aarai proposed a prototype of the electric wheelchair controlled by eye-gaze specifically developed for paralyzed people [9]. Researchers suggest controlling a wheelchair with help of eye gaze and not blinking as it commonly used, because constant blinking might be jerky for a user. System is including NAC Company's camera mounted on glasses of the wheelchair user and designed to be used in the following environment: illumination is less than direct sunlight (around 1400 lux), the surface with no slopes, minimum required rotational space is $2m^2$. Unfortunately, this system is not supported for people with squinting problems and intensive make-up.

This wheelchair will require assistance from a nurse or assistant in order put glasses, switch on power for computer and motors, unlock wheelchair's brakes. Of course, in order to finish a wheelchair's usage the same help would be needed from the assistant, but performed in reversed order. The tested includes: infrared camera, personal computer, a

microcontroller and an adapted wheelchair. Ultrasonic sensor amounted in frontal part is used for collision and obstacle avoidance.

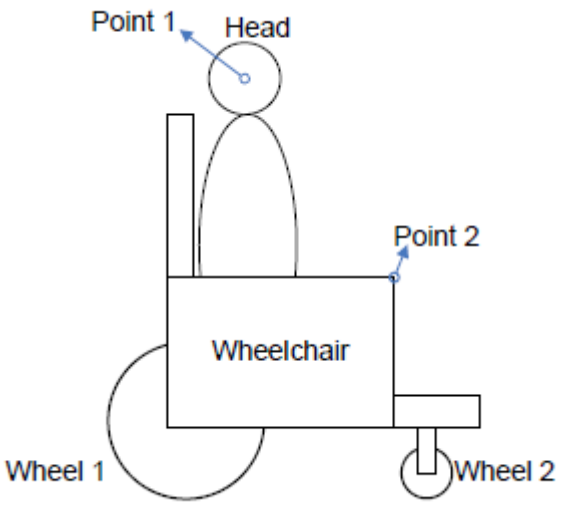


Figure 3 Camera placement comparison. Usually a camera placed at point 2, but in proposed prototype, it is place at point 2

The goal of this project was to create a wheelchair that can be used robustly nevertheless vibration, changes in lightening and type of user (his/her pupil's color, size, etc.). For these purposes LED camera, placed in front of the user (see Figure 3), compensates inaccurate caused by illumination changes and stabilizes the image. Microcontroller connects wheelchair and the PC, converts data from RS 232 serial connection into signal needed to control the wheelchair. Software is developed by using C++ Visual Studio 2005 and OpenCV library for image processing. Using complex image processing algorithms, eye gaze signal is detected and converted into wheelchair commands, which PC sends to the wheelchair. The microcontroller converts serial data to I/O data, which is used to move a relay in order to get analog output, which is linked to the main controller in order to make it possible for the personal computer to control the wheelchair using serial RS 232 serial communication.

There are four key commands on an invisible layout that are used to move the wheelchair (see Figure 4). These keys are intuitively understandable, i.e., when the user looks at right key – the wheelchair will be navigated to the right side until further notice of the user. In case when the user stops looking at the keys or eye gaze of the user inside the free zone, the wheelchair stops its navigation, since it is considered much safer, rather than spending time to hit the stop/off button. In this configuration, backward movement is not supported

since this kind of movement might be considered extremely dangerous for a paralyzed person. The user can switch to hold state by looking upward.

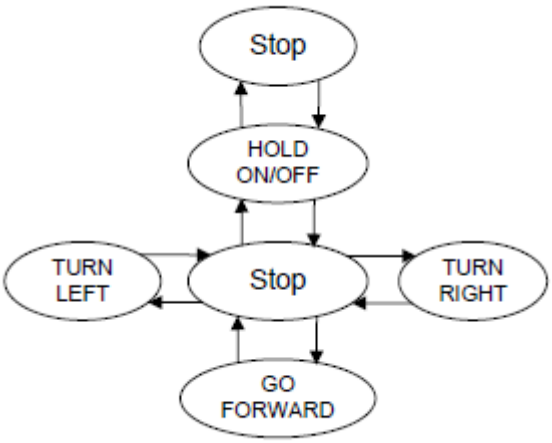


Figure 4 Wheelchair control with user of four key commands on a user interface

This prototypes were tested with help of five participants of different nationality, eye gazed detection is worked adequately. However, the accuracy was lower when the light of the surrounding environment was too bright (see Figure 5). As regards control, users considered it easy to use and control. Unfortunately, manual navigation of the same wheelchair against eye gazed navigation is four time faster. Nevertheless, eye-gazed controlled wheelchair is proved to be adequate and efficient enough to be used under certain circumstances. Thus, developed prototype is a fully realized electric wheelchair controlled by means of eye-gaze.

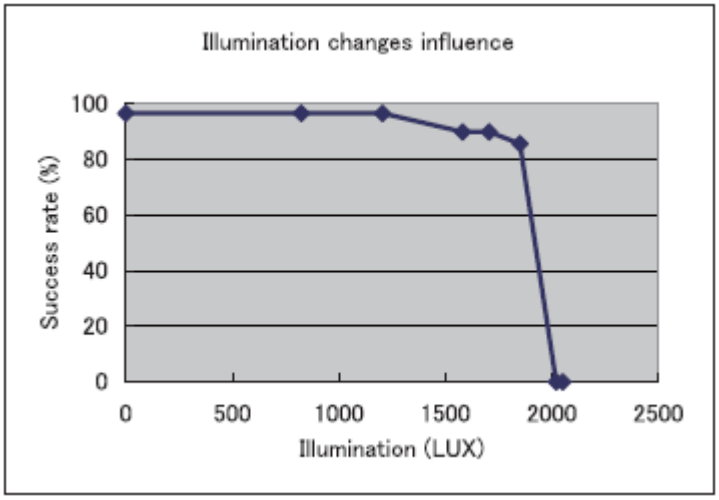


Figure 5 Illumination influence on a success rate

Similar research [10] suggest that eye gaze control might be used as hands free control for wheelchair. It common to exploit three types of a wheelchair control:

- Direct navigation by looking at the target point on the screen;
- Indirect navigation by staring at UI buttons on a screen, so called keys “forward”, “left”, “right” etc.;
- By looking on an image of the front view;

In current research, the last type of interface is used with direct feedback loop; there are no visible UI components.

Figure 6 Invisible UI layout put on top of a video scene image

As it can be seen from the Figure 6, the UI organized in a way that the user can change rotation angle by staring along X-axis, while Y-axis is controlling a speed. The system is updating commands every 100ms, simultaneously adjusting the navigation. If a user will look on an obstacle, the speed will be reduced in a way avoiding an obstacle. The goal of the developed user interface was to ensure hands free control of a mobile robot. During the experiments, it was discovered that the stability of eye-tracking device is a crucial aspect, very good calibrated eye-tracker can provide the same accurate control as a mouse

2.4 Autonomous Mobile Robot Navigation in Indoor Environment

An intelligent wheelchair is a mobile robot, thus it needs to be able to localize itself in the environment in order to perform assigned task with high performance level. To be precise, navigation technologies are concerned about robot’s localization and pose estimation [11]. Data from several studies suggest that localization methods using dead reckoning in order to establish a distance between the initial and given points. This approach is not ideal, since it gathers errors from the each iteration without using external referencing for error correction. Up to now a number of studies suggests usage of external sensors such ultrasonic sensors, cameras, GPS, etc. In addition to that, these external sensors in order

to evaluate robot's position and pose use landmarks. For example, GPS is widely used for abovementioned purposes, which is used mostly for outdoor navigation. However if receivers are not synchronized with the satellites and cannot be precise enough to be used for localization purposes of mobile robots. Sunhong Park and Shuji Hashimoto in their research carried out a number of experiments in this field using exclusive radio-frequency identification. In their work, the robot is able to estimate its position with help of IC tags and trigonometric functions and information of its location recorded dynamically. The usage of IC tags is determined by the low cost and small size, i.e., the tags could be easily installed inside the apartment. The robot in this experiment is based on an electric wheelchair for elderly and disabled people, with mounted distance and touch sensors. The increased amount of IC tags are improving system's accuracy, however the cost is increased accordingly, therefore in this research it is compromised with use of polar Cartesian coordinates of the IC tags, thus these tags are arranged accordingly. In contrast with traditional methods offered algorithm reverberates robot's posture each time it senses a new IC tag. The method works as follows:

- User identifies initial position and desired location to be reached;
- Robot navigates toward the goal by calculated rotational angle between start and finish points;
- Its pose is estimated;
- The angle is recalculated according to a newly reached position;
- The systems reads new IC tags with interval of 0.2 seconds, estimating rotation angles, robot rotates accordingly. Otherwise, it keeps moving forward.

Above mentioned algorithm works in loop until the target is reached. The validation of results showed that the method is precise and outcome the previous studies in similar topics. Despite the fact that the approach require low speed in order to perform accurately, the method is proved to be robust against dirt, wear, various covers and is suitable to be used for elderly and disabled people.

2.5 Commercialization of intelligent wheelchairs

There are relatively few commercial wheelchairs are available on the market, even though wide research was held on wheelchair studies. Often researchers purchase different wheelchairs with aim to conduct a research on it. Nevertheless, the usage of these wheelchairs is limited by a scope of laboratory room, lack of commercialization means that smart wheelchair are not widely spread in hospitals and clinics. This situation is due to the fact that most of the wheelchairs are very advanced and extravagant which makes it very challenging to purchase, since the cost is too high and configuration is complicated. However, there are few commercial wheelchairs available nowadays. One of an important

example is Smile Rehab Limited, which was developed by the University of Edinburg. Compared to the above-mentioned wheelchairs, the model relatively simple and cheap. Sensors are basic-line followers and contact switches. The end-user can add advanced functionality to the wheelchair with help of the serial port with additional expansion protocols. Another example is Boo Ki Scientific prototype, Robotic Chariot, developed for a Pride Mobility Jazzy wheelchair. This wheelchair supports advanced obstacle avoidance routine and path planning. The end-user has an opportunity to customize its navigation functionality.

This chapter has reviewed wheelchairs that were developed in laboratory premises with research purposes. Each model has its own pros and cons as it can be seen from the table below.

Wheelchair	Advantages	Disadvantages
ROS-based Indoor Autonomous Exploration and Navigation Wheelchair	Low cost, human friendly interface, user interaction, consistent with the laboratory results	Hardware might be optimized, not possible to be used by paralyzed people, UI yet too basic, too demanding algorithms
Autonomous wheelchair navigation using Fuzzy Logic controller with help of Intermediate Targets	Effective path planning routine, great execution time, obstacles avoidance	Cost, human machine interaction is poor
A prototype of electric wheelchair controlled by eye-only for paralyzed user	Great and safe user-interface for paralyzed people	Too advanced image-processing algorithm for eye tracking. Currently, can be replaced by the eye-tracking device
Autonomous Mobile Robot Navigation Using Passive RFID in Indoor Environment	Cheap and effective localisation	Lacking User Experience

Table 1 Comparison of all related wheelchairs

Above-mentioned models inspire the author to create a commercial eye-gaze controlled wheelchair with adaptable user interface. In the prototype, author will use Aruco markers placed on point of interests and intermediate targets (doors) in order to ensure cheap localization and effective path planning.

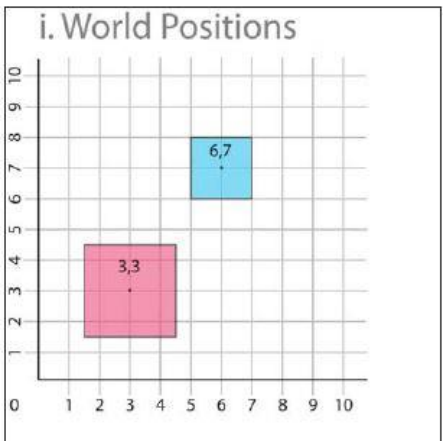
2.6 Environment and common techniques for developing UI

Unity 3D is a game engine developed by Unity Technologies [12] which supports more than 27 platforms. This engine can be used to develop 3D and 2D games as well as many various applications. For this master thesis, it was required to use Unity 3D by the task description since the project implementation needed to use rich graphics and AR features.

Nevertheless, this constraint was not causing additional challenges to the author. This section is focused on a software description and a common software technique for developing HMI.

2.6.1 Programming environment

In order to create high fidelity user interface fulfilling wheelchair functionality and being able to be adaptive, Unity 3D is chosen. It gives a developer a lot of freedom to create powerful user interface with high graphical capability. Since Unity is a 3D development kit, it requires a certain level of understanding 3D space and 3D development. Moreover, it is essential to understand the difference between local space and world space. In any 3D world 3D Cartesian space is used. The Z-axis represents depth in addition to the X for horizontal and Y for vertical axes, in such way can be represented positions, dimensions, rotational values, etc. and should be described as follows (X, Y, Z) due to programming reasons. In every world space, there is a term "origin" which is represented by the position (0, 0, 0). Thus all world positions in are relative to the "origin" or "would zero," to make it more simple it is common to use local spaces and define object's position relative to another, so-called parent-child relationships. In Unity 3D it is easy to form these relationships by drugging and dropping elements one into another in a hierarchy of elements. Thus, the position of a child will be relative to the position of a parent. Using local spaces means that every object has its origin or zero points, the point where it is X, Y, Z axes are merged, which is usually a center of an element. When child-parents relationships are established, it is possible to calculate a distance from other elements by using coordinates of local space. Figure 7 depicts two diagrams, the first diagram (i) shows two objects coordinates with respect to world space, where a big pink cube has coordinates x equal to 3 and y to 3 and a small blue cube with coordinates 6 and 7. In the second diagram (ii), the small cube is a child of the bigger cube. Thus, its coordinates are said to be (3, 4), the same goes for 3D space.



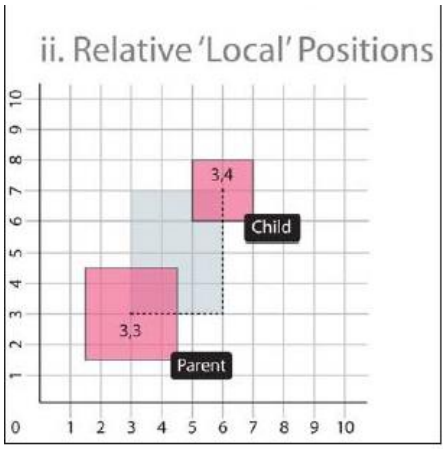


Figure 7 Representation of World Space and Local Space coordination systems

2.6.2 Cameras

Cameras are significant in the 3D world due to the fact that they represent a viewport for the screen. In Unity, cameras can be placed at any point in space, can be attached to any Game Object or Game character. A Scene can have many cameras. However, it is assumed that exists a single main camera that will be rendering what the user sees. Unity creates the Main Camera project whenever a new scene is created. Cameras components are shown in the following figure (see Figure 8):

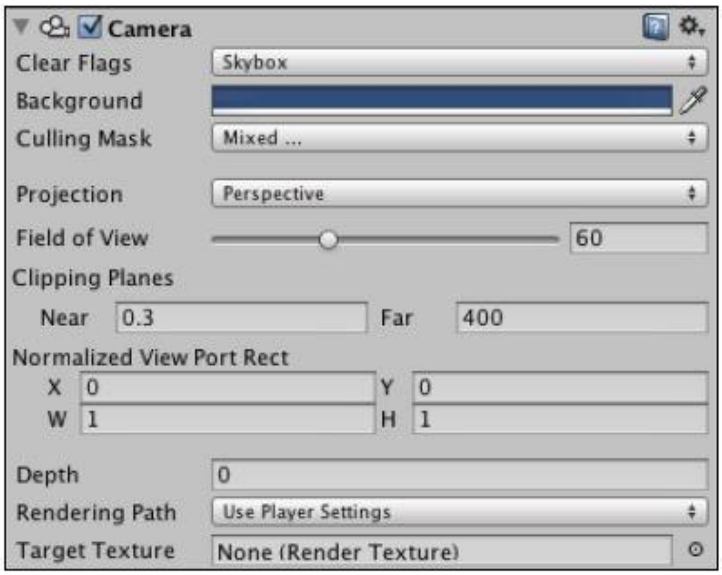


Figure 8 Camera components

Clear flags are set to Skybox by default to allow the camera to render the sky-box material currently applied to the scene. However, to allow the developer, to manage the use of multiple cameras to draw the game world, the Clear Flags parameter exists to allow a developer to set specific cameras, as well as to render specific parts of the game world.

The background color is the color rendered behind all objects in a game, in case if no skybox material is applied to the given scene. Clicking on the clock block will allow a user to change the color using the color picker or ink dropper icon.

The Normalized View Port Rect parameter allows a user to give dimensions and a position for the camera view. The X, Y coordinates are being set to 0; thus the camera view comes in the bottom-left of the screen. Values Width and Height are set to 1, so the view from this camera is filling the screen because these values are set in Unity's coordinates system ranging from 0 to 1.

Clipping Planes (Near/Far), the near plane is the closest distance to start rendering, and the far plane is the furthest start of the drawing.

Field of View, this parameter establishes the width of the camera's viewport in degrees. Currently, it is set to be 60 degrees, which gives the effect of a human vision.

Depth is used in case of multiple cameras in a scene. This parameter establishes an order of priority for cameras views, the camera with higher depth value will be rendered in front of cameras with lower depths. By default camera has Perspective Projection mode, which has a pyramid-shaped Field of View (FOV). The projection mode of a camera depicts whether is rendered in 3D (Perspective) or 2D (Orthographic).

Culling Mask parameter works with Unity's layers allowing deselecting the layer if needed.

2.6.3 Assets

In any Unity project exist Assets folder and mirrored in the Project panel, which contains building blocks such as images, 3D models, sounds effects. Scenes represent an individual level of a game content, some developers using only one Scene for developing a game, however in some cases using two or more scene is essential. It is essential to bear in mind that only one scene can be active at the moment. Scenes are manipulated employing Hierarchy and Scene Views. GameObject is an active object is the currently used scene, which can be placed in a hierarchy, establishing child-parent relationship and always contains at least one component, which is Transform that tells a developer a position, rotation, and scale of the object in X, Y, Z coordinates as is shown in a figure below.

When utilizing fonts in any Unity project it is necessary to import into project as any media file needed for given project. This can be done by uploading media file into assets folder or by following steps, Assets-> Import New Asset.

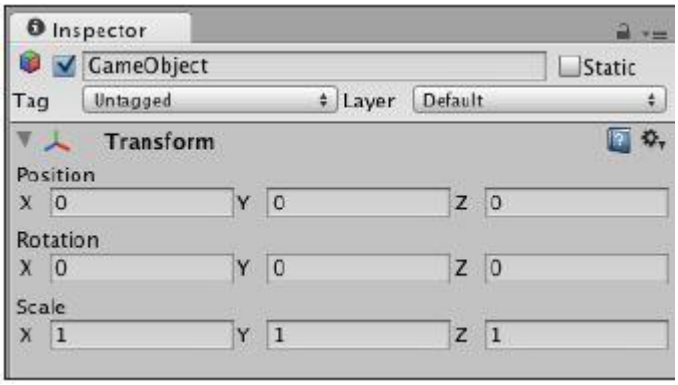


Figure 9 Transform component of a GameObject.

Components and Scripts

There are many different types of components, defining behavior, appearance and other functionality of an object. Exists built-in components such as Rigid body, establishing physics of an object, and some simpler such as cameras, lights, etc. It is possible to extend functional abilities of an object by writing a script and attaching to the object. Scripts can be written in C#, JavaScript or Boo (a derivative language of Python). To write scripts, Unity provides standalone script editor, Monodevelop, a separate application that is launched any time when a script is edited. However, it is possible to designate a script creation and editing to other editors such as Visual Studio. Usually, games operate at a certain number of Frames per Second (FPS), and function Update() is called every frame. Thus, it is mostly used for detecting any changes happening in the game, which happens in a real-time, for example, mouse click or key press.

2.6.4 UI development essentials

Canvas is a component that contains all components of a UI. Thus buttons, panels, images and other elements of the UI are children of the canvas component. When a UI component is instantiated, the canvas – a parent of the component – is created automatically. Components order determines a rendering order is a hierarchy, the first object is rendered first, and the last object is rendered the last and placed over other components.

The canvas component representation depends on a “Render Mode” which has three different options of the Canvas rendering:

1. Screen Space-Overlay. The canvas is rendered over all elements of a Scene. The great analogy for this rendering mode are stickers on a window of a train, where the canvas is a window and stickers are UI elements. The “stickers” remain the same, even if the world behind the window is changing. This rendering mode change size of the Canvas when the size of a screen is changed. Given rendering mode is suggested for a static object such as a panel of instruments, score panel, etc.

2. Screen Space-Camera – This is similar to Screen Space-Overlay, but in this render mode the Canvas is placed a given distance in front of a specified Camera. The UI elements are rendered by this camera, which means that the Camera settings affect the appearance of the UI. If the Camera is set to Perspective, the UI elements will be rendered with perspective, and the Camera Field of View can control the amount of perspective distortion. If the screen is resized, changes resolution, or the camera frustum changes, the Canvas will automatically change size to match as well [12].
3. World Space – In this render mode, the Canvas will behave like any other object in the scene. The size of the Canvas can be set manually using its Rect Transform, and UI elements will render in front of or behind other objects in the scene based on 3D placement. This is useful for UIs that are meant to be a part of the world. This is also known as a "diegetic interface" [12].

Each element displayed as a rectangle. To manipulate UI elements, Rect Tool and Rect Transform components are used. Rect Transform component has following fields – Pos X, PosY, Pos Z. Anchors: Min, Max, Pivot, Rotation X, Y, Z, Scale X, Y, Z.

Anchors determine how the size of an element is changed concerning changing the size of a parent element. Each component has four anchors for each vertex of its rectangle. A position and size are counted based on a distance between a vertex and anchor and a position of an anchor itself. A position of the anchor is determined by a percentage ratio of a size of a parent element. Thus, if all four elements of an anchor are placed in the same place, the size of a component remains the same. In case if two anchors of the same plane (left and right, up and top) are placed in the same point, the element will not be stretched with respect to this plane. However, if anchors are not in the same point, the position of each anchor will be counted in a percentage and to the given value will be added distance to the vertex (which is not changed).

Reason for choosing C# over UnityScript

C# is a well know and widely used programming language developed by Microsoft in order to create Windows application and web-based applications. There are plenty of materials on Internet helping to learn fast how to use his programming language. UnityScript is a programming language similar to JavaScript but still different. Therefore material used to find a solution in JavaScript may not be applicable for UnityScript. In addition to that, it is better to apply already familiar language rather than learning entirely new programming technique from scratch. C# gives developer flexibility to use scripts without attaching them to GameObjects, in addition to that developed State Machine in given Master Thesis is more natural to be done by using C# rather than UnityScript. Since C# is known as a strictly-typed language, thus Unity will catch errors at the moment, thus more comfortable to correct errors. UnityScript shows mistakes only when the developer

is compiling a file, which makes it more challenging to create a valid code. For an overview of a topic, a Reference Manual was used chiefly for Scripting Reference.

Each script in Unity uses inheritance. By default, scripts are inheriting from MonoBehaviour class, which means MonoBehaviour is making few of its variables and methods available for the default script. When C# is used, it must explicitly derive from MonoBehaviour. When a developer uses UnityScript (a type of JavaScript), you do not have to derive from MonoBehaviour explicitly. Note: There is a checkbox for disabling MonoBehaviour on the Unity Editor. It disables functions when unticked. If none of these functions are present in the script, the Editor does not display the checkbox.

The functions are:

- Start()Start is called on the frame when a script is enabled just before any of the Update methods is called the first time.
- Update()Update is called every frame, if the MonoBehaviour is enabled.
- FixedUpdate()This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
- LateUpdate()LateUpdate is called every frame, if the Behaviour is enabled.
- OnGUI() OnGUI is called for rendering and handling GUI events.
- OnDisable() This function is called when the behavior becomes disabled () or inactive.
- OnEnable()This function is called when the object becomes enabled and active.

GameObjects have Components that make them behave in a certain way. Any component of any GameObject is a script, developed by Unity Team or by the user, that defines a class. Which means that the properties we see in Inspector are just variables of some type.

If we use public variables in a script, we can change it from Inspector panel.

2.6.5 Software development. State Machine approach.

The best part of State Machine is simplicity; the great example of a State Machine is an everyday life of a person. For example, when a person sleeps he/she is in a “Sleep State” and during this state it is impossible to do anything else. The idea to remain in a particular state is that a person is allowed to do only what is allowed to do in this kind of state. When we use a State Machine we force a computer or in our case a wheelchair to be in a particular state. It will stay in one state until it is told to change to another state. Benefits of implementing a State Machine are

1. Clean layout of a software control;

2. Very cleans points to add software functionality or feature;

3. It is easy to extend software logic by adding another state;

4. Code is smaller, cleaner and specific for each state.

Developed HMI is not a simple code. Instead, the features are added on the fly and future improvements will require adding new states. Without implementing a State Machine it would be very difficult to edit code in order to follow all changes. In addition to that keeping track of variables, storing data would be a mess. All of this would lead to a difficult to understand and edit code. Therefore, it was decided to implement a state machine code instead of attaching components to GameObjects.

Following diagram demonstrates the basic concept if a State Machine controlling a software:

- Unity calls Update() method every frame;
- The State Manager script works as a manager of the whole mechanism of the State Machine, a component has the Update() method;
- The code block Update is responsible for delegating control to the active state;
- States represent regular C# Scripts, which are not components of Game Objects;
- The active state determines what is happening at this specific moment, therefore it plays a role of a logic controller of the software;
- The active state decided when and which State will be active further.

In s State Manager script this is a Unity class, so it inherits from Mono Behavior class, this script is attached to a GameObject to become a component. State Manager has a three core features:

1. Delegating control to a State;
2. Switching to another State when called to do so;
3. Keeping track of the active State.

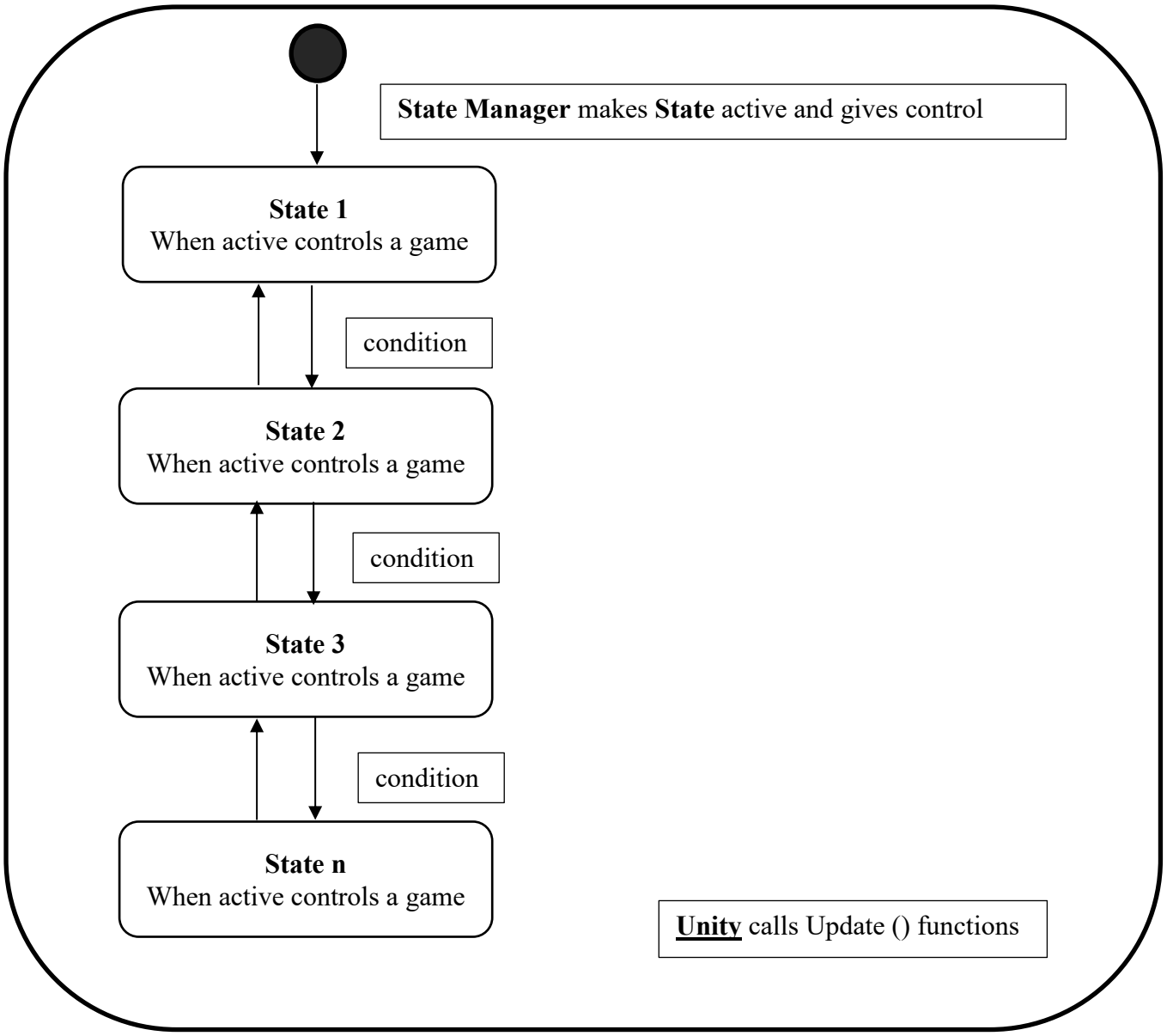


Figure 10 State Machine diagram

2.6.6 Delegating game control to a State

The State Manager script is attached to a GameObject and becomes a Component object. This script uses the Update() method in order to pass the game control to the active state as it shown in the following diagram (Figure 11):

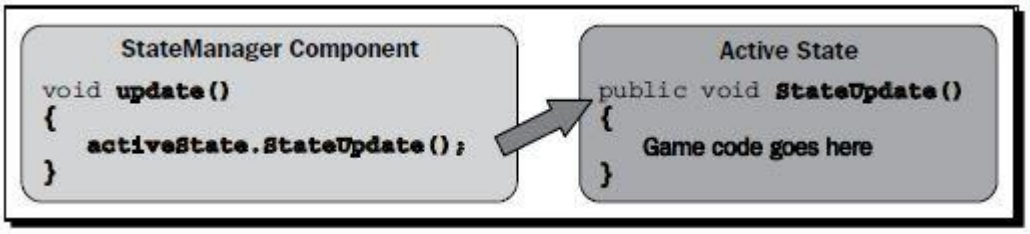


Figure 11 State Manager control delegation

The game control code that is usually is used in an Update() method is instead delegated to the StateUpdate() on the active State object. So every time Update() method is called on the State Manager Component, the StateUpdate() method is called on the currently active State object.

No matter how many states are in the software, it is remember the following points:

- Every State should have the StateUpdate() method;
- However, each logic block will be different for each StateUpdate() method on every State, depending on what you want each State to accomplish;
- Only one state is active at any moment of time.

Switching to another State when called to do so

Each state determine when to switch to another state and at which conditions, this should be coded inside the active state what will be a trigger to switch to another State. This is implemented by SwitchState() method on a State Manager Component.

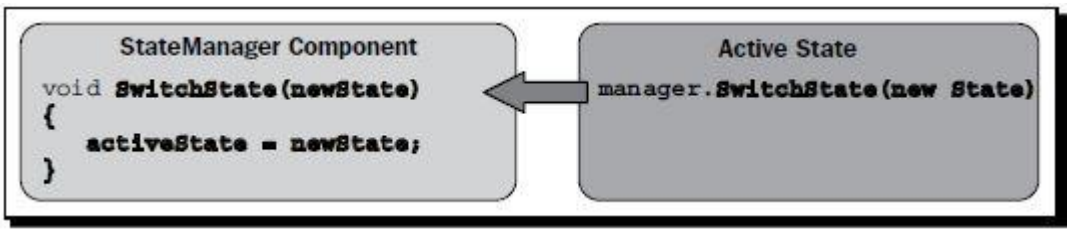


Figure 12 State Machine SwitchState() diagram

When active state is determined, the software switches to another State, so SwichState () method is called with the argument of the following active state.

2.6.7 Keeping track of the active State

Newly created State is then assigned to the active state variable, to keep track of the active state. This is needed to give a State to control the game; State Manager needs to know which state is active now. So when Unity calls Update() on the State Manager script control is passed to the newly created State that is stored in the active state variable by calling its StateUpdate() method. The State Machine cycle of changing to a new State complete.

These steps are as following:

• The active state determined when its time to switch to a new state;

• A new state object is created and passed to the StateManager script using the Switch State() method;

• This new State object is assigned to the active State variable;

• When Update() is called on State Manager, the Update() method delegates control to the StateUpdate() method on the new State;

• So it starts from the beginning.

Unity creates Components behind the scenes

As mentioned previously, a Unity script is just a file that defines a class, thus attaching a script to the GameObject allows Unity to create a component object in a Memory when button “Play” is clicked.

Instantiate means to create an object from a class. A namespace declares where a class file is located in the folder in the Unity Project’s file structure, using namespace means that the code is allowed to use any class file located in this folder.

All states of a state Machine have to perform particular methods that StateManager class can call. In order to make State Machine works correctly, it was created an interface, which guarantees that the required methods are used.

As it can be seen all states are equal from a structural point of view and inherits from IStateBase class, which provides guarantees that States will have the required methods which State Manager needs to access. In addition to that activeState variable of type, IStateBase requires implementing IStateBase.

2.6.8 Changing Scenes destroys the existing GameObjects

When more than two scenes are used appears two main issues:

Every GameObject is destroyed while switching between scenes, while another Scene is loaded. In addition to that, every time the first scene is loaded UI manager GameObject with StateManager is created. For this purposes used DontDestroyOnLoad() for not destroying the GameObject consisting the StateManager script and detecting whether a GameObject has existed thus a new GameObject is not created. StateManager is static means that each instance of the StateManager will share the same value, plus as its private, it cannot be changed outside the StateManager class.

2.6.9 UI control. Eye Tracking.

With technology development, it is crucial for society to have an ability to keep up with the advancement; otherwise, the technology becomes not in use. Since the personal computer became a standardly used device, corporations are trying to create more user-friendly solutions. The primary goal of manufacturers is to create the most favorable devices, for example, personal computers, phones, and tablets, to stay on top of the market and, thus, staying one-step ahead of new trends. A notable example of advancing technology is a touch control system used by Apple Corporation, which after quickly becoming fashion-able expanded to other devices. Voice control illustrates this point clearly as being used on computers for people with limited abilities to communicate. Thus, manufacturers are looking for a new disruptive control technology that will be used for commercial purposes for different business units. Eye control is a technology that allows computers to detect at which point a user is looking. Eye-tracking devices, which recognize and track eyes' movements. The term gaze-tracking software refers to software that calculates the eye gaze from the features in a process called gaze estimation. Four functions are defined to be key functions for most eye tracking devices: connection, calibration, synchronization, and data streaming [13]. Connection has been broadened to include establishment with eye tracking device, then calibration has been used to define the user's eyes location and synchronization the computer's display with the eye movement by displaying calibration points. Another example of what is meant by calibration is a dot moving around the screen and the user's eye following it to let the eye-tracking device to know the particular eye moves. Synchronization can be loosely described as data streaming which is sent to the software of the eye tracker's device, thus allowing the user to see what is happening at this particular moment.

An eye tracker is a helpful device for many different fields. "We may presume that if we can track someone's eye movements, we can follow along with the path of attention deployed by the observer" [14] which give clear understanding what the user perceived and interpreted whatever he or she saw. It is a well-known fact that eye-tracking might be a helpful tool for people with disabilities that limit movements and voice usage; thus keyboard cannot be used.

One of the leaders on the current market is Swedish company – Tobii, which was founded in 2001 and currently has more than 1000 employees in six different countries. Tobii is trying to provide this technology in many industries – medicine, gaming, diagnostics, software development. [14]. Andrew Duchowski in his book "Eye-Tracking Methodology: Theory and Practice" explain Tobii, a camera and infrared LED optics are embedded under LCS display, which is looking like a flat panel display. The camera and infrared light are required for the eye-tracking technology to detect the user's eye position and movement. The hardware of Tobii is designed for high-performance sensors; it consists of specially designed projectors, customizable image sensors, and optics, as well as custom processing with built-in algorithms. Tobii uses sophisticated algorithms to interpret

images; these algorithms are considered to be a “brain” of the system. Moreover, the technology has an intelligent application layer to allow the many chooses of the technology usage. The figure below provides an overview of Tobii system [14].

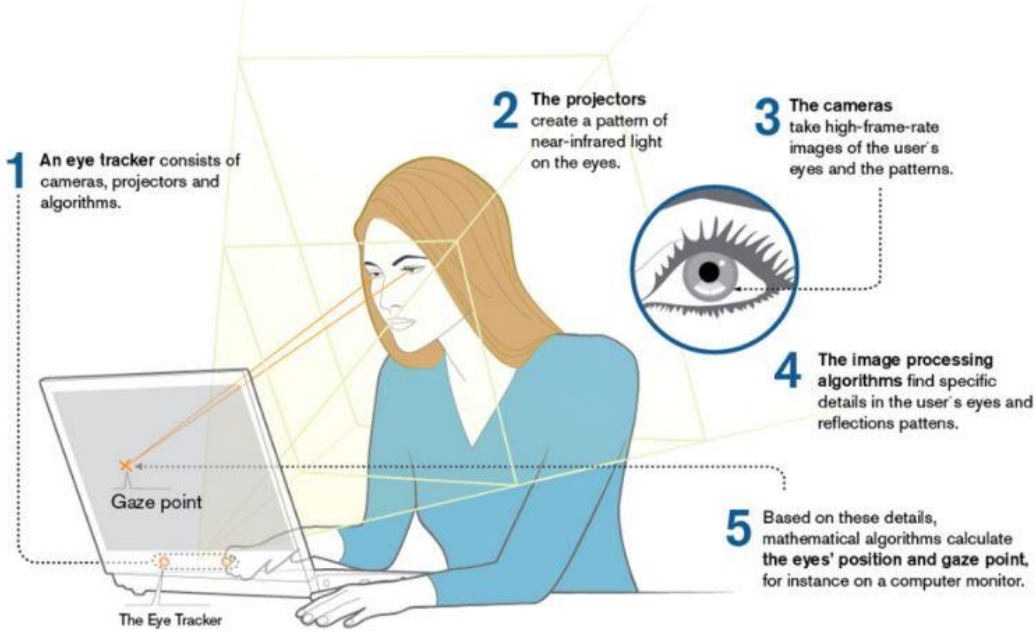


Figure 13 Five steps of a high-performing Tobii eye-tracking system

Since eye-tracking technology provides a measurement of the gazed point on a monitor, this technology continually improving. Moreover, nowadays is it considered suitable to be applied for controlling HMI, which is a primary advantage over standard control means, because people with severe disabilities broaden the end-users of this interface.

3. THEORETICAL PROPOSAL

Based on literature review made in chapter 2, the author made decision to implement a user interface controlled by eye-gaze using eye tracker. The software will be developed on Unity 3D following State Machine approach, so it can include the previous work done prior to the thesis work. Mapping of the environment implemented with use of CAD drawings, forming graph-based map. At the same time, localization is performed with help of Kinect camera acquisition and ArUco markers. These markers are placed on each point of interest inside environment and form each node of a graph. Path planning is implemented with use of Dijkstra algorithm. Above-mentioned solution are presented in more details in following chapter.

3.1 User experience for Eye-Tracking based HMI

Free software, downloaded from [12] was used for implementation of the human-machine interface. A standard free version of Unity gives an opportunity to build developed software for:

- PC and Mac standalone;
- Mac OSX Dashboard widget;
- Web Player.

The version for Professional or Pro version of Unity can be used to develop all of the above and for mobile devices and consoles. Pro Mac Standalone building will create a folder containing an exe file and the associated assets required to run the game in folders alongside it.

Unity supports two modes to create a GUI and UI. In case of GUI the interface is created by scripting and using function OnGUI() and UI approach, when it is needed to drag and drop UI elements such as Buttons, Images, Toggles etc. Both methods are good, however according to a small research on Unity users' forums it was decided to use UI since it is more visible and gives more freedom to design and ensure full graphical support.

Given interface is able to adapt to any given resolution. Is tested to be adaptive to standard resolution such as Standalone (1024x768), 16:9, 16:10, 3:2, 4:3, 5:4. This was achieved by using the Canvas's Render Mode – "Screen Space Overlay". As it was mentioned above, there are three available Canvas's Render Modes, which are Screen Space Overlay, Screen Space – Camera, World Space. In case of Screen Space Overlay mode the UI elements are placed on the screen rendered on top of the scene. Thus is the resolution of the screen is changed or the screen is resized the Canvas is automatically change its size in order to match this [12].

In addition to that, it is an important step to add a component to the Canvas a “Canvas Scaler”. Canvas Scaler aims to control the overall scale and pixel density of UI elements in the Canvas, so the scaling effects all children of Canvas (Figure 14). This is important to choose UI Scale Mode as Scale with the Screen Size out of three possible options, which are Constant Pixel Size, Constant Physical Size and Constant Pixel Size. Screen Match Mode is Match Width or Height.

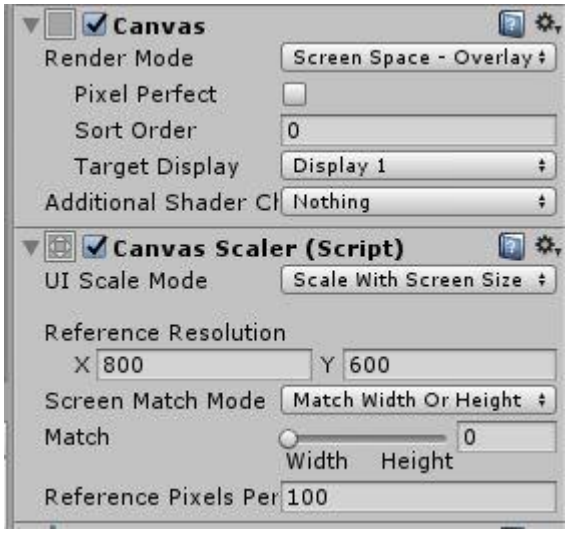


Figure 14 Canvas components and it's settings

In order to obtain correctly Rect transform of the button it is important to take into account the coordination system of the Rect Transform, position of the buttons and position of the Mouse Pointer. In order to use *x,y* position of the eye tracker it is important translate the position of the rect of the button with respect to the world coordinate system. By obtaining Rect transform of the buttons, the HMI is ready to be used for every and each tracking system, since this approach require only position of eye-gaze without using specific libraries such as Tobii Pro. Main thing to be taken into account while creating user interface using eye-tracker, is critical to choose correct position of buttons in a way that user use it efficiently. Buttons should not overlap between scenes or be located too close to each other. In addition to that, it was decided to locate button on a bottom panel of the screen, according to the questionnaire which has been help by the team in prior to the user interface creation.

User interface should follow state machine approach as it was stated in chapter 2. States are following: Idle, Manual, Semi-autonomous, Map autonomous navigation, Path-planning. By pressing buttons user can switch between states ensuring smooth usage of the software. The software is fully performed using C# script.

3.2 Path planning

3.2.1 Localization of the wheelchair inside domestic environment

The localization of the wheelchair is performed by fusing the incremental recursion from encoders (see Eq.1) [15] and a custom designed absolute localization method on a data received from ToF.

$$\begin{cases} x_{k+1} = x_k + \pi \times \frac{n_{Rk} \times R_R + n_{Lk} \times R_L}{n_o} \times \cos \delta_k \\ y_{k+1} = y_k + \pi \times \frac{n_{Rk} \times R_R + n_{Lk} \times R_L}{n_o} \times \sin \delta_k, \\ \delta_{k+1} = \delta_k + 2\pi \times \frac{n_{Rk} \times R_R - n_{Lk} \times R_L}{n_o \times b} \end{cases} \quad (7)$$

where x_k, y_k, δ_k are estimated position and attitude at k of the odometric update recurcion loop n_{Rk} and n_{Lk} are the number of counts from the right and left encoders respectively between two subsequent step, n_o is the number of counts of the encoder, R_R and R_L are the wheel radius and b is the wheel base [15]. It is a well-known fact that odometric localization has incremental nature; therefore, it is crucial to evaluate uncertainty propagation. As for the absolute localization, this was designed differently from the canonical ones, which are usually based on the matching of range data with a preventively created map. A vision based solution that foresees the use of augmented reality (AR) functionalities, from [16] enables the robust localization of the wheelchair without the use of a map. Main advantages or this approach can be listed as follows: simplification of data structure, lower computation cost, possibility to used cheap sensors.

RoboEye is created to be used inside the house of a user; therefore, the environment of the home is structured. In this case, it can be assumed that furniture represents reliable landmarks, since then they are not moved very often. AR tags reference doors, tables, sofas and different targets that might be a target point for the user of the wheelchair. Possible target pose [*x,y,θ*] is defined for each reference point of interest. This can be illustrated briefly by some aspect of interest such as TV. This point of interest is defined concerning the tag referenced to this device, thus using "semi-autonomous" navigation user can choose TV as a target and move to this POI.

Algorithm 1 Localization algorithm

---

while *KeepRunning* **do**
    If *CameraConnected* **then**
    Acquire RGB and depth frames
    Calculate plane equation relative to the floor
    Determine camera pose w.r.t te wheelchair
    Transform cloud point into the wheelchair reference system
    Search ArUco marker in RGB image
        for *EachMarkerDetected* do

Determine pose of the POI related to each market
end for
Send POIs detected information to the HMI
end if
end while

Algorithm 1 provides the detailed steps of the localization process of the wheelchair. These steps as follows:

• Data acquisition: ToF collects information regarding color and depth stream data. Later this data is passed to processing block;

• Assessment of Kinect position: this algorithm determines the height and attitude transformation between sensors and ground, employing a RANSAC (Random Sample Consensus) plane fitting. This crucial algorithm compensates mobility of a camera attached to a frame;

• Roto-translation of the 3D points: the depth frames (3D cloud points) are transformed from the Kinect reference system (raw data) to the wheelchair one. This step enables the organization of a more versatile and efficient AR framework;

• Target detection: an ArUco libraries help to the system to analyze RGB frames needed for marker search. If the system detects markers, it evaluates 3D point concerning the tags. This strategy does not require specific knowledge of the intrinsic parameters of the camera. Later data about localized markers and relative POIs are transferred to the HMI as an optional target and anchor for autonomous navigation.

3.2.2 Dijkstra algorithm

One of the most important problem solved in graph theory is the short path finding, which is divided into:

- Finding shortest path between two nodes;
- Finding shortest paths starting from a particular node;
- Finding shortest paths leading to a particular end-point;
- Finding all the shortest paths.

In general, the single source shortest path problems are solved not only graph-based algorithms, but also dynamic programming, neural networks, hybrid and improved algorithms [17]. Currently, the most widely used path planning algorithms are Dijkstra algorithm, A* algorithm, Johnson algorithm, Floyd-Warshall algorithm, Bellman-Ford algorithm [18]. However, Dijkstra approach was proposed by Edsger Dijkstra in 1959 and remains the best-known algorithm in theory [19]. In addition to that Zhan and Noon have utilized 15 various types of algorithms in Cherkassky 17 specie research on an actual

transport work. The wide research showed that Dijkstra algorithm is more suitable for finding the shortest path between two nodes [20]. Thus, this approach was selected for its reliability and validity.

Dijkstra algorithm finds the shortest path in a graph by calculating the length of paths. It solves the short path problem from one node to another in a graph-based map with a single source. First, at all, the algorithm calculate the path from the initial node to its adjacent nodes. Then the conclusive of the shortest oath is considered as an intermediate node, and then Dijkstra approach is searching for the shortest path from the intermediate node to its neighbor nodes. Once every node is crossed over, the algorithm is completed, thus the shortest path is found. It is obvious that in such a way all found sub-paths are shortest paths between any node in the system until the target node.

This algorithm is considered as a labeling algorithm. Let's consider a graph *G = (V, E)* with *n* arcs and *e* arcs. *V* is a set of nodes, while *E* is a set of arcs. *C (A,B)* represents the weight of arc between node *A* and *B*. The weight of non-existent arc is considered as an infinity value (in current master thesis is taken as a large integer value). Array *DIST (X)* denotes a distance to be traveled from the source node v to the target node *x*. *A* list *S* is indicating which nodes where included in the algorithm, initially its contains only the initial point of the path. *V* denotes a list of new nodes, which are not yet in use. The algorithm works as follows [21]:

- Initialization: The initial node is identifies and marked as v, the set *S = S U {v}*;
- Between the *V-S*, adjacent node i with respect to the node v is found, is the weight of the corresponding arcs is the minimum the node *i* is added to the set *S*;
- Node *i* is used as an intermediate target, repeat previous step to find a new adjacent node *j* from *V-S*. The distance between the initial node *v* to the node *j* is changed, according to the following: if *DIST(j)>DIST(i)+C(i,j)* then *DIST(j)=DIST(i)+C(i,j)*. Add node *j* to the list *S*.
- Two previous steps are repeated *(n-1)* times. The shortest path is obtained as a sequence of nodes, initial node, intermediate nodes and the target node.

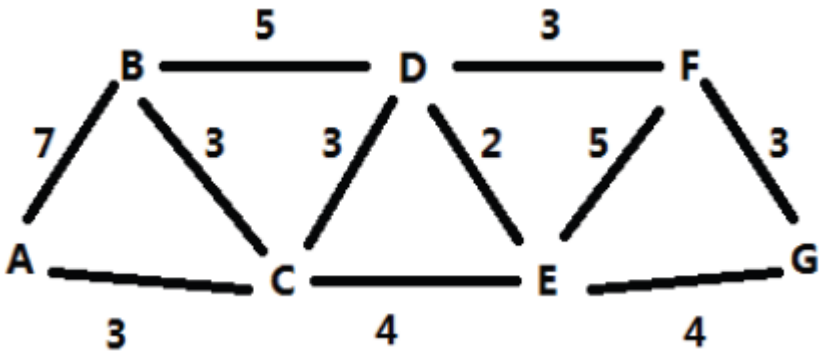


Figure 15 Example of the Dijkstra algorithm

Figure 15 depicts a random graph with nodes from *A* to *G* and some weights of arcs. According to Dijkstra algorithm the shortest path between the source node *A* to the target node *G* is as follows: *A-C-E-G*. As it was mentioned above, the sub-paths are obtained as well. Thus, the shortest path from *C* to *G* is *C-E-G*.

In our case, graph nodes represents rooms and point of interests. As initial node the application detects a node which is the nearest to any marker or point of interest. A goal node is chosen by the user with means of the user interface. After that, an application marks all nodes as "new" of unvisited. For the each graph node *V* in the graph, we set a distance from this initial node *Vo* to this node *V* as infinity. While exists any node which is not visited, the application will be running. For each node *U* with the least distance, application compares this distance of each neighbor's node V with a sum of distance of node *U* plus length of link between nodes *V* and *U*. If this distance is greater, then it is replaced with the calculated sum. The function works until it finds the target node. The block diagram (see Appendix Figure 44) depicts how algorithm works for this particular case.

3.2.3 Environmental Map of Point of Interests and Graph-Based Adaptation

Autonomous navigation is a difficult task which includes mapping, localization and path planning. Most are of research deals with uneven and unknown environment. Since RoboEye prototype is designed to be used inside domestic environment, exist a map of a point of interests. These points of interest are not changed during the whole period of usage, hence we focus on a navigation inside known environment. For mapping we are using ready CAD description of a flat, in this particular case map of the laboratory in the university of Trento. The CAD diagram serves as a global map of the environment, even if it is not accurate enough it makes the process of mapping easier.

The information from CAD is used as follows:

- Rooms are defined;
- Doors are defined;
- Point of interest are marked;
- Representing this information in a form suitable for planning.

For about mentioned reasons to the each point of interest is assigned Aruco marker, which helps the system to detect relative position of these points.

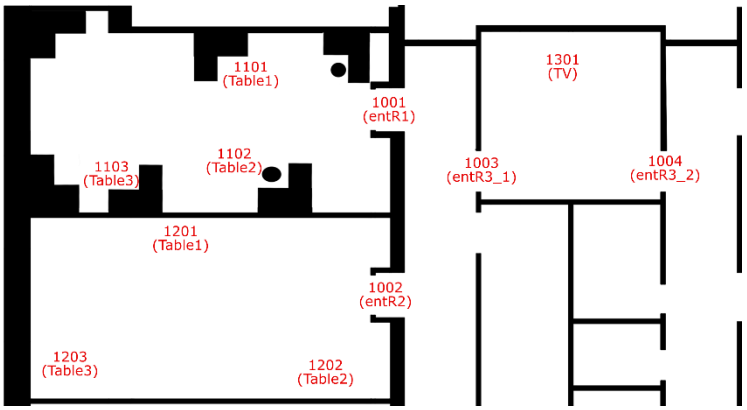


Figure 16 CAD map of the environment

First we have proceed the CAD drawing to extract number of rooms which can be dealt independently, giving a user freedom to choose particular room. Then doors and point of interest are determined, further they are formed into topological map of the environment where doors and POIs are nodes of the graph and distance between them are edges of the graph. Manually we create a file of point of interests, where each point of interest has related marker. Each marker gives to a robot information what kind of POI is in front of it. From the map, the robot can understand to which room is it related, passing point and the distance between nodes.

A map is formed as a JSON file, which is used for the graph initialization and has a structure as follows:

Later the software is able to extract the graph from a JSON file. On a Figure 18 Graph class and Graph Nodes) below show created class Graph and GraphModes, where Graph represent a list of a list of Nodes.

MAP
+ ID of POI + Related marker + X, Y, θ of the POI + Reachable POI + Distance to each POI

Figure 17 Representation of a JSON map of the environment

The map consist all necessary information to create a graph-based map (see Figure 17).

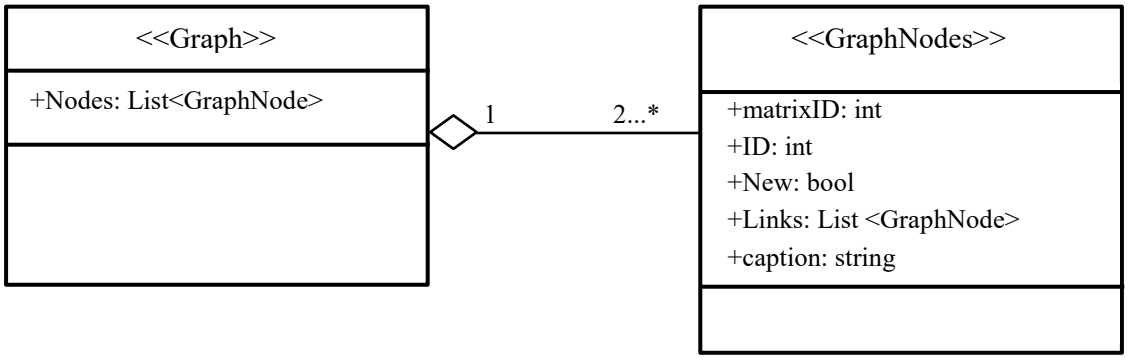


Figure 18 Graph class and Graph Nodes

In addition to that JSON wrapper class in created. Which contains extracted information from the JSON map.

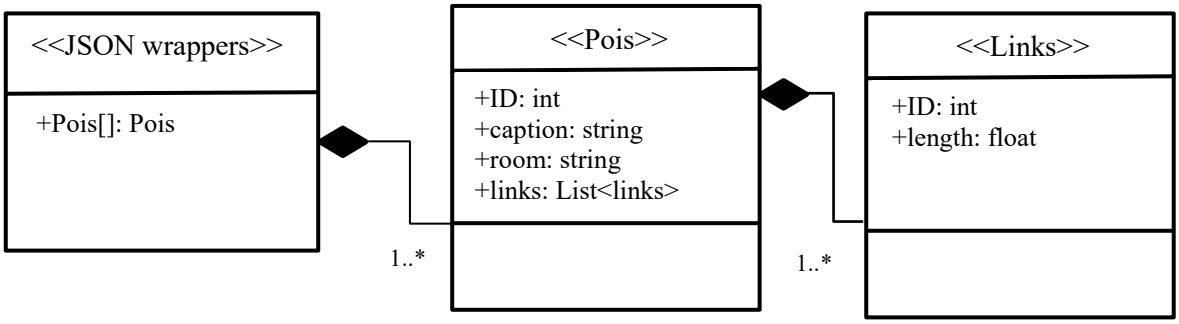


Figure 19 UML diagram point of interest

The function below shows how the data is read from the file:

```
void ReadData()

    {dataIsRead = true;

      try
```

```
{myGraph.Nodes = new List<GraphNode>();
  matrixNumber = 0;
  if (System.IO.File.Exists(pathJSON))
  {string contents = System.IO.File.ReadAllText(pathJSON);
    JsonWrapper wrapper = JsonUtility.FromJson<JsonWrapper>(contents);
    poisList = wrapper.Pois;
    // Debug.Log("fileisread");
    foreach (Pois po in poisList)
    {GraphNode newGraphNode = new GraphNode();
      newGraphNode.matrixID = matrixNumber;
      newGraphNode.ID = po.ID;
      newGraphNode.caption = po.caption;
      newGraphNode.New = true;
      newGraphNode.Links = new List<GraphNode>();
      foreach (links li in po.links)
      {GraphNode newGraphNode2 = new GraphNode();
        newGraphNode2.ID = li.ID;
        newGraphNode.Links.Add(newGraphNode2);
      }myGraph.Nodes.Add(newGraphNode);
      matrixNumber++;}}
  else{Debug.Log("Unable to read the file, file does not exist");}}
catch (System.Exception ex){Debug.Log(ex.Message);}}
```

Program 1. *ReadData function*

The figure below illustrates the block-diagram for above mentioned program code.

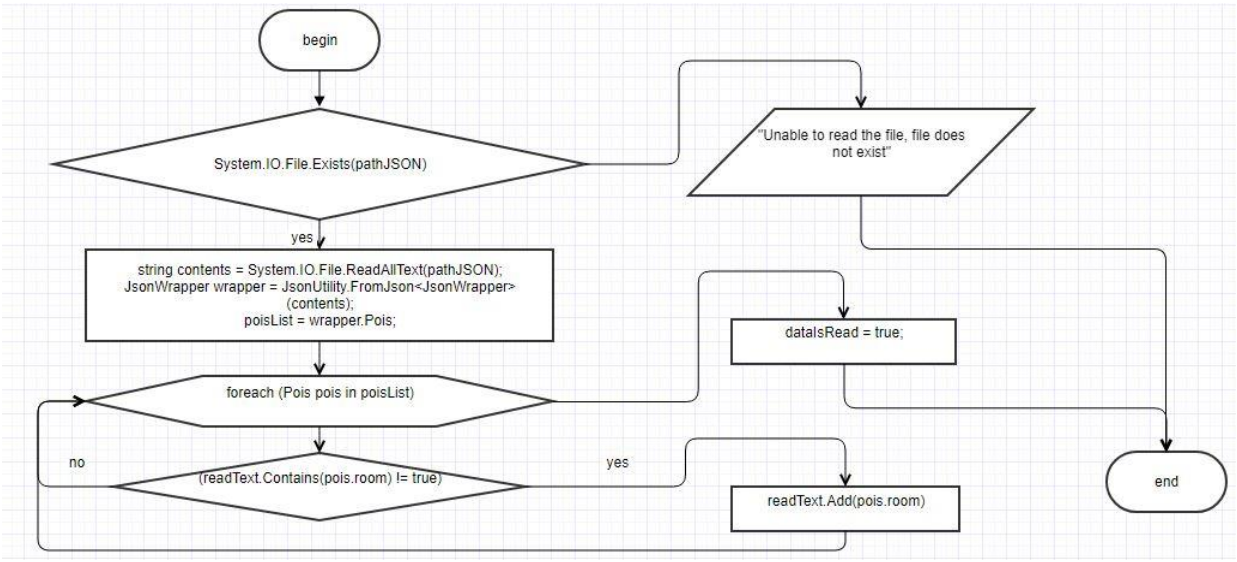


Figure 20 ReadData() function's block diagram

4. PROOF OF IMPLEMENTATION OF THE DEVELOPED SOLUTION

The theoretical approach discussed in chapter 3 implemented on a testbed provided by the laboratory of the University of Trento. The research team is responsible for hardware building and creating services to maintain smooth usage of the wheelchair. Developed solution is not only extends functionally of the wheelchair but also fit into existing ones. The code used before was so called “hardcoded”, thus adding new lines of code required significant changes. The developed software is changed from a scratch following State Machine method in order to include previous functionalities and make possible to add code for future improvements. The following chapter explains results and achievements of the project.

4.1 Testbed description

4.1.1 Architecture overview

RoboEYE is a modified version of an advanced wheelchair GR558 of Nuova Blandino (Figure 21). For this project electronics enabling control by joystick are removed and added hardware specified for eye tracking interface. Moreover, to support advanced robotic technologies needed for semi-autonomous and autonomous navigation. Odometric localization is enabled by two encoders which mounted on the wheels. Commercial driver ensuring required power supports the standard motor. Couple centimeters over a position of legs of a user is a place where a time of flight (ToF) Microsoft Kinect V2 camera is mounted. This position is identified by a questionnaire which was held between a group of wheelchair users, including standard and specific power wheelchairs. Respondents were asked to identify the best position of the wheelchair by comparing images on a screen projected from the camera. The majority of participants indicated that it is important to see their knees on display. Some of those interviewed suggested that this position of a camera is classified as the most useful and productive in-depth perception of the indoor environment, especially when a wheelchair moves close to obstacles or narrow passages. It worth mentioning that the camera is attached to a frame through a revolute joint, thus the position is adjustable to each person's preferences. Images transmitted from ToF is displayed on a screen mounted in front of the user. The system is controlled by interacting with the eye-tracking device, which is attached below the monitor and a screen which serves as an output device for the Human-Machine Interface. The personal computer manages a logic of the system, which collecting data about a position from encoders and control parameters required to lead drivers of the wheelchair, moreover, supervise the eye-tracking device, monitor, and ToF.

Figure 21 RoboEye prototype

4.1.2 Manual navigation

In this modality user navigate wheelchair and choose desired velocity directly looking at the reserved areas of the screen. In this case, the wheelchair control is easy to understand since it is intuitive. However, this is causing significant stress to the user, since this navigation type require constant attention and eye movement. Even though eye movement is considered the fastest, it is main role that is exploration, rather than control. The interface foresees the continuous deviation of frontal and angular velocity in order to prevent jerks, improve smoothness of a movement, which ensure comfortable user experience to a user. User can change speed values by moving eye gaze from bottom to the top of a screen, from minimum value to the maximum. Speed values are calibrated with respect to the eye-tracking device's uncertainty. For both control functions a "rest" zone is a place on a screen where both speeds are equal to zero, this zone is reserved for UI buttons. Worth mentioning maximum speed can be adjusted with assigned button in a settings panel. Figure below illustrates velocity functions.

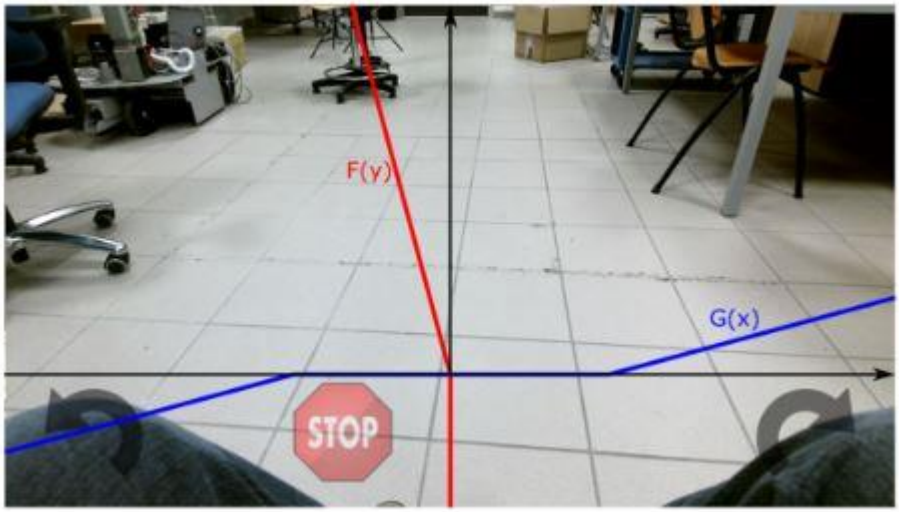


Figure 22 Control law of angular and frontal velocities

Following equation calculates control value of forward velocity, considering y_P as the actual position of eye gaze on a monitor; H is a height of the screen and y_{NAZ} (y normalized at zero is set to be *0.25*), which characterized by the part of the screen where the speed is set to be zero.

$$F_{(y)} = \frac{1}{1-y_{NAZ}} \times \frac{y_P}{H} - \frac{y_{NAZ}}{1-y_{NAZ}}. \quad (8)$$

Forward speed following the law depicted on the equation below, where maximum speed is multiplied by F(y):

$$Ffrw\ Speed = \begin{cases} VfrwM \times F(y), if\ F(y) > 0 \\ 0, if\ F(y) \leq 0 \end{cases}. \quad (9)$$

The lateral speed is calculated analogically to forward speed. Control laws for both right and left movement is depicted below:

$$GR(x) = -\left(xP - \frac{W}{2}\right) \times \frac{2}{W} \frac{1}{(1-X_{NAZ})} - \frac{X_{NAZ}}{1-X_{NAZ}}, \quad (10)$$

$$GL(x) = -\left(xP - \frac{W}{2}\right) \times \frac{2}{W} \frac{1}{(1-X_{NAZ})} + \frac{X_{NAZ}}{1-X_{NAZ}}, \quad (11)$$

where X_{NAZ} is x normalized at zero, is set to *0.15*, xP as the actual position of the eye on the screen, W is a width of the screen.

Later speed is determined following this approach:

$$Lat\ Speed\ R = \begin{cases} V\ lat\ M \times Gr(x), if\ GR(x) > 0 \\ 0, if\ Gr(x) \leq 0 \end{cases}, \quad (12)$$

$$Lat\ Speed\ L=\begin{cases} V\ lat\ M\ \times Gl(x), if\ GL(x) < 0 \\ 0, if\ GL(x) \geq 0 \end{cases}, \quad (13)$$

where $V\ lat\ M$ is the maximum values of the lateral speed.

4.1.3 Semi-autonomous navigation

Semi-autonomous navigation modality was developed to reduce user's fatigue. This technology includes techniques from industrial mobile robotics which give the wheelchair the capability to investigate surrounding environment, to detect a point of interest (POIs). Moreover, the user can choose the POI from the offered variation and the wheelchair capable of moving "semi-autonomously" to this point. The system being in semi-autonomous state searches for potential POIs in the surrounding and computes the most efficient path to reach a chosen position. Human-machine interface illustrates to the user detected POIs and calculated paths if feasible. Besides, it is essential for a user to select desirable POI by looking continuously at it for some particular time to activate above-mentioned point. In the case of activation of POI, the wheelchair initiates independent movement toward the target. During this movement, the user has an opportunity to cancel the navigation.

Since the "semi-autonomous" navigation requires implementation of advanced algorithms, including image and 3D data processing, which involve specific libraries usage, thus cannot be directly realized at the human-machine interface level. For these purposes serves an operative core of RoboEye or C++ DLL. As it can be seen from the figure below (Figure 23), the DLL has three levels and two parallel tasks. Two threads perform absolute localization of the wheelchair and communication system with drives. In addition to that, DLL wrapper connects C++ level with the C# level of Unity 3D software.

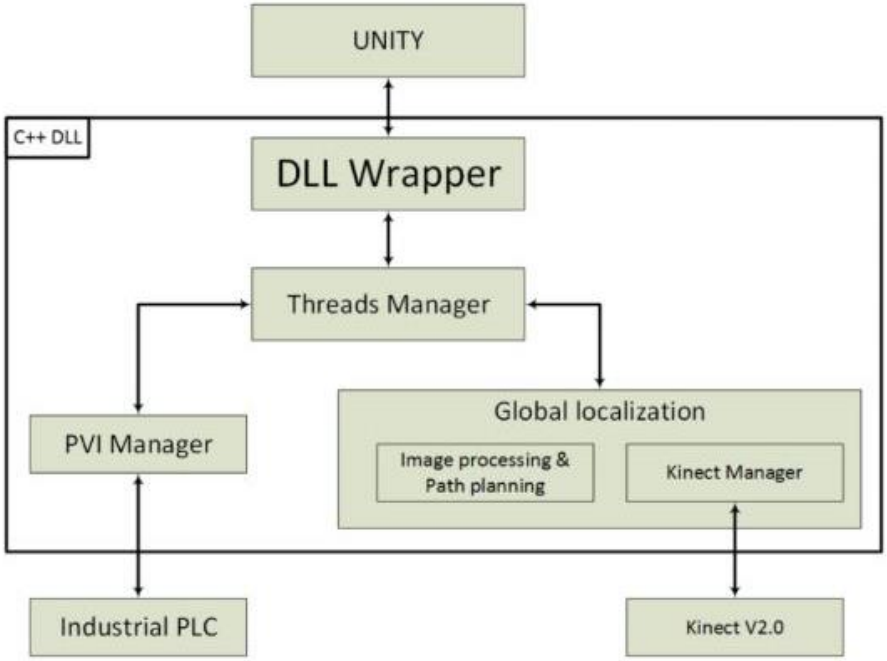


Figure 23 DLL structure [2]

4.1.4 Software overview

RoboEye runs on a Windows Operating System (OS), which manages hardware and maintain a PID controller. That detect the variance between target and real set point velocities, calculating a mistake, which is corrected afterwards to keep deviation minimal. RS323 serial communication is establish to ensure smooth network between high level software interface to low level data (such as variables, modules and functions controlling the wheelchair's movement).

As it can be seen from a Figure 45 (Appendix), software includes the following services. Service 3 (Serial communication) is responsible for manual drive and autonomous commands. Service 1 is responsible for detecting points of interests in surrounding environment, establishing current location of the wheelchair and performs path planning for semi-autonomous navigation, which is sent to the interface. Human machine interface sends data concerning chosen path to this Service. Afterwards the path is used by Service 2, in order to perform path following and odometric localization.

Service 4 performs data collection from TOF camera Kinect or Real Sense cameras, depending on the wheelchair's configuration, and then this data is used by Services 1 and 5. Service 6 manages tilt of sitting, backrest and footboards and any other possible wheelchair setting, which can be controlled by user from HMI, thus status, is displayed on the

monitor. Service 9 and the rest services are working following Master -Slave relationships, thus performs acknowledgement from all services and publish statuses to all services.

4.2 Human machine interface implementation

Software is performed following state machine approach, block diagrams are presented in the Appendix A. Application switching between following scenes: scene idle, autonomous navigation, manual navigation, autonomous map, path planning, settings.

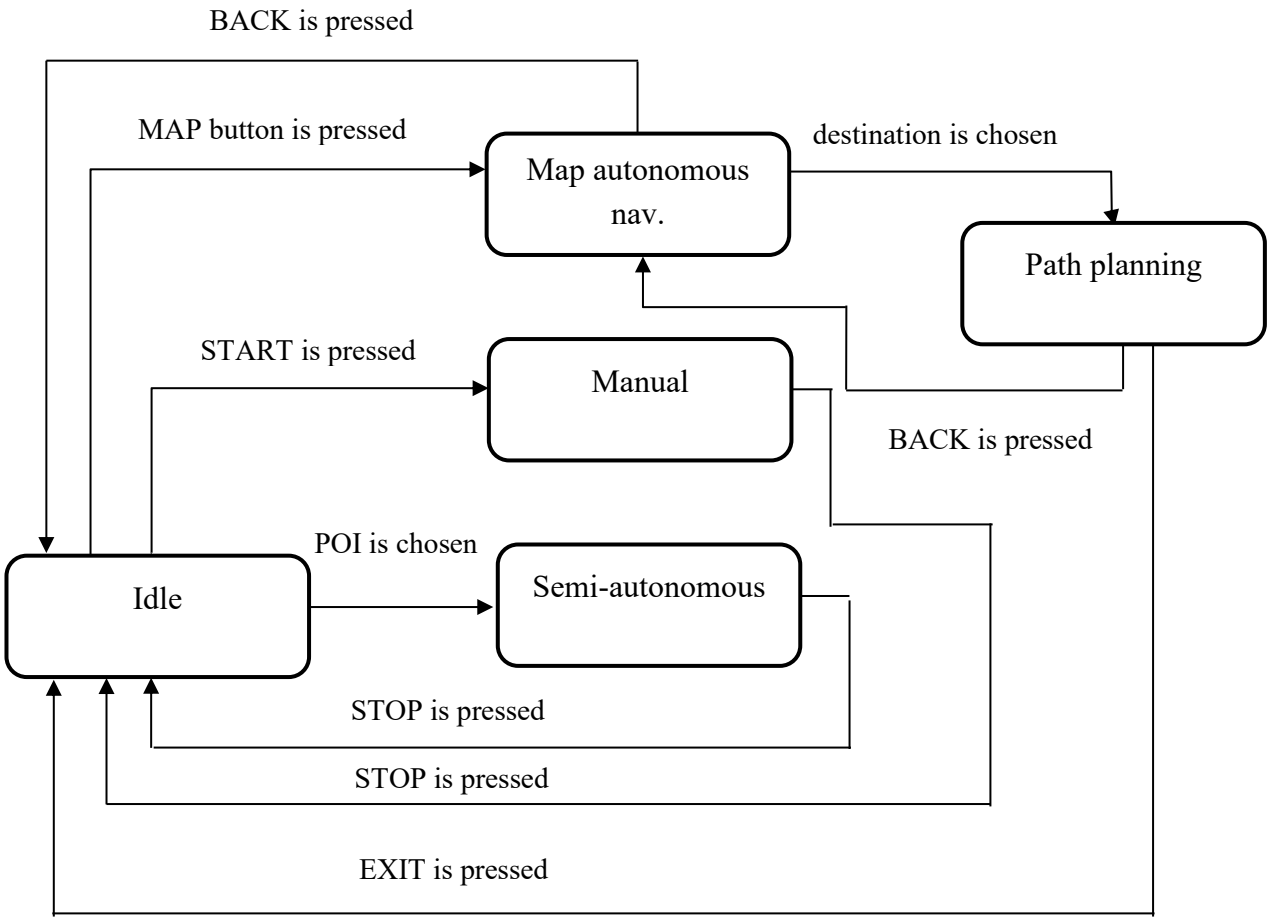


Figure 24 Developed State Machine of the Human-Machine Interface

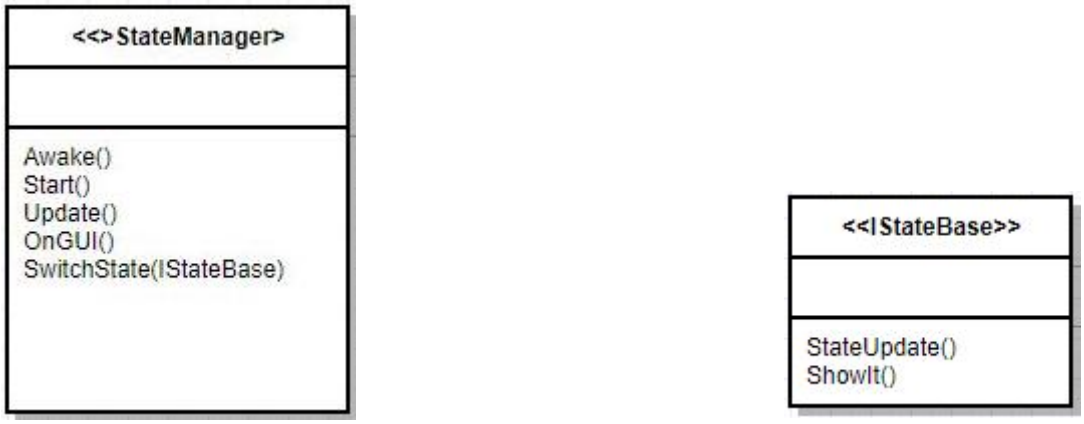


Figure 25 State Manager and IState base classes

The software were give to be tested for experienced and non-experienced users. Due to several limitations, the team of the project is considered as experienced users, while random people have no previous experience with a wheelchair or eye-tracking devices usage. For this case, we selected three team members as experienced users to test the software with use of mouse and eye-tracking device and five unexperienced volunteers. The test cases and results are given below. The testing were in held in four sessions, until all bugs were not fixed. Therefore, the test cases below has no fails.

Test Case ID	**Test Steps**	**Expected results**	**Actual results**	**Pass/Fail**	**Comments**
IDLE _1	Gaze on a "Start" button (Green button) for > 2 sec	Cursor - timebar counting lasts for 2 sec, filling is proportional to the time. Afterwards the systes switches to the manual navigation, manual navigation scene appears.	As expected	Pass	Tested online, thus no movement of the wheelchair is expected
IDLE_2	Gaze on a "any" button for < 2 sec	Cursor timebar counting lasts for the input time, filling out accordingly. After gazing is interrupted, the cursor is filled for 100%.	As expected	Pass	
IDLE_3	Gaze on a "MAP" button for 2 sec	The system switches to the "autonomous navigation" state	As expected	Pass	
IDLE_4	Gaze on a "Pin" button for > 2 sec	The system switches to the "semi-autonomous navigation" state, semi-autonomous scene appears	As expected	Pass	

IDLE_5	Gaze on a "Settings" button for > 2 sec	The system switches to the "Settings" state and Settings scene appears	As expected	Pass	
IDLE_6	Move Aruco marker to the right	Suggested path changes accordingly	As expected	Pass	
IDLE_7	Move Aruco marker to the left	Suggested path changes accordingly	As expected	Pass	
IDLE_8	Hide Aruco marker	No suggested path is visible, pin button is not active	As expected	Pass	
IDLE_9	Move Aruco marker online	The path changes accordingly	As expected	Pass	
IDLE_10	Explore the screen with the eyes gaze	The cursor is moving along the eye-gaze	As expected	Pass	
IDLE_11	Try above-mentioned steps on a different monitor's sizes	The size of the buttons and distance between buttons/objects changes accordingly	As expected	Pass	

Table 2 Test cases to be followed for the IDLE state

Test Case ID	**Test Steps**	**Expected results**	**Actual results**	**Pass/Fail**	**Comments**
MANUAL_1	Gaze on "Stop" button for < 2 sec	The movement is stopped, the state is switched to "Idle" mode, cursor filling is changed accordingly	As expected	Pass	Due to limitations, the movement is not performed
MANUAL_2	Gaze on "Stop" button for < 2 sec	Cursor is filled proportionally to the time	As expected	Pass	

Table 3 ***Test cases to be followed for the MANUAL state***

Test Case ID	**Test Steps**	**Expected results**	**Actual results**	**Pass/Fail**	**Comments**
SEMI_AUT_1	Gaze on "Stop" button for < 2 sec	The movement is stopped, the state is switched to "Idle" mode, cursor filling is changed accordingly	As expected	Pass	Due to limitations, the movement is not performed

SEMI_AUT_2	Gaze on "Stop" button for < 2 sec	Cursor is filled proportially to the time	As expected	Pass	

Table 4 Test cases to be followed for the SEMI-AUTONOMOUS state

Test Case ID	**Test Steps**	**Expected results**	**Actual results**	**Pass/Fail**	**Comments**
SETTINGS_1	Gaze on "small" button for > 2 sec	The buttons size are changed to the small size	As expected	Pass	User should check all buttons in all the states, if satisfy then the test case is passed
SETTINGS_2	Gaze on "Stop" button for < 2 sec	Cursor is filled proportionally to the time	As expected	Pass	User should check all buttons in all the states, if satisfy then the test case is passed
SETTINGS_3	Perform above-mentioned steps for the "medium" and "large" buttons	The buttons size are changed to the medium and large sizes accordingly, cursor's filling is changed accordingly	As expected	Pass	User should check all buttons in all the states, if satisfy then the test case is passed

Table 5 Test cases to be followed for the SETTINGS state

Test Case ID	Test Steps	Expected results	Actual results	Pass/Fail	Comments
MAP_PLAN_1	Gaze on "corridor" button for > 2 sec	The corridor room is chosen, POIs appeared	As expected	Pass	POIs are following: EntR3, EntR2, EntR3_1, EntR3_2
MAP_PLAN_2	Gaze on EntR3 for >2 sec	The state is switched to the path planning state, scene path planning is loaded, path is calculated	As expected	Pass	Path is calculated from the initial point to the POI
MAP_PLAN_3	Gaze on EntR2 for >2 sec	The state is switched to the path planning state, scene path planning is loaded, path is calculated	As expected	Pass	Path is calculated from the initial point to the POI
MAP_PLAN_4	Gaze on EntR3_1 for >2 sec	The state is switched to the path planning state, scene path planning is loaded, path is calculated	As expected	Pass	Path is calculated from the initial point to the POI
MAP_PLAN_5	Gaze on EntR3_2 for >2 sec	The state is switched to the path planning state, scene path planning is loaded, path is calculated	As expected	Pass	Path is calculated from the initial point to the POI
MAP_PLAN_6	Gaze on "Room1" button for > 2 sec	The Room1 is chosen, POIs appeared	As expected	Pass	POIs are following Table 1, Table 2. Table 3

MAP_PLAN_7	Gaze on Table 1for >2 sec	The state is switched to the path planning state, scene path planning is loaded, path is calculated	As expected	Pass	Path is calculated from the initial point to the POI
MAP_PLAN_8	Gaze on Table 2 for >2 sec	The state is switched to the path planning state, scene path planning is loaded, path is calculated	As expected	Pass	Path is calculated from the initial point to the POI
MAP_PLAN_9	Gaze on Table 3 for >2 sec	The state is switched to the path planning state, scene path planning is loaded, path is calculated	As expected	Pass	Path is calculated from the initial point to the POI
MAP_PLAN_10	Gaze on "Room 2" button for > 2 sec	The Room 2 is chosen, POIs appeared	As expected	Pass	POIs are following: Table 1 Table2 Table 3
MAP_PLAN_11	Gaze on Table 1for >2 sec	The state is switched to the path planning state, scene path planning is loaded, path is calculated	As expected	Pass	Path is calculated from the initial point to the POI
MAP_PLAN_12	Gaze on Table 2 for >2 sec	The state is switched to the path planning state, scene path planning is loaded, path is calculated	As expected	Pass	Path is calculated from the initial point to the POI
MAP_PLAN_13	Gaze on Table 3 for >2 sec	The state is switched to the path planning state, scene path planning is loaded, path is calculated	As expected	Pass	Path is calculated from the initial point to the POI
MAP_PLAN_14	Gaze on "Room 3" button for > 2 sec	The Room 3 is chosen, POIs appeared	As expected	Pass	POIs is TV
MAP_PLAN_15	Gaze on TV for >2 sec	The state is switched to the path planning state, scene path planning is loaded, path is calculated	As expected	Pass	Path is calculated from the initial point to the POI
MAP_PLAN_16	Gaze on Exit for >2 sec	The state is switched to the Idle state, scene Idle is loaded	As expected	Pass	

Table 6 Test cases to be followed for the MAP AUTONOMOUS state

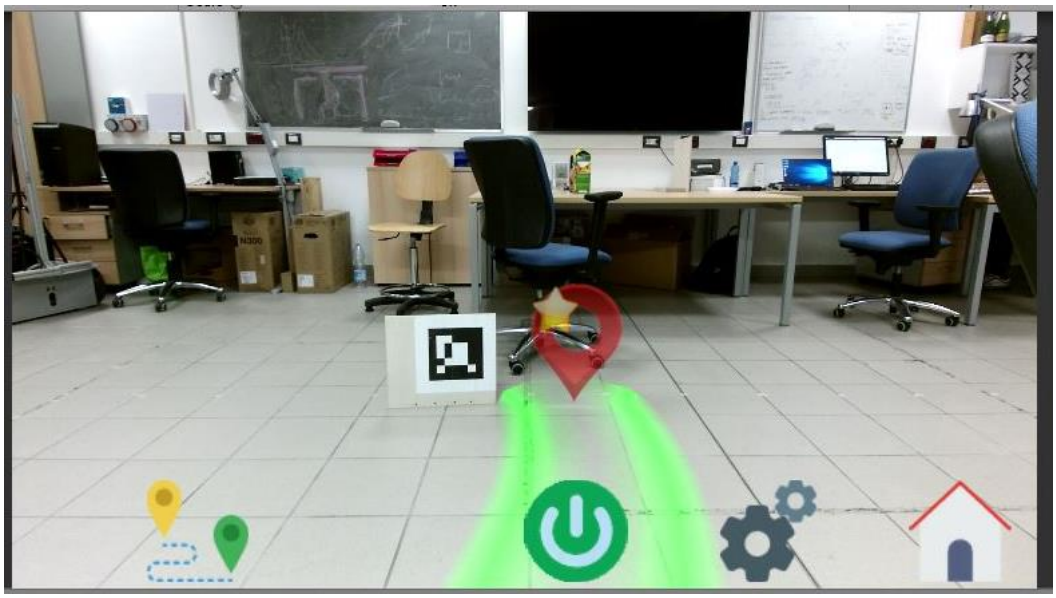


Figure 26 Main screen human machine interface

The image is obtained from the kinect acquisition service and rendered in front of a camera view and it is updated in real time. Here are four buttons: autonomous navigation map, start, settings and home button. The user can "press" the button by keeping the eye contact for 2 seconds. While the user hovering button, application runs timer ensuring user friendly design. By pressing start button user can navigate in manual mode. The application communicating with services, when detects Aruco marker – creates a button in shape of a navigation pin near it. In addition, to that path is calculated and illustrated graphically from the current location to the point of interest. If customer prefer to navigate in semi-autonomous mode can hover this navigation pin. Home button is reserved for future purposes, because navigation part will be a part of ready software, which is going to have more functionalities such as phone calls, email receiving and sending, calendar, etc. In menu settings envisaged speed changes - low, medium and high, in addition to that button's size – small, medium and high as it can be seen from a figure below.

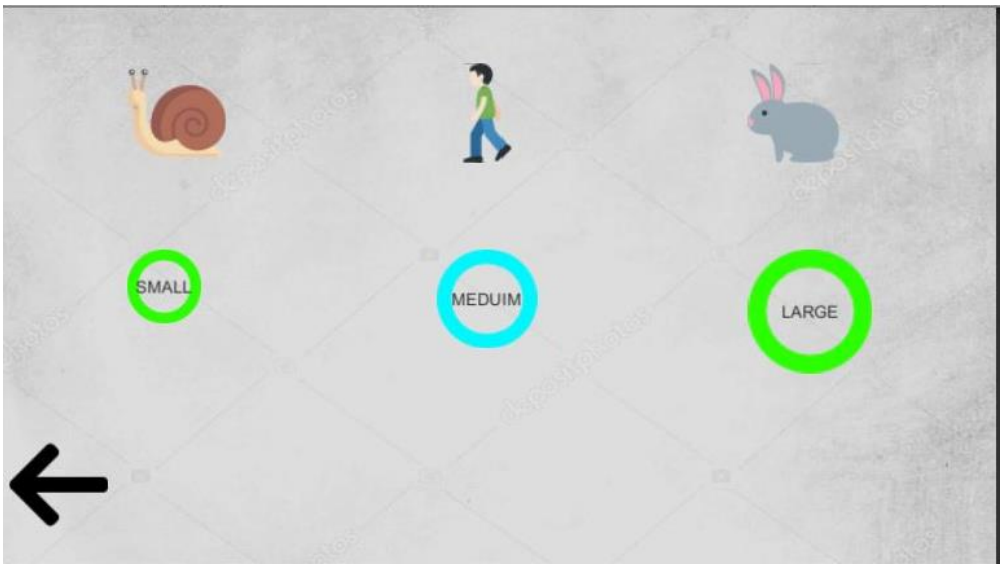


Figure 27 Menu settings

In manual or autonomous navigation the user, see following screen (Figure 28)

Figure 28 Manual navigation mode

Figure 29 Semi-autonomous navigation mode

In this case functionalities are the same, if stop buttons is pressed application stops navigation and scene is changed to main scene. Also if the user does not look anymore at the screen, the navigation is stopped.

In order to check UI design user's satisfaction the anonymous survey were conducted. Volunteers were invited to test the interface using eye-tracking device or an adapted Marvel [22] prototype where a mouse pointer is simulating eye-tracking output. To check how intuitive UI is, responders were asked to use an UI or a prototype with no instruction. Almost 90% of responders reported that the design of the interface is simple and intuitive, while the rest mentioned that they had some misunderstanding while using UI. There were 20 positive responses to the question "Would you like to use this application again?", while five were not certain. In the comment field, those users mentioned that icons design were poor, therefore the author enabled opportunity to customize icons design according to each user preferences. In summary, the results show that created interface satisfy basic user experience and confirm need to be adaptive.

4.3 Path planning solution

4.3.1 Environmental map validation, graph based adaptation

It consists information about a marker related point of interest and links, or rooms that can be reached from this particular point. For user experience in order to ensure readability and user-friendly design added name of the room and its points of interest. Developed application reads information from file generates a graph and offers a user possibility to choose a destination (See Figure 30).

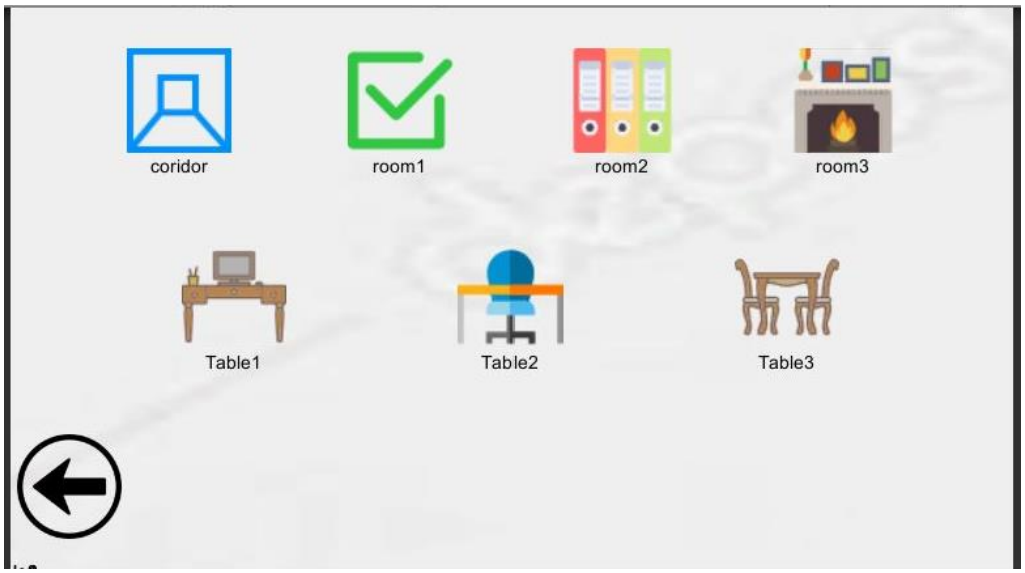


Figure 30 User interface, rooms menu

From the map application read that there are four rooms, after choosing room1 user can choose POs inside this room which are: Table 1, Table 2 and Table 3. Buttons position, captions are dynamic, depending on a JSON file content, and amount of buttons and the size of a screen. As regards design of icons, the user can easily change it by replacing default pictures in related folder. After choosing destination point, the wheelchair define current position and calculates shortest path from detected starting point to the chosen by the user a target point. The solution is shown as follows (Figure 31)

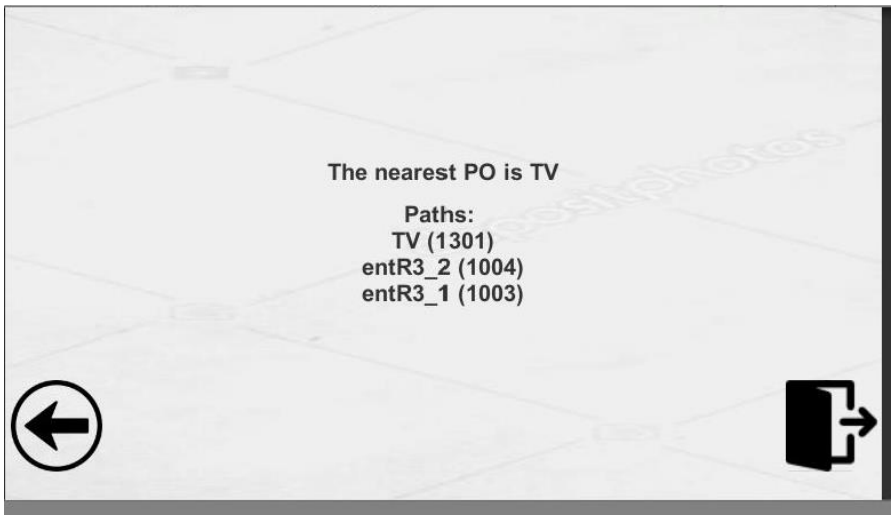


Figure 31 Path planning scene user interface

At this point of software development, the wheelchair is able to move using a list of passing point, when facing an obstacle application stops movement. However, the team developing a new solution which should perform smooth obstacle avoidance algorithm, which is under development and cannot be implemented yet. Therefore, for testing purposes was decided to print out suggested path.

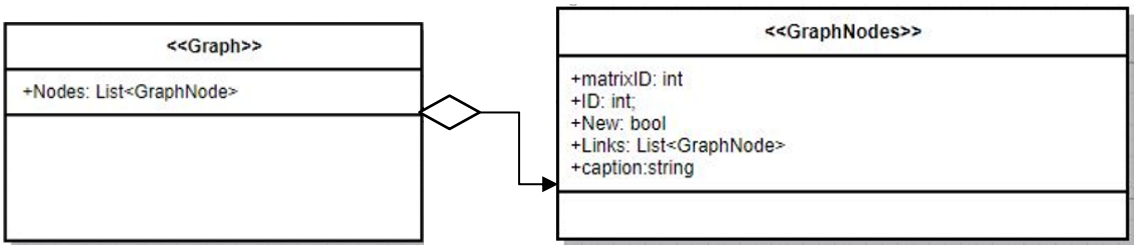


Figure 32 Class diagram Graph, GraphNodes

4.3.2 Dijkstra algorithm simulation

In this work, the environmental map is represented as a graph of point of interest and can be seen as follows.

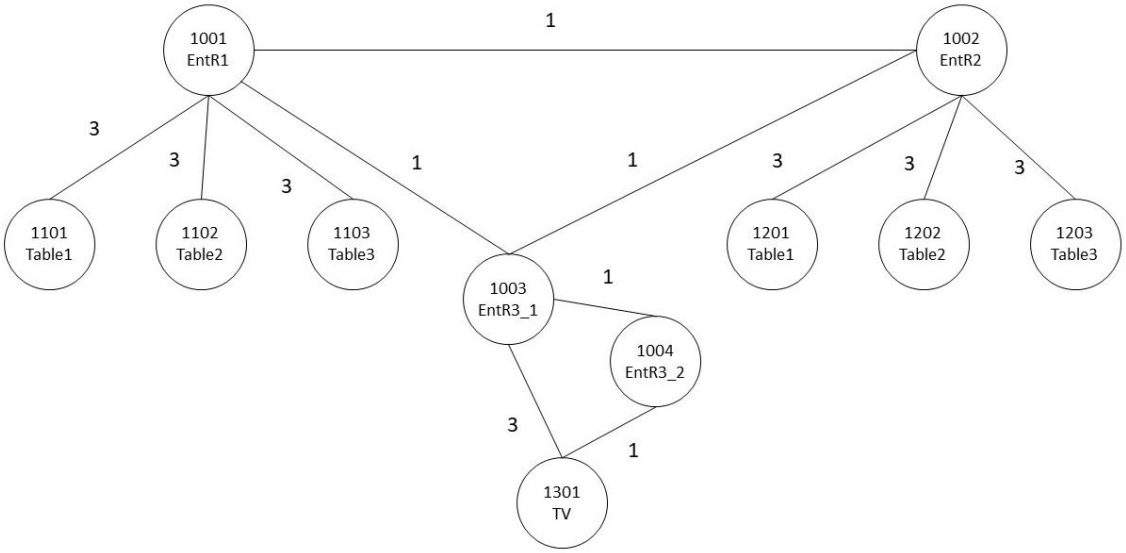


Figure 33 Graph-based map

Above-mentioned map is formed in order to check if path planning works well and gives adequate results. However, for user comfort results are shown in a readable form Figure 31**Ошибка! Источник ссылки не найден.**.

The simulation were run from every node to each left nodes in this particular laboratory map, results are shown below in the tables Table 7 -Table 9Table 16.

1001	1101		
1001	1102		
1001	1103		
1001	1002		
1001	1003		
1001	1002	1201	
1001	1002	1202	
1001	1002	1203	
1001	1003	1004	
1001	1003	1004	1301

Table 7 Simulation results from node 1001 to the rest of nodes

1002	1001		
1002	1003		
1002	1201		
1002	1202		
1002	1203		
1002	1001	1101	
1002	1001	1102	
1002	1001	1103	
1002	1003	1004	
1002	1003	1004	1301

Table 8 Results of simulation from the node 1002 to the rest of nodes

1003	1001	
1003	1002	
1003	1004	
1003	1004	1301
1003	1001	1103
1003	1001	1102
1003	1001	1101
1003	1002	1201
1003	1002	1202
1003	1002	1203

Table 9 Results of simulation from node 1003 to the rest of nodes

1004	1003
1004	1301

1004	1003	1001		
1004	1003	1002		
1004	1003	1001	1101	
1004	1003	1001	1102	
1004	1003	1001	1103	
1004	1003	1002	1201	
1004	1003	1002	1202	
1004	1003	1002	1203	

Table 10 Results of simulation from the node 1004 to the rest of nodes

1101	1001			
1101	1001	1102		
1101	1001	1103		
1101	1001	1002		
1101	1001	1002	1201	
1101	1001	1002	1202	
1101	1001	1002	1203	
1101	1001	1003		
1101	1001	1003	1004	
1101	1001	1003	1004	1301

Table 11 Results of simulation from the node 1101 to the rest of nodes

1102	1001			
1102	1001	1101		
1102	1001	1103		
1102	1001	1003		
1102	1001	1002		
1102	1001	1002	1201	
1102	1001	1002	1202	
1102	1001	1002	1203	
1102	1001	1003	1004	
1102	1001	1003	1004	1301

Table 12 Results of simulation from the node 1102 to the rest of nodes

1103	1001		
1103	1001	1101	
1103	1001	1102	
1103	1001	1003	
1103	1001	1002	
1103	1001	1002	1201
1103	1001	1002	1202
1103	1001	1002	1203

1103	1001	1003	1004	
1103	1001	1003	1004	1301

Table 13 Results of simulation from the node 1103 to the rest of the nodes

1201	1002			
1201	1002	1001		
1201	1002	1003		
1201	1002	1202		
1201	1002	1203		
1201	1002	1001	1101	
1201	1002	1001	1102	
1201	1002	1001	1103	
1201	1002	1003	1004	
1201	1002	1003	1004	1301

Table 14 Results of simulation from the node 1201 to the rest of the nodes

1202	1002			
1202	1002	1001		
1202	1002	1003		
1202	1002	1201		
1202	1002	1203		
1202	1002	1001	1101	
1202	1002	1001	1102	
1202	1002	1001	1103	
1202	1002	1003	1004	
1202	1002	1003	1004	1301

Table 15 Results of simulation from the node 1202 to the rest of the nodes

1203	1002			
1203	1002	1001		
1203	1002	1003		
1203	1002	1201		
1203	1002	1202		
1203	1002	1001	1101	
1203	1002	1001	1102	
1203	1002	1001	1103	
1203	1002	1003	1004	
1203	1002	1003	1004	1301

Table 16 Results of simulation from the node 1203 to the rest of the nodes

These results indicate that the path planning algorithm adequate and can be used for navigation purposes. Implemented solution is simple and efficient, does not require many calculations. Moreover, mapping, localization, path routines tested to be fit into adaptive and intuitive interface. User can change one eye-tracker to one another or Kinect camera with another ToF camera. These changes will not significantly effect code of the project. In addition to that, UI is adaptive to screen sizes, resolution etc, which was separately tested by changing resolution of Unity dashboard.

5. CONCLUSIONS AND FUTURE IMPROVEMENTS

There are some solutions for the wheelchair navigation available for people with major disabilities, however, those solutions face lack of usability due to limited functionalities or high cost. Human machine interface based on eye-tracking device, Kinect camera, combining augmented reality and greatly developed software is one of the promising directions in this areas. Path finding strategies are well-studied, thus, benefits and limitations are quite known. One of the goals of thesis was to find the most appropriate solution according to the given requirements. Therefore, suggested solution might be not the most technologically advanced but provides the required usability in trade off the suitable price.

Developed solution for path finding satisfy the requirements detected by the author. Note, it is needed to install Aruco markers inside apartment of the user before using the wheelchair. It is also necessary to define the points of interest to be included in autonomous navigation map. This significantly reduces cost of development, however, requires additional work of engineers and if user decides to move to another apartment, the map should be recreated from a scratch.

UNITY Game Engine is used for the development of the HMI. Moreover, ArUco was used as a library for Augmented Reality AR applications based on OpenCV to identify the possible targets and then OpenCV for image processing. The proposed AR-based application can recognize POIs visible to the camera, to plan a path and to give to the patient the possibility to eventually perform the preferred path after a proper checking. From an applicative point of view, when a POI enters in the field of view of the camera the user can select it, starting in this way the autonomous navigation. The tests on the application developed were performed evaluating the repeatability of the maneuvers starting from different positions. In this way, the impact of the uncertainty of the camera position was evaluated, with respect to the wheelchair, on the reached position. Moreover it is used also Microsoft Visual Studio for the image processing and Matlab.

Developed human-machine interface is reusable, meaning that if eye tracking device or TOF camera will be replaced with another one, for example Kinect camera replaced by SensEye camera, the interface still will be useful and will not require a lot of changes. Design of interface is easily changeable according to the needs and preferences of the user, there is a folder with icons, these icons can be used by default or replaced with the preferred ones. In addition, to that interface is adjustable to any resolution and screen size.

During the research, it was clear that:

- Mapping using CAD significantly reduces cost of the development. It is proven to be simple and elegant solution, because the environment stays the same. To be short, it is simple, inexpensive and suitable for autonomous navigation in the flat of a patient.
- The localization of the wheelchair inside the environment is necessary to allow a mobile robot to move autonomously. ArUco markers is cheap and suitable solution to be used to perceive relative position and define POIs.
- Graph-based map created using separately created JSON file and CAD is convenient addition to Dijkstra algorithm. This algorithm proven to be classical and simple approach in path planning. The path planning was fast and did not require any improvement.
- Last but not the least; State Machine is great solution to create UI. It not only allows including previously developed code but also gives future developers opportunity to add new states, keeping code simple and clear.

During the thesis completion, the project faced many different changes proposed by stakeholders; therefore, the interface's and autonomous navigation solutions have been modified significantly. Due to high level of bureaucracy in Italy, it was impossible to test the solution with real users. The application for testing permission was submitted around a year prior to the publication of this thesis while the approval is still pending. Due to above-mentioned limitations, the final solution was tested only in laboratory premises. Project is dynamic and ongoing, therefore the wheelchair was not available for the final testing. However, in the nearest future the team is expecting to test this solution on a real wheelchair and results will be demonstrated in September for the stakeholders.

6. REFERENCES

[1] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss and Wolfram Burgard, "A tutorial on Graph-Based SLAM," *IEEE INTELLIGENT TRANSPORTATION SYSTEMS MAGAZINE,* no. Winter, pp. 31-43, 2010.

[2] M. De Cecco and L. Maule, "Roboeye-project description," [Online]. Available: http://www.miro.ing.unitn.it/index.php/robotics/191-roboeye-project.

[3] D. L. Jaffe, "An ultrasonic head position interface for wheelchair control," *Journal of Medical Systems,* p. pp. 337–342, 1982.

[4] H. A. Yanco, "Wheelesley: A robotic wheelchair system: Indoor navigation and user interface," *Assistive Technology and Artificial Intelligence,* pp. pp. 256-268, 31 May 2006.

[5] F. E. R. g. o. t. S. p. M. Mazo, "the Modularity if the Electronic Guidance Systems of the SIAMO Wheelchair Allows for the User-Specific Adaptability Based on Environment and Degree of Handicap," *An Integral System for Assisted Mobility,* pp. 46-56.

[6] Zhengang Li, Yong Xiong and Lei Zhou, "ROS-Based Indoor Autonomous Exploration and Navigation Wheelchair," in *10th International Symposium on Computational Intelligence and Design*, Wuhan, Hubei, China, 2017.

[7] Y. X. L. Z. Zhengang Li, "ROS- Based Indoor Autonomous Exploration and Navigation Wheelchaur," in *10th International Symposium on Computational Intelligence and Design*, 2017.

[8] M. N. M. J. Khaoula Maatoug, "International Conference on Advanced Systems and Electric Technologies," in *Autonomous Wheelchair Navigation in Indoor Environment Based on Fuzzy Logic Controller and Intermediate Targets*, 2017.

[9] R. M. Kohei Arai, "A prototype of Electric Wheelchair Controlled by Eye-Only for Paralyzed User," Saga, Japan, 2010.

[10] A. A. J. S. A. H. H. S. Martin Tall, "Gaze - controlled driving," Spotlight on Works in Progress, Session 2, Boston, MA, USA, April 4-9, 2009.

[11] "Autonomous Mobile Robot Navigation Using Passive RFID in Indoor Environment," *IEEE TRANSACTIONS ON INDUTRIAL ELECTRONICS, VOL 56, NO 7,* July 2009.

[12] U. developer, "Unity Documentation," 23 Februart 2018. [Online]. Available: https://docs.unit3d.com.

[13] A. Duchowski, Eye Tracking Methodology: Theory and Practice. 2nd ed., London: Springer-Verlag London Limited, 2007.

[14] "Tobii: This is Eye Tracking," [Online]. Available: https://www.tobii.com/group/about/this-is-eye-tracking/. [Accessed May 2018].

[15] De Cecco, Mariolino; Baglivo, Luca; Pertil, Marco;, "Real-time uncertainty estimation of odometric trajectory as a function of the actual manoeuvres of autonomous guided vehicles.," *In Advanced Methods for Uncertainty Estimation in Measurement,* pp. 80-85, 2006.

[16] R. M.-S. F. J. M.-C. a. M. J. M.-J. Sergio Garrido-Jurado, "Automatic generation and detection of highly reliable fiducial markers under occlusion.," *Pattern recognition,* pp. 2280-2292, 2014.

[17] Y. Y. Z. M. Xu G.B, "The Present Simulation and Future of Technologies of Intelligent Mobile Robots," *Robot Technology and Application,* pp. 29-34, 2007.

[18] W. H. Yin Chao, "International Conference on Computer Design and Appliations (ICCDA)," in *Developed Dijkstra Shortest Path Search Algorithm and Simulation* , Zibo, China, 2010.

[19] D. E. W., "A note on problems in connection with graphs," *Numer Math ,* pp. 269-271, 1959.

[20] W. Fen, "Dijkstra shortest path algorithm in the car navigation research and real," Shanghai: Shanghai Normal University, 2006.

[21] Z. Z. X. Y. Guo Qing, "Path-planning of Automated Guided Vehicle Based on Improved Dijkstra Algorithm," College of Information Science and Technology, Beijing University of Chemical Technology , Beijing.

[22] "Marvel prototyping online tool for UX designers," [Online]. Available: https://marvelapp.com/.

[23] A Harrison, G. Derwent, A. Enticknap, FD Rose and EA Attree , "The role of virtual reality technology in the assessment and training of inexperienced powered wheelchair users," *Disability and rehabilitation,* Vols. 11-12, no. 24, pp. 599-606, 2002.

[24] Aayush Patial, Dhvanil Mandalia, Nikhil Nandoskar, G.T.Haldankar and P.V.Kasambe, "FIS based autonomous navigation," in *8th ICCCNT 2017*, Delhi, India, July 3-5.

[25] Chaoqun Wang, Lili Meng, Sizhen She, Ian M. Mitchell, Teng Li, Frederick Tung, Weiwei Wan, Max. Q. -H. Meng and Clarence W. de Silva, "Autonomous Mobile Robot Navigation in Uneven and Unstructured Indoor Environments," in *International Conference on Intelligent Robots and Systems (IROS)*, Vancouver,Canada , 2017.

[26] W. Coldsone, Unity 3x Game Development Essentials (Game Development with C# and JavaScript), Birmingham - Mumbai: Packt Publishing , 2011.

[27] R. H. Creighton, Unity 4.x Game Development by Example Beginner's Guide, A seat-of-your-pants manual for building fun, groovy little games with Unity 4.x, Birmingham-Mumbai: Packt Publishing , 2013.

[28] K. D'Aoust, Unity Game Development Scripting, Write efficient C# scripts to create modular key game elements that are usable for any kind of Unity project, Birmigham: Packt Publishing, 2014.

[29] F. Duchoň, "Path Planning with Modified A Star Algorithm for a Mobile Robot," *Procedia Engineering,* pp. 59-69, 2014.

[30] N. C. U. A. A. G.Pires, "Autonomous Wheelchair for Disabled People," Institute of Systems and Robotics, University of Coimbra, Polo II, Coimbra, Portugal.

[31] Hafid Niniss and Adbellah Nadif, "Simulation of the behaviour of a powered wheelchair using virtual reality," in *In 3rd International Conference on Disability, Virtual Reality and Associated Technologies*, 2000.

[32] Ian Stott and David Sanders, "The use of virtual reality to train powered wheelchair users and test new wheelchair systems," *International Journal of Rehabilitation Research ,* vol. 4, no. 23, pp. 321-326, 2000.

[33] Khaoula Maatoug, Malek NJAH and Mohamed JALLOULI, "Autonomous Wheelchair Navigation in Indoor Environment Based on Fuzzy Logic Controller

and Intermediate Targets," in *International Conference on Advanced Systems and Electric Technologies (IC_ASET)*, 2017.

[34] Luca Maule, Alberto Fornaser, Malvina Leuci, Nicola Conci, Mauro Da Lio and Mariolino De Cecco, "Development of innovative HMI strategies for eye controlled wheelchairs in virtual reality," in *In International Conference on Augmented Reality, Virtual Reality and Computer Graphics*, Springer, 2016.

[35] M. De Cecco, A. Fornaser, M. Leuci, N. Conci, M.Daldoss and L.Maule, "Eye trackers uncertainty analysis and modelling," in *XXIII A.I.VE.LA. National Conference*, Perugia, Italy, 2015.

[36] Mariolino De Cecco, Matteo Zanetti, Alberto Fornaser, Malvina Leuci and Nicola Conci, "INTER-EYE: Interactive error compensation for eye-tracking devices," in *12th International A.I.VE.LA. Conference on Vibration Measurements by Laser and Noncontact Techniques*, Ancona, Italy, 2016.

[37] Muhannad Mujahed, Dirk Fischer and Barbel Mertsc, "Robust Collision Avoidance for Autonomous Mobile Robots in Unknown Environments," GET Lab, University of Paderborn,Pohlweg 47-49, 33098 , Paderborn, Germany.

[38] T. Norton, Learning C# by Developing Games with Unity 3D, Mumbai: Packt Publishing, 2013.

[39] A. Okita, Learning C# Programming with Unity 3D, New York: CRC Press, Taylor&Francis Group, 2015.

[40] Rainer Kummerle, Michael Ruhnke, Bastian Steder, Cyrill Stachniss and Wolfram Burgard, "Autonomous Robot Navigation in Highly Populated Pedestrian Zones," Department of Computer Science, University of Freiburg, Freiburg, Germany.

[41] A. Thorn, Learn Unity for 2D Games Development (Technology in Action), New York: Apress, 2013.

[42] Yimin Zhou, Guolai Jiang, Guoqing Xu, Xinyu Wu and Ludovic Krundel, "Kinect Depth Image Based Door Detection for Autonomous Indoor Navigation," in *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, Edinburgh, Scotland, UK,, August 25-29, 2014.

APPENDIX A:

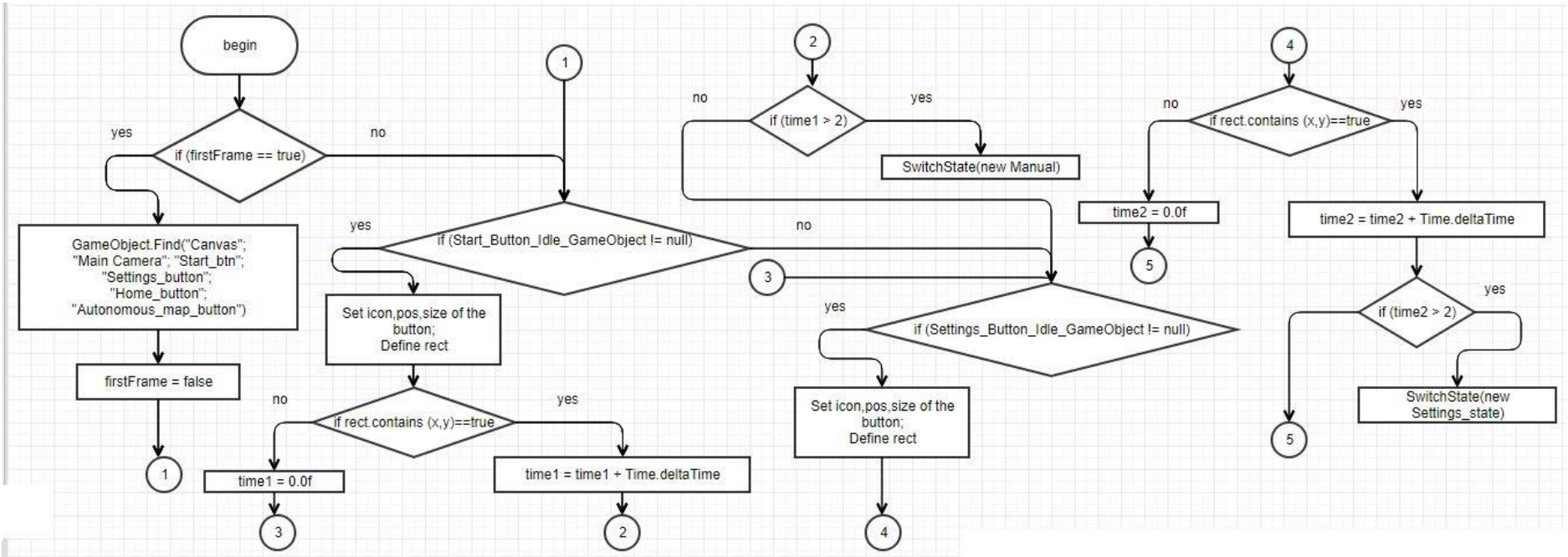


Figure 34 Scene Idle, flow chart, StateUpdate() part 1/2

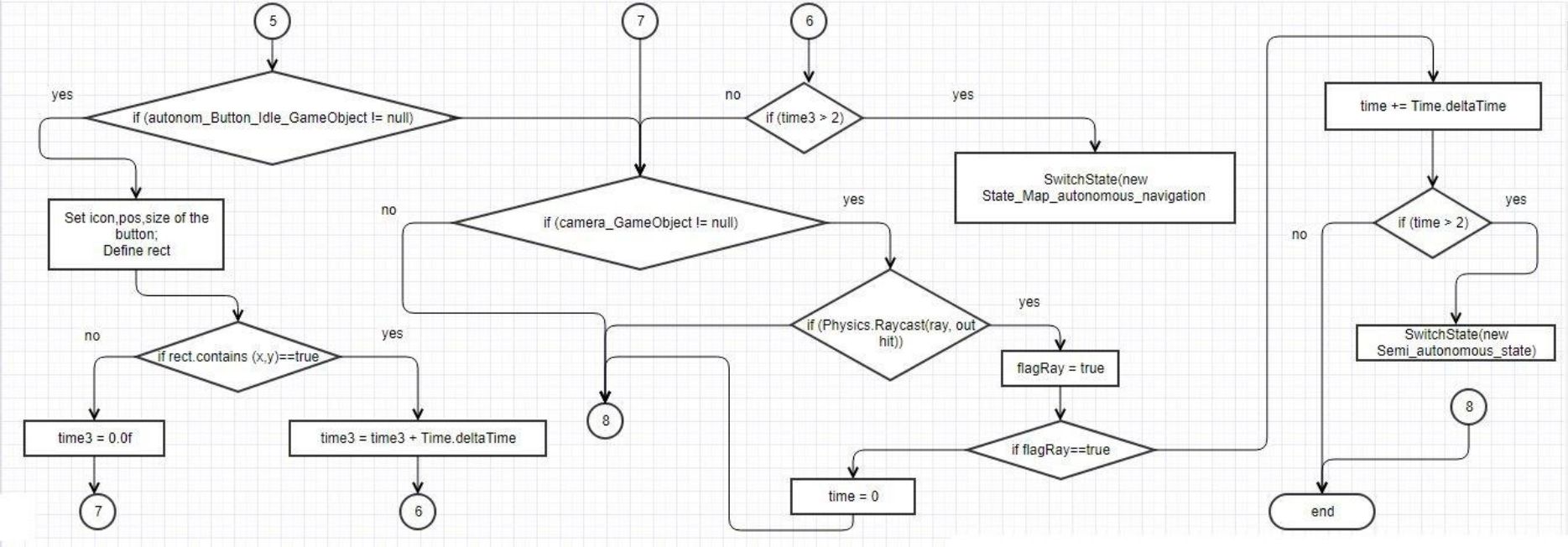


Figure 35 Scene Idle, flow chart, StateUpdate(), part 2/2

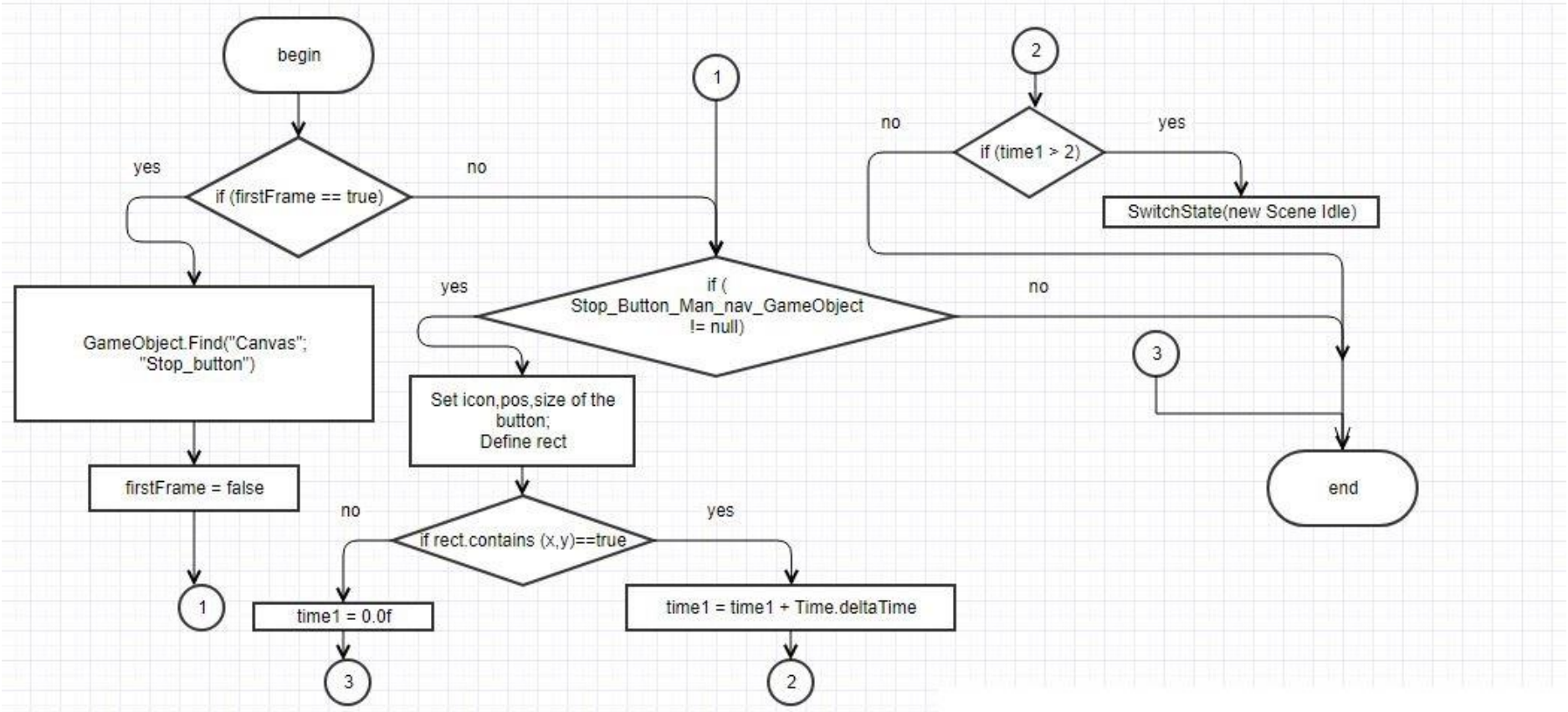


Figure 36 Manual/Semi-autonomous scenes, flow chart, StateUpdate()

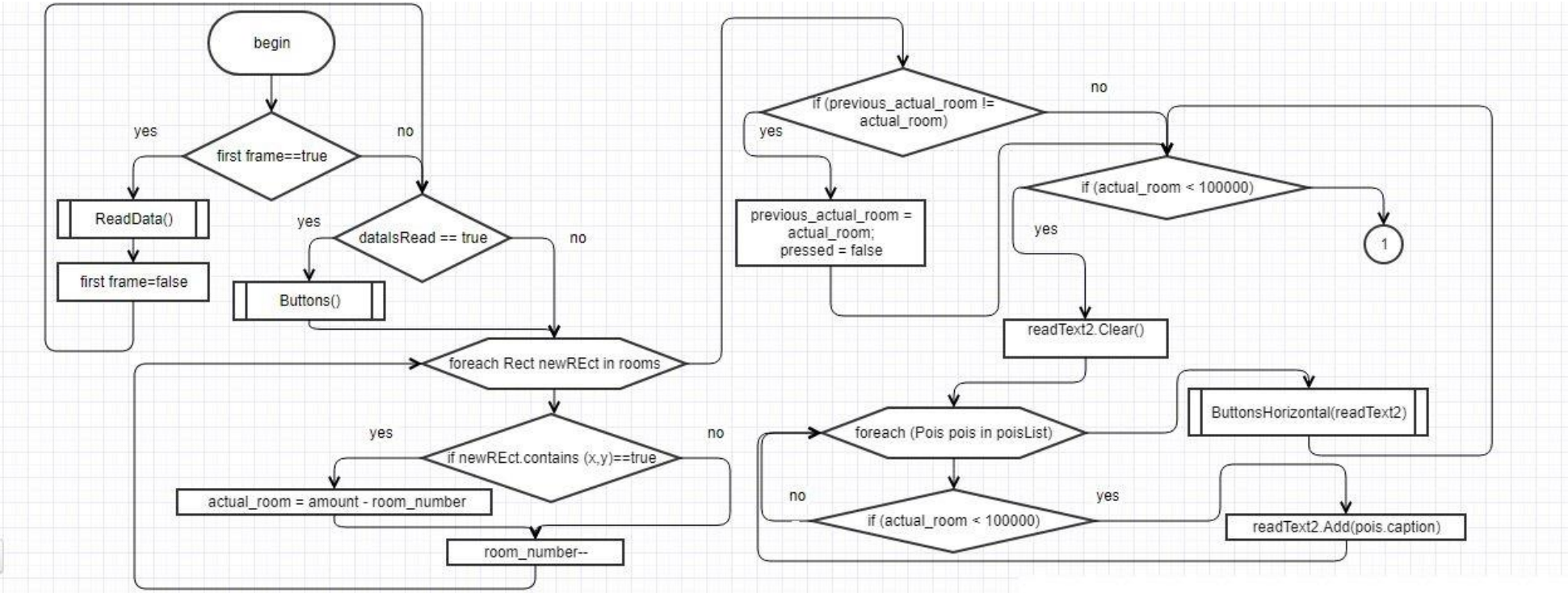


Figure 37 Map autonomous navigation, Flow chart, StateUpdate(), part 1/2

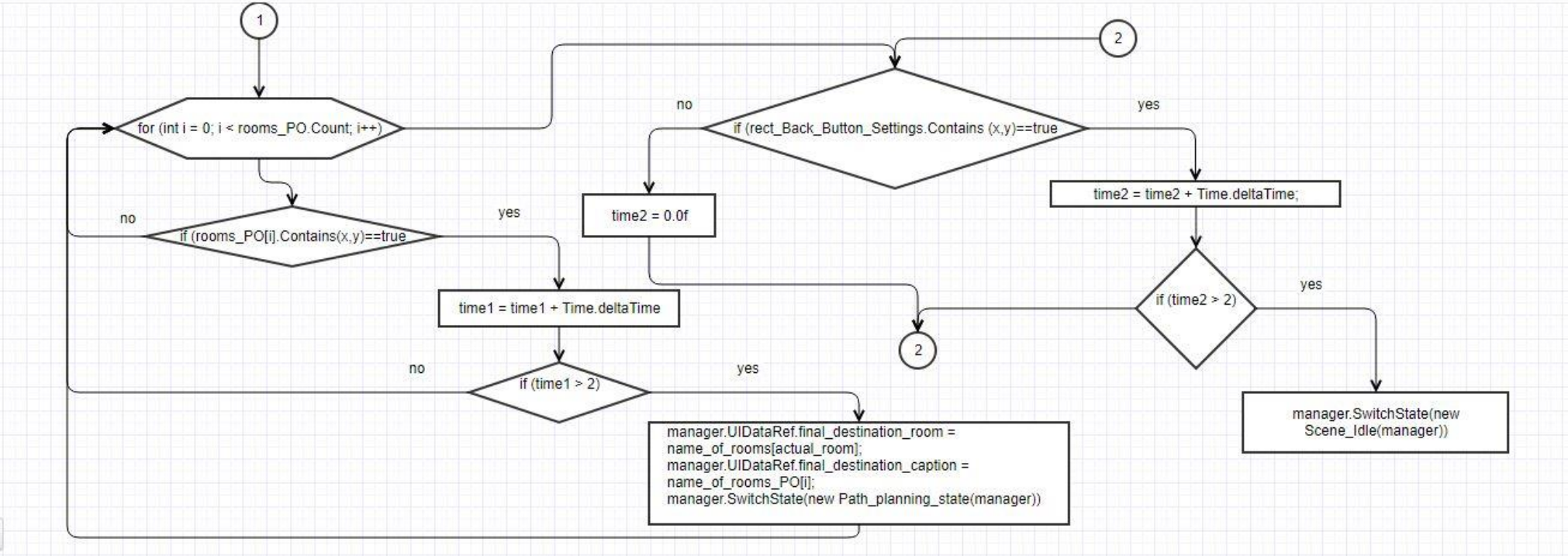


Figure 38 Map autonomous navigation, Flow chart, StateUpdate(), part 2/2

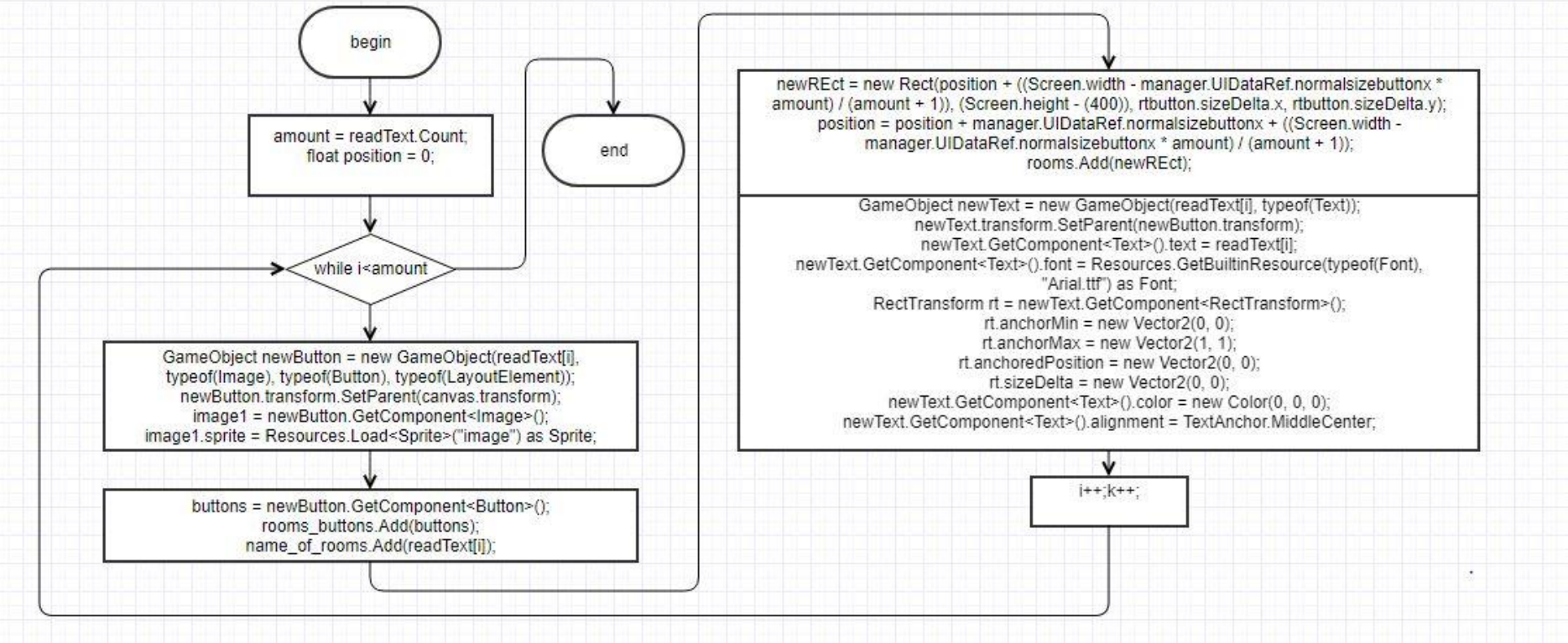


Figure 39 Map autonomous navigation, flow chart, Buttons()

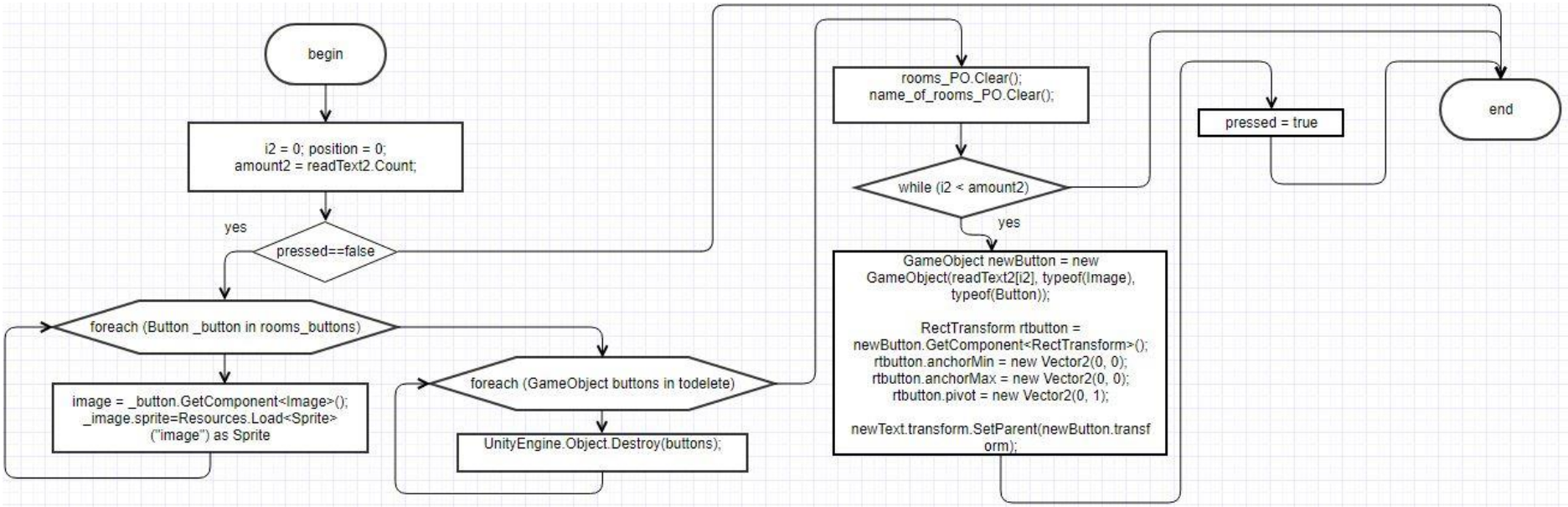


Figure 40 Map autonomous navigation, flow chart, ButtonsHorizontal()

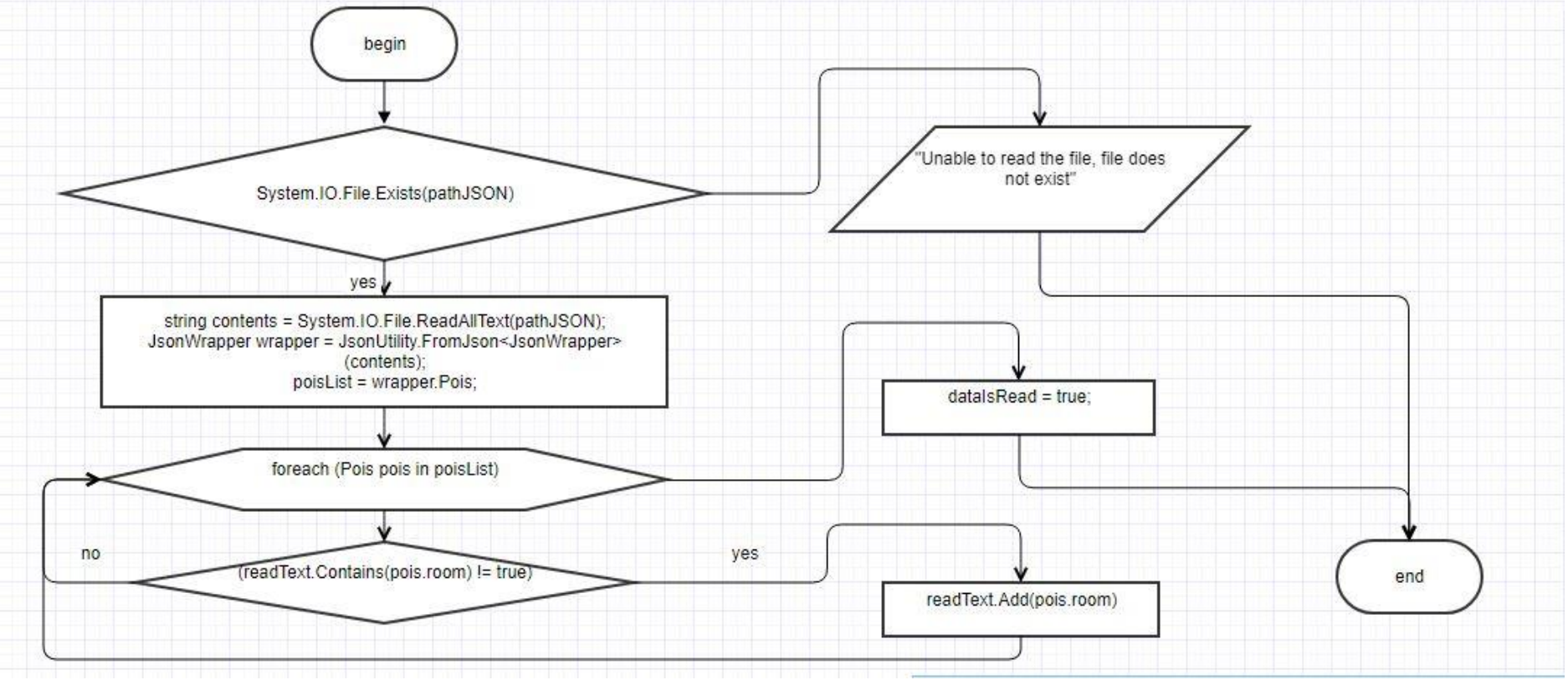


Figure 41 Map autonomous navigation, flow chart, ReadData()

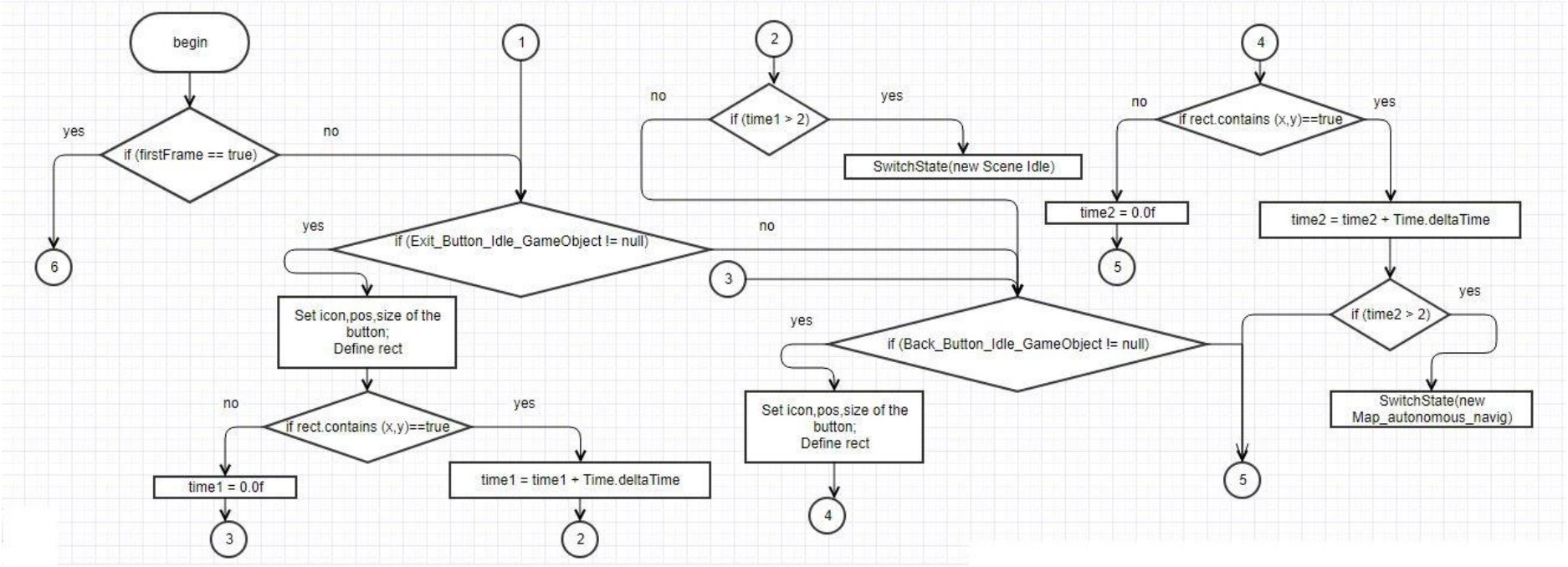


Figure 42 Path planning, flow chart, StateUpdate(), part 1/2

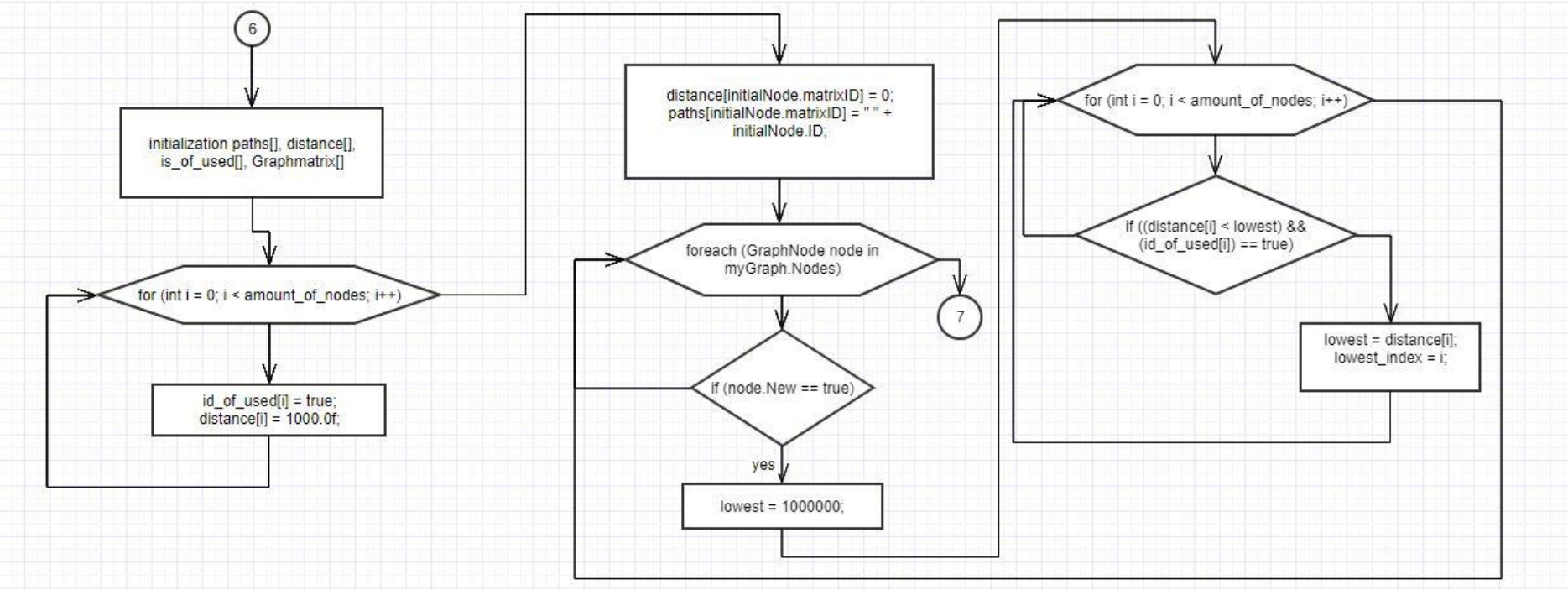


Figure 43 Path planning, flow chart, StateUpdate(), part 2/2

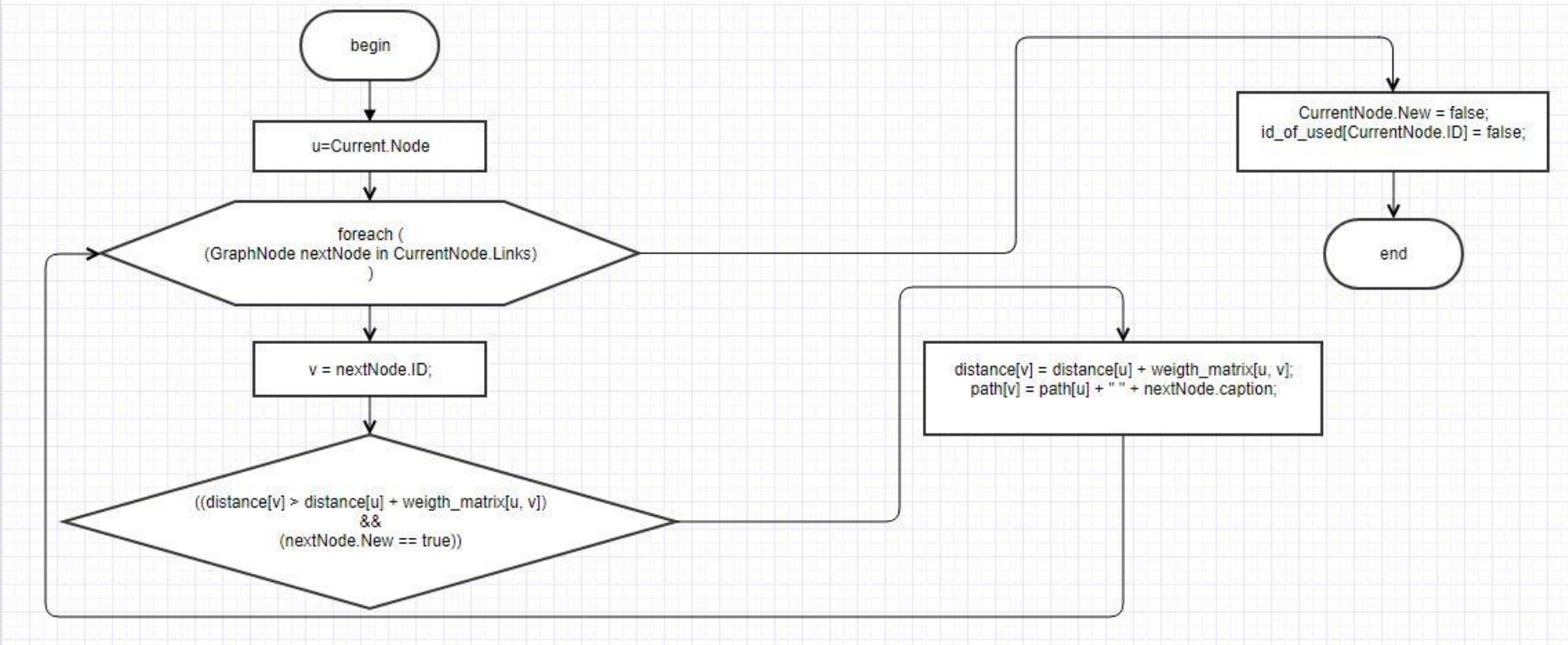


Figure 44 Path planning, flow chart InternalFind()

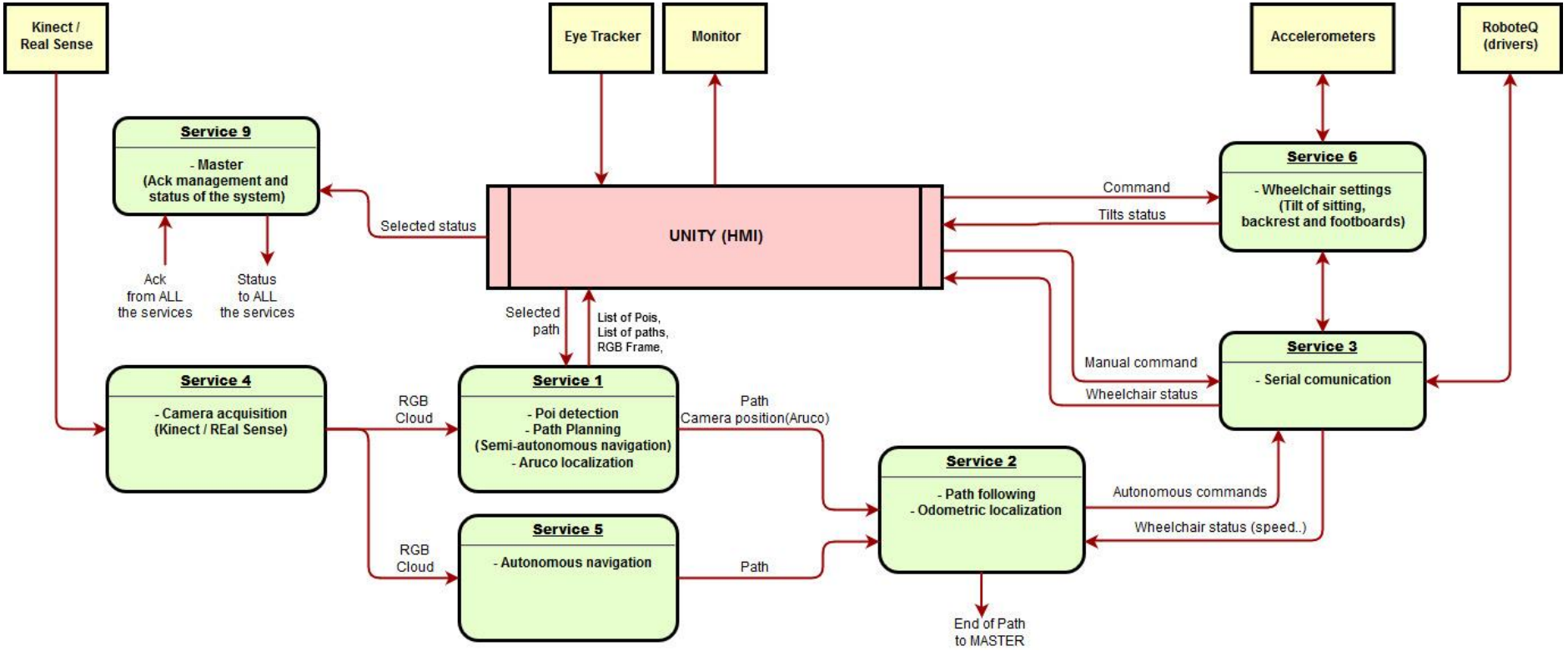


Figure 45 Software architecture scheme