České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačů

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Jedlička Jan

Studijní program: Otevřená informatika
Obor: Softwarové inženýrství

Název tématu: Intermediální systém navigace v chodbách

Pokyny pro vypracování:

Prostudujte metody používané pro identifikaci a sledování pohybu osob v uzavřeném prostoru (chodby, vstupní haly) založené na programovatelných modulech s bezdrátovou komunikací, případně mobilních telefonech a proveďte jejich srovnání z hlediska náročnosti jejich implementace a nasazení v reálných podmínkách (použitelnost pro různé cílové skupiny uživatelů, vhodnost pro různé typy budov, infrastruktury, atd.). Vyberte metodu sledování uživatele vhodnou pro vnitřní strukturu chodeb budovy fakulty a halových laboratoří a použijte ji v návrhu konkrétního řešení navigačního systému, který bude vzdáleně konfigurovatelný, rozšiřitelný a bude k navigaci využívat audiovizuální výstup. Navigační systém zrealizujte. Při řešení řízení audiovizuálního výstupu využijte zkušenosti z vlastní bakalářské práce. Dále navrhněte metodu testování tak, aby bylo možné prokázat výše požadované vlastnosti navigačního systému a jeho spolehlivost. Navigační systém otestujte v režimu navigace jednoho i více uživatelů současně v budově halových laboratoří fakulty. V testovacích scénářích zohledněte alespoň dvě trasy, dále návrat bloudícího uživatele na trasu a odolnost proti svedení uživatele z cesty. Výsledky porovnejte s alespoň dalšími dvěma známými přístupy. Zobecněte v textu použití navigačního systému také pro jiné vnitřní prostory.

Seznam odborné literatury:

[1] Bajko, G. Method providing positioning and navigation inside large buildings. US Patent 8,259,692. 2012

Vedoucí: Ing. Roman Berka, Ph.D.

Platnost zadání do konce zimního semestru 2018/2019

L.S.

prof. Dr. Michal Pěchouček, MSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 18.4.2017

Czech Technical University in Prague

Faculty of Electrical Engineering

Department of Computer Science

Intermedia Corridor Navigation System

Jan Jedlička

Master's Thesis

Supervisor: Ing. Roman Berka, PhD.

25th May 2017

Acknowledgements

I’d like to thank the supervisor of this thesis Mr. Ing. Roman Berka, PhD. for the support and advices, he has given me during the development of this thesis. Furthermore, I’d like to thank Blue Sky Media company for letting me to work on this thesis and test it in their offices.

Declaration

I hereby declare that I wrote this work independently and I quoted all the sources used in this thesis in accord with Methodical instructions about ethical principles for writing academic theses.

Prague, 25th May 2017

……..

Abstract (EN)

Goal of this work is to research available options of localization and tracking of persons inside of buildings, and on the basis of that then design and develop an audiovisual navigation system. This system should be capable of guiding users to any available destination in the given building, while being as passive and least intrusive for the user as possible.

Keywords: indoor navigation, indoor positioning, IPS, WPS, WiFi, RSSI

Abstrakt (CZ)

Cílem této práce je prozkoumat možnosti zaměření a sledování pohybu osob uvnitř budov, a na základě této studie navrhnout a implementovat systém audiovizuální navigace. Tento systém by měl umožňit uživatelům být navedeni k libovolné dostupné destinaci v rámci dané budovy, přičemž by měl být z hlediska uživatele co nejméně náročný na spuštění a používání.

Klíčová slova: indoor navigace, indoor lokalizace, IPS, WPS, WiFi, RSSI

CONTENTS

1 Introduction 1
1.1 Motivation 1
1.2 Structure of this work 2
1.3 Goals of this work 2
1.3.1 Functional Requirements 2
1.3.2 Non-functional requirements 2
1.4 State of the Art 3
1.5 Bachelor Thesis 4
2 Analysis and Design 5
2.1 Infrastructure configurations 5
2.1.1 Server based positioning 5
2.1.2 Client based positioning 5
2.2 Positioning Techniques 5
2.2.1 Trilateration 5
2.2.2 Triangulation 5
2.2.3 Fingerprinting 6
2.3 Positioning technologies 6
2.3.1 WiFi based positioning 6
2.3.2 Bluetooth based positioning 7
2.3.3 Visible light positioning 7
2.3.4 Magnetic positioning 8
2.4 Positioning models 8
2.4.1 Location based model 8
2.4.2 Continuous positioning model 8
2.4.3 Grid based model 9
2.5 Solution Research 10
2.5.1 Passive Tracking 10
2.5.2 WiFi Active Service Discovery 10
2.5.3 MAC randomization 11
2.5.4 User Onboarding 11
2.6 Solution Proposal 13
2.6.1 Visual navigation 13
2.6.2 Localization and tracking technique 13
2.6.3 Client application 13

2.6.4 Pathfinding 14
2.6.5 Handling of multiple users 14
2.6.6 Shades of maximum contrast 14
2.6.7 Failsafe alerts 15
2.6.8 Audio cues 15
2.6.9 Proposed infrastructure 16
3 Implementation 17
3.1 Selection of platform and language 17
3.2 System architecture 18
3.2.1 Components 18
3.2.2 Use Cases 19
3.3 Deployment 21
3.4 Server backend 22
3.4.1 System entities 22
3.4.2 Server REST API 22
3.4.3 Development environment and libraries 23
3.4.4 Server Configuration 23
3.5 FIND Service 25
3.6 Navigator Hardware 26
3.6.1 Controller 26
3.6.2 LED signs 28
3.6.3 Daughter board 29
3.6.4 Enclosure 30
3.6.5 External WiFi dongle 32
3.6.6 Audio Support 32
3.7 Navigator Software 33
3.7.1 Navigator REST API 33
3.7.2 Operating System 33
3.7.3 WiFi configuration 34
3.7.4 LED driver bindings 34
3.7.5 Animations in JS 35
3.7.6 Strip endings addressing 35
3.7.7 Audio Playback 35
3.8 Android Client App 36
3.8.1 User Interface 36
3.8.2 Tracking Service 36

3.9 Management App 37
3.9.1 React 37
3.9.2 Material-UI 37
4 Evaluation 38
4.1 Development Testing 38
4.2 Validation testing 38
4.2.1 Navigation speed testing 38
4.2.2 Error correction testing 39
4.2.3 Multiple users testing 39
4.3 Goals fulfilment 39
4.3.1 Functional Requirements Fulfilment 39
4.3.2 Non-functional requirements 40
5 Conclusion 41
5.1 Results 41
5.2 Further work 42
5.3 Parting note 42
References 43
A List of abbreviations 45
B Installation Manual 46
C User Manual (Android App) 47
D Bill of materials 48
E Daughter board schema 49
F Source files 50
GitHub Repositories 50
Contents of the CD 50

List of Figures

Figure 1: Navigation visual cue principle 13
Figure 2: HSV Color Space 15
Figure 3: Conceptual System Architecture 16
Figure 4: Use case diagram 19
Figure 5: User registration sequence diagram 20
Figure 6: Deployment Diagram 21
Figure 7: Completed prototype 26
Figure 8: ESP8266 based NodeMCU board 27
Figure 9: Raspberry Zero W (Raspberry Pi Foundation, 2017) 27
Figure 10: LED sign shapes 29
Figure 11: Daughter board detail 29
Figure 12: Enclosure rendering 31
Figure 13: Navigator unit without enclosure 31
Figure 15: Assembled navigator unit in printed enclosure 32
Figure 14: Navigator unit during assembly 32
Figure 16: Navigation screen 36
Figure 17: Destination picker screen 36
Figure 18: navsys-manager interface 37
Figure 19: Daughter board schema 49

List of Tables

Table 1: Server client API 22
Table 2: Client Navigators API 23
Table 3: Client Locations API 23
Table 4: Navigator REST API 33
Table 5: Navigation speed testing 38
Table 6: Error correction testing 39
Table 7: Bill of Materials 48
Table 8: Contents of the CD 50

1 INTRODUCTION

Global positioning systems and their widespread availability in our smartphones revolutionized ways how we navigate around. Anytime we need to find a way to a place we haven't been to before, we just pull out a phone of our pocket, type in the desired destination and we immediately know which way to go. There is a problem with these systems though - they don't work inside of buildings. Yet there are many buildings big and complex enough - like hospitals, schools, or offices, which are hard to navigate in for newcomers, and lot of times these are the buildings which most of the visitors visit only occasionally.

Aim of this work is to research options available for indoor positioning and navigation, and develop a navigational system which could be deployed in buildings with complex corridor network, helping visitors find their destinations easily. This navigation system shall be audiovisual, physically present and integrated into the corridors themselves, so the users would be able to use it without the need of using their phones during the navigation.

1.1 MOTIVATION

As stated, global positioning systems have a major drawback, which is that it does not work well in enclosed spaces. One of the reasons for that is dampening of the signal caused by the walls. The other is, that the GPS uses for calculation of location time of flight principle, and being enclosed by wall creates multiple reflections, which causes multipath propagation of the signal, because of which problems with correctly determining the time or direction of arrival of signals arise (Pourhomayoun, Jin, & Fowler, 2012).

Because of that, numerous other approaches were tried and tested to provide indoor positioning, using varying physical and fundamental principles, but none has panned out yet to be the ultimate one way to go. Though in isolated cases indoor localization and navigation systems were already deployed, most public buildings are still relying on printed signs and maps to show people the way around. While they are in most cases able to lead visitors to their destination, it usually takes them much longer on their first visit than when they already know where are they going. That is the point where smart navigation systems could help, as they might lower the time people need to find their location to almost the same as if they would already know the way.

1.2 STRUCTURE OF THIS WORK

This work will be structured in the following way:

1. In the introductory part goals of this work will be defined, current state of the art in this field will be discussed, and finally a comparison with my bachelor thesis will be made
2. In the second part techniques and technologies for indoor positioning and navigation will be researched and solution will be proposed for the given implementation problem
3. In the third part realization of the navigation system will be described, all used technologies will be overviewed and problems which were dealt with will be discussed
4. In the fourth part the developed system will be evaluated, and testing of its functionality discussed
5. In the last concluding part, the results of this work will be described, and further works and future possibilities discussed
6. In the appendices at the very end of this document will be located schemas, manuals, links to source codes and other related materials.

1.3 GOALS OF THIS WORK

First goal of this work is to analyze and research technologies which are currently being used for indoor positioning and navigation purposes and select from them one which would be the best fit for further proposed navigation system.

The other goal is to implement the proposed navigational system. From the project assignment and consultations with the thesis supervisor a following set of functional and non-functional requirements for this system was defined:

1.3.1 Functional Requirements

Create a navigation system which will satisfy following functional criteria:

- System will guide users along the shortest path to their destination
- System will guide users using visual cues present in the physical world
- System will enable users to choose any destination available
- System will be capable of using audio output to provide navigational cues and entertain users
- System will be able to guide users back to their correct route should they take a wrong turn

1.3.2 Non-functional requirements

Additional non-functional requirements were set:

- System will be capable of guiding multiple users at the same time
- System will be remotely configurable
- System will be horizontally scalable
- System will be extensible
- System have to be capable of continuous operation
- Deployed hardware will use wireless communication

1.4 STATE OF THE ART

While several systems of global positioning systems were already deployed and are active, which enables precision tracking of persons, vehicles or goods anywhere outside around the world, they have problems working indoors. Having access to outdoor positioning systems helps individuals navigate around the city, or reach their destinations when travelling by cars, but today most of the time people spend indoors. The newest global positioning system Galileo is actually capable of providing indoor localization (European Global Navigation Satellite Systems Agency, 2017), but in most cases the precision is still in order of magnitude of tens of meters.

Partially because of the localized nature of the indoor positioning problem, and multiple technologies available to solve it, there are now several competing approaches, none of which is currently being a clear leader. Most solutions use technologies which are not locating the actual user, but the phone he is carrying, as are predominantly WiFi and Bluetooth. Both of these technologies are being used in a client based or server based settings, when the phone acts as a receiver or transmitter, respectively (infsoft GmbH, 2016).

Technologies of image based recognition could be used for tracking the actual user himself, but are harder to use than when tracking the phone. User has to be identified first as the object being tracked, and this identification has to hold as he leaves and enters field of view of different cameras.

Interesting is the possibility of using ambient magnetic fields for the task of positioning as well (Han-Sol Kim, 2017), even though the solution proposed needs an encoder to keep track of current velocity.

Most of market available solutions are implemented as a smartphone map app, though again not yet any one solution has prevailed against the others and got widespread adoption. Though now there are Google Indoor Maps[1], which are getting integrated into the regular Google Maps[2]. This might be the closest we get to a widespread generic solution, unfortunately they are not yet available to use or test in Czech Republic,

Nice example of large scale deployment can be seen at Swiss railways Zurich station, where more than a thousand Bluetooth beacons were deployed (infsoft GmbH, n.d.), and provides a complete navigation solution over all of the station premises. This solution uses a native smartphone app, which has to be installed for the user to be able to use this system[3].

[1] https://www.google.com/maps/about/partners/indoormaps/
[2] https://www.google.com/maps/
[3] https://play.google.com/store/apps/details?id=ch.sbb.oevnavi&hl=en

1.5 BACHELOR THESIS

This work is a loose continuation of my bachelor thesis (Jedlička, 2013). While title of that thesis was simply 'Light chain control', the work was already aimed at using LED strips as part of an audiovisual navigation system. In that work hardware navigation units based on Arduino platform were created, which could activate each other when a presence of human - visitor was detected using a PIR sensor. However, these units were meant to be placed only along a single path - from entrance of the Faculty of Electrical Engineering to the Institute of Intermedia, so that system provided way only to a single destination. There was no backend server - no centralized control which could affect which units should light up and when, and there was no other mechanism of user detection than the PIR sensor, so individual user tracking was not possible and the navigation was not personal.

While this work will be building upon the same general idea, nor code or hardware from the bachelor thesis will be reused, only the idea of audiovisual navigation and usage of digital LED strips. It is because this time the goal is much broader - a personal navigation to any destination the user picks. That leads to a need of developing and deploying a far more complex system. The user has to be tracked, so the system can decide at each intersection where to lead him next - that tracking information has to be obtained somehow, user's path must be stored somewhere, and all of that acted upon in order to light up the right units navigating towards the right destination. Where in my bachelor thesis it was sufficient to develop the navigation units alone, this time they will be only a single part of the whole system architecture.

2 ANALYSIS AND DESIGN

In the second chapter will be analyzed and researched various techniques and technologies related to the indoor positioning and navigation problem. Next a solution for the problem of navigation system implementation will be researched and proposed.

2.1 INFRASTRUCTURE CONFIGURATIONS

There are several technologies available for positioning, where the ones which are being majorly used are wireless. In any kind of data transfer there are two participants of th at communication – a transmitter and a receiver. Similarly, there are two complementing configurations used for infrastructure of positioning technologies - server or client based positioning.

2.1.1 Server based positioning

In server based positioning, the user's device acts as a beacon and transmits identifiable signal. This broadcast is then captured by the receiving infrastructure, which has to be installed in the observed space, and position of that device is then calculated from the signal strength captured by individual receivers and known fixed position of those receivers.

2.1.2 Client based positioning

In client based positioning, the roles are reversed. The installed infrastructure is broadcasting and user's device captures broadcasted signals. Then the user's device either calculates its position on its own, if the tracking application contains positions of the transmitters, or sends the captured signal strengths back to the tracking system for processing.

2.2 POSITIONING TECHNIQUES

2.2.1 Trilateration

Trilateration is a process of finding the position by measuring the distances of points using geometry of circles or spheres. With wireless technologies, a signal strength is being used for calculation of the needed distances. Reason, why we can use signal strength for trilateration, is based on the fact that radio waves propagate according to the inverse-square law, so the travelled distance can be approximated from the relation between transmitted and received power.

There is a problem with long range trilateration inside of buildings, because in such environment the signals are not propagating in a free space. Because of that a lot of random reflections or absorptions from walls occurs, which can alter the signal strength in unpredictable ways. Short range signal sources can be used to remedy this, but they'd have to be placed in a very dense arrays, which could make deployment of the positioning system prohibitively expensive. Because of that the signal strength readings are usually being filtered using various statistical procedures such as Kalman filtering (Chui & Chen, 2009).

2.2.2 Triangulation

Triangulation is a process of finding the position by forming triangles to the position from known points. In positioning technologies, the triangulation is used when we are able to calculate the Angle of Arrival of the signal. Angle of Arrival can be determined by using an array of multiple sensors. These can be highly directional - so at a given angle only a specific sensor from the array

would receive the signal. With this approach to get a high precision a very large amount of sensors would have to be used. Other possibility is measuring the time difference of arrival. In that setting all sensors of the array receive the signal, but being in a slightly different location, the same signal will be reach the individual sensors in a different moment.

Triangulation is limited to be used in a server based positioning, as there is no smartphone equipped with such a sensor array to be able to resolve from which direction the signal was received.

2.2.3 Fingerprinting

Fingerprinting technique is an empirical method, when all of the locations which the system should be able to distinguish are catalogued, and each of them is paired with a unique identifier – a fingerprint.

The position is then calculated from a fingerprint of an unknown location using algorithms such as k-nearest neighbor (Altman, 1992), which will determine which is the most similar fingerprint from the original set to the one presented. A more advances approach is to use machine learning classifiers such as Naïve Bayes (Rish, 2001), which would be first trained using several fingerprints for each of the locations. That makes the classification of the unknown sample more robust. It is especially effective for use with wireless technologies, as the gathered fingerprints tend to contain a lot of noise.

2.3 POSITIONING TECHNOLOGIES

Today there are several technologies used for indoor positioning and tracking, each of which has advantages in some areas and lacks in others. The two most often used are WiFi and Bluetooth, complemented by others like visible light communication (VLC) or magnetic field based positioning.

2.3.1 WiFi based positioning

WiFi based positioning or WiFi positioning system (WPS) is based on the received signal strength indication (RSSI), which can be processed in two fundamentally different ways.

One approach is to check the RSSI levels for MAC addresses from which the signals originated, and if there are levels of APs of which the physical location is known, location can be calculated using trilateration principle.

The other approach is to use the fingerprinting technique, where the captured RSSI levels and MAC addresses of the transmitters are used as a fingerprint, which can then be used to identify that particular location.

2.3.1.1 Properties

Benefits of WiFi based positioning lays mainly in its long range, therefore expanding the area in which tracking can be done. Another benefit is that in the client based positioning there is a possibility of utilizing infrastructure that is already in place for other purposes. In developed residential areas, it's hard to find a place where are no WiFi networks present. Downside of utilizing existing independent networks is in the lack of control. At any time in the future the infrastructure of APs can change. This can be partially solved by using RSSI only from trustworthy APs which are unlikely to change much, like a university WiFi networks.

The positioning itself is less accurate compared to other methods. There are multiple factors present that have impact on accuracy. Firstly, the quantity of available access points and their

strength. Another aspect is the material structure of the location, reflections of signal and shielding through walls which can all negatively affect accuracy, especially when using trilateration technique.

2.3.1.2 Client based

For a client based WiFi positioning it is required to have an app installed on the device, which will take care of scanning for nearby APs and location calculation or sending of the fingerprints to the tracking service. What sets this case apart from all the others, is that the location can be acquired by scanning already existing access points in that location. That means that in a client based scenario, there is not no need for installing additional infrastructure the WPS.

2.3.1.3 Server based

For server based WiFi positioning however, app might not be used at all. All WiFi enabled devices are over the time broadcasting 'probe packets', which help actively discover available networks. Exploiting these, we can in server based scenario passively position and track all WiFi devices in the observed area. This way of tracking is however not precise, as the probe requests are being sent out irregularly, and also it is inherently anonymous, unless it would be known to which devices does the observed MAC addresses belong.

2.3.2 Bluetooth based positioning

Generally, all of the principles and approaches of WiFi positioning are valid for use with Bluetooth technology, as it provides RSSI and MAC information as well.

2.3.2.1 Properties

Bluetooth though suffers from a shorter range than WiFi. On the other hand, because the deployed infrastructure has to be denser due to the limited range, the positioning has a higher precision than a usually more sparsely deployed WiFi. Disadvantage when compared to WiFi client based positioning is, that the existing infrastructure cannot be used - simply because there usually isn't any.

2.3.2.2 Client based

In this setting beacon transmitters have to be distributed across the monitored space and a user's phone then acts as a receiver. The transmitters can be a more complex devices having Bluetooth capabilities - such as Raspberry Pi, which can be configured to publish iBeacon or Eddystone packets. Or it could be specialized Bluetooth beacons, which are basically microcontrollers specifically set up for the single purpose of broadcasting of beacon packets.

2.3.2.3 Server based

In this case the user's phone is configured to transmit the same beacon packets, or a beacon keychain could be given to the user. Across the monitored space Bluetooth receivers has to be placed, which would monitor the beacon devices to be positioned.

2.3.3 Visible light positioning

Visible light positioning is a client based positioning, which utilizes LED sources, that are set to intentionally flicker in a unique pattern which can be used as their ID. These light sources are then installed instead of regular lighting in the locations which should be distinguishable. The system is then able to resolve the position by decoding the ID of LED source which is illuminating the smartphone's camera.

2.3.4 Magnetic positioning

Magnetic positioning is another client based positioning technique, which supposedly utilizes compass chip with which most of smartphones are equipped with to gather sensor data, which then can be used for positioning. There is only one commercial closed source solution using this approach - IndoorAtlas[4], and the only paper where using ambient magnetic field for absolute positioning has been described was written by the founder of IndoorAtlas. It is questionable of how this localization technique stays stable in a long term, as the Earth's magnetic field is gradually changing. It is also questionable how abrupt changes in the ambient magnetic fields are being handled, for example when metal objects like lifts move around, which furthermore also utilize electromotors and magnets, deepening the problem.

2.4 POSITIONING MODELS

Apart from the technology used for obtaining data there are several fundamentally different approaches how to model and represent user's location.

Two approaches that are used the most are the discrete location positioning and continuous spatial positioning, accompanied with a somewhat hybrid approach - grid system.

2.4.1 Location based model

The location based model is an empirical method, using a fingerprinting technique, when we populate the system with a given set of discrete locations and their fingerprints.

Unless there is absolutely no match between the given sample fingerprint and those from the original dataset, this system will always return the closest match.

This system is useful, if we are not actually interested in user's precise location, but only the closest place from the given set. Problem of this technique is, that the fingerprint data, which the user could potentially provide from a physical location in between the ones in the dataset, might be a linear mixture of the closest two physical locations. There is a possibility that the nearest known neighboring fingerprint might be actually from a completely different, much farther location.

These false positives might be prevented by providing a location graph to the system, which with information about the last vertex where user's location was confirmed, can be used to limit the subsequent locations to the adjacent vertices of the that last known vertex.

For this system it doesn't matter whether the data originated from short range source like Bluetooth or long range WiFi signals, or whether they got reflected along the way - as long as there is enough of unique data for each of the indexed locations.

2.4.2 Continuous positioning model

Continuous positioning model is quite the opposite of the location based approach. In the existing solutions it is usually referred to as a 'blue dot' navigation.

Instead of using a set of discrete locations, we are trying to calculate user's position on a two-dimensional plane, or even in a three-dimensional space and track his movement continually.

[4] http://www.indooratlas.com/

For this we usually rely on a long range sensor data, and either their signal strength or time of arrival for a trilateration calculation, or their angle of arrival for triangulation.

2.4.2.1 Dead reckoning

Dead reckoning is a method helping to provide the continuous positioning, even when only discrete location data are available. It is a process of calculating the estimated current position from a previously known one, adding to it the estimated bearing, speed, and time passed. It is then when providing continual tracking used to interpolate the current position, until the next actual position gets determined.

2.4.3 Grid based model

Grid based model is a sort of a hybrid between the previous two. Same as with the continuous positioning the goal is to find a position in a multidimensional space, but in this case the space is divided into an indexed grid, and based on the available data the positioning system is trying to determine which cell is the current position (Wutjanun Muttitanon, 2007).

For this model it might be possible to use both fingerprinting or trigonometry based positioning techniques, but for high resolution it would be especially helpful in this case, if the data were provided from a dense network of short range signal sources. Because signal strengths can be highly erratic indoors, the short signal range would ensure that signals can't be received in distant cells, and at the same time strong signal readings will be obtainable only in directly adjacent cells.

2.5 SOLUTION RESEARCH

To select the most viable technology we have to take into account different factors as are characteristics of the space we want to navigate through, specifics of the users, our use-case and last but not least cost of the hardware, and amount of work needed to develop and set up the whole system.

Hand in hand with selection of technology goes the selection of the hardware. Not only because of the cost, but also because of different capabilities and limitations of different platforms.

Our use-case is navigation of visitors through corridors of Faculty of Electrical Engineering, to get them to their desired destination. We want to achieve this using system of physical navigational units, which will lead user to the next checkpoint using audiovisual cues. System of corridors in this building is not dense, but corridors tend to be quite long and there are multiple doors which the visitor might have to go through, not always looking inviting. Most of these doors tend to lead into specialized departments, and it is not quite obvious there could be classrooms inside.

In these corridors we want to place navigational units at the wall corners at intersections, to signalize to the user using LED based signs mimicking in their shape the intersections they are being placed at, which way he should continue.

While shapes of these units in X or T junctions seems obvious, aside from corridors there are also several lobbies - open spaces, through which the user has to find his way as well. Placement of the units in these situations might prove tricky.

What we need to detect is, when the user gets close to one of these checkpoints - hallway junction of some kind.

For this use-case it seems unnecessary to perform continuous spatial localization, as we are not interested much in where exactly user actually is. As he is not navigating in open space, it should be enough if his location can be verified at each intersection.

2.5.1 Passive Tracking

First idea was to design a solution which would track user passively without the need of installation of any client application on his mobile handset. User would walk through the entrance and get a notification of a physical web node containing the URL of an onboarding web app. In this Web app he would just choose a desired destination, and that would be it. System would then passively track location of his phone, and accordingly navigate user with the use of navigator units.

However, this idea, while technically possible fell short because of smartphone OS restrictions, as it does aim at what should not be possible - passively track the device, without any obvious user's consent.

To limit possibilities of passive tracking, smartphone operating systems - both iOS and Android started to impose several different restrictions.

2.5.2 WiFi Active Service Discovery

This passive tracking was generally possible because the smartphone from time to time refreshes state of available networks. While he could listen for beacon advertisements of AP's in range, it would take a while before he would collect all of them - wasting battery life in the process. So to accelerate the WiFi state readout the phone does so called active discovery

lookup - the phone advertises itself as looking for networks, and all the AP's in range responds with their beacon advertisements immediately.

This active discovery lookup is a public broadcast, so it can be 'heard' by all active WiFi radios in range. Interesting fact for the tracking case is, that the packet also includes a MAC address of the broadcasting device. MAC address as a unique identifier can be used to link separate lookups to a single device and in such a way tracking of that device can be achieved.

2.5.3 MAC randomization

To circumvent this, starting from Android version 6 Marshmallow and iOS version 8 (Mathieu Cunche) should've started to use MAC randomization - sending out a different fake MAC address with each discovery lookup request. This measure alone doesn't completely prevent the tracking to happen though, as for example the device usually broadcasts several of these requests in succession, so a timing attack can be used, to identify the device even though the MAC address cannot be used anymore.

Successful test has been made using ESP8266 set into promiscuous mode intercepting these discovery requests. Tests were done using devices running Android 6, and another running iOS 10, and in both cases it has been found out, that the MAC randomization wasn't actually being employed. But anyway since both systems should be using this technique by their documentation, it is not viable to count with using this method in the future. Another problem of this approach is, that the device usually does this lookup only once in a while, and if this navigation system should be robust, it can't rely on coarse data which are being sent out in an unspecified interval.

But even if the MAC randomization was not employed, and the requests were being sent out in a steady rate, the system would still need to know the MAC address of user's device beforehand, so it'd know which MAC addresses to track and to which user they belong.

And access to the device's MAC address is not only impossible from the browser environment, starting Android 6 even native apps cannot read it.

Because of these reasons, the idea of using passive tracking was abandoned, and it was agreed that user will have to install an application on his device to use the navigation

2.5.4 User Onboarding

Since the first idea of the user onboarding with the help of web application was rejected, different options had to be explored.

Even though not original in any way, still the most common and reliable approach is to make use of a native application, which has the access to WiFi state, can keep the device running, and provide an interface streamlined into the phone, rather than having to use the browser.

Another idea though of how to onboard the user into the system was, that on the intersections could be placed informational panels with lists of destinations and corresponding buttons. User would press a button, and the navigation would light up to show the way. This way though there wouldn't be any tracking possibilities, so the system would only show the way to the next intersection, where the user would have to pick a destination again.

2.5.4.1 Beacons

Not only smartphones can be tracked. Bluetooth beacon technology exists, where the phone can be configured to act as a beacon, but standalone Bluetooth beacon fobs can be bought as well. Handing out of these beacons was ultimately denied in the end, as the probability of not getting them back from the visitors is very high, and the cost for their often replacement would be not acceptable

This schema would on the other hand probably work well with employees, as they could have the Bluetooth fob for example attached to their ID and the probability of theft would be comparatively very low. Though on the other hand for navigational system to make sense it would have to be a very large and complex building, as otherwise employees would learn the ways around fast and won't be in need of having a navigational system anymore.

2.6 SOLUTION PROPOSAL

2.6.1 Visual navigation

For providing the visual navigation, digital LED strips will be used. Navigation units shall be placed at the intersections, and from these strips LED signs in the geometric shape of given intersections will be created – for example X or T shapes. Users will be then navigated by visual cues created by activating only arms of these shapes corresponding to the path user should take, animated in the direction the user is going.

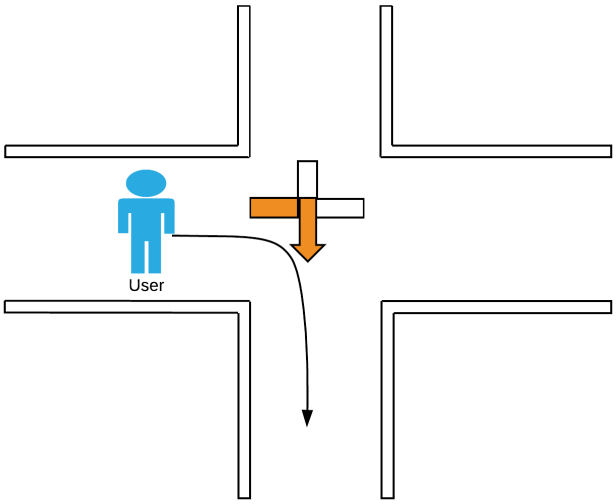

Figure 1: Navigation visual cue principle

2.6.2 Localization and tracking technique

After evaluating all the researched techniques, it was decided to use WiFi client based fingerprinting.

Fingerprinting provides a robust discrete location detection, and is not as complex to develop as the continuous localization. At the same time, it perfectly fits onto the model of location graph in which the navigation units will be displaced, where the continuous location tracking is not needed, as the activation of navigation units will be discrete in nature anyway.

The reason of using WiFi is, that we can try to use for fingerprinting the existing AP infrastructure, and compare it to using only the AP created by the navigation units.

2.6.3 Client application

Because of the client based approach coupled with the need for providing some user interface, a mobile app should be created. Downside of the selected approach is, that it can't be used on iOS devices, as iOS limits access to RSSI information. Thus the client will be only Android application at first.

The selected approach is though compatible with Bluetooth beacons, as they provide the same kind of information – UUID and signal strength, and iOS grants access to the Bluetooth RSSI. So later an iOS application could be created as well, in case Bluetooth beacons would get positioned throughout the navigable space.

2.6.4 Pathfinding

An algorithm is needed for finding a way from the user's current location to his desired destination.

Proposed solution is to represent locations as distinct nodes in a graph, whose edges have weights set according to the real world distances between those locations. This allows to solve this problem as a regular shortest path in a graph problem.

2.6.4.1 Dijkstra's algorithm

An algorithm invented by a computer scientist Edsger Wybe Dijkstra, widely used for solving the shortest path problem.

This algorithm works by performing a breadth first search through a graph, starting at the node where we want the path to begin. At each node it visits it also notes how long is the distance - the sum of all edge weights crossed - to this node. If the sum weight it lower than the one currently marked for this node, it lowers the value to the better one and stores with it the path taken. After visiting all the neighbors of the current node, it marks it as closed, so the neighbors at next level won't search again through these already processed nodes. When we want the shortest path only to a given destination, the algorithm can stop after closing the destination node. Otherwise if left running it will calculate the shortest path to all nodes of the graph from the starting point.

2.6.5 Handling of multiple users

Since the system needs to be able to navigate multiple users at the same time, and different nodes can be passed at the same time by users with different destinations a user friendly visualization is needed. Each user as he selects a destination automatically gets assigned with a color. This color is shown in mobile app and then used in animations at each node. There is still a probability of two users arriving at the same node simultaneously. Clear distinction of correct path of each users is ensured by color and therefore animations of each path can be shown on one node in two ways. Either blend the two colors, using the individual LEDs in strips, or periodically switch different animations in time cycles. The latter was chosen, for clearer distinction and to avoid confusion. With user accessibility in mind, colors needed to be with maximum contrast and time cycles optimized.

2.6.6 Shades of maximum contrast

When researching an algorithm for generating the most contrasting colors, it soon became obvious that such task is not as easy as it seems. First idea how to solve this, was simply using HSV system to take any default hue with maximum saturation and value and rotate it by a given amount of degrees.

Problem with this approach is, that the perceived difference of hue is not the same when changing the hue by equal steps. When changing a value of hue by step of 30, value 50 and 80 are one orange and the other lime, while 110 and 140 are both a slightly different shade of green.

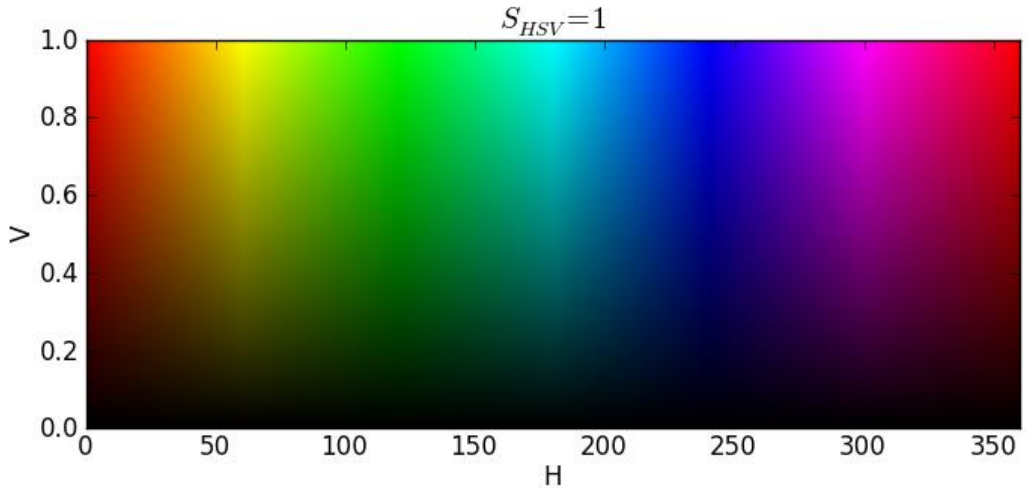

Figure 2: HSV Color Space

Solution I've chosen to use is to pick colors from a list of 22 predefined colors by Kenneth Kelly (Green-Armytage, 2010), which are designed to have a maximum perceived contrast possible.

2.6.7 Failsafe alerts

Not only signals indicating the right way, but also those giving a wrong way feedback improve user's ability to successfully reach his destination. When user arrives at a wrong node two messages are communicated. First one is that of wrong way, represented by red color, a visual cue easily decoded. Because user needs to be informed not only about an error, but also provided with an alternative solution, error message animation is (in a similar way as if two users arrive at same node) alternated with signal returning user to his correct path.

2.6.8 Audio cues

Audio cues should be used not to substitute, but only supplement visual signals. Mainly for handling user alerts of wrong way and to make user experience more pleasurable, implementing commentary, music at special events and others.

2.6.9 Proposed infrastructure

In the proposed architecture, WiFi fingerprints are collected on user's smartphone from the surrounding access points. Then they are send over to the server, which should evaluate the data and calculate users position. According to this location, server should then activate corresponding navigator units.

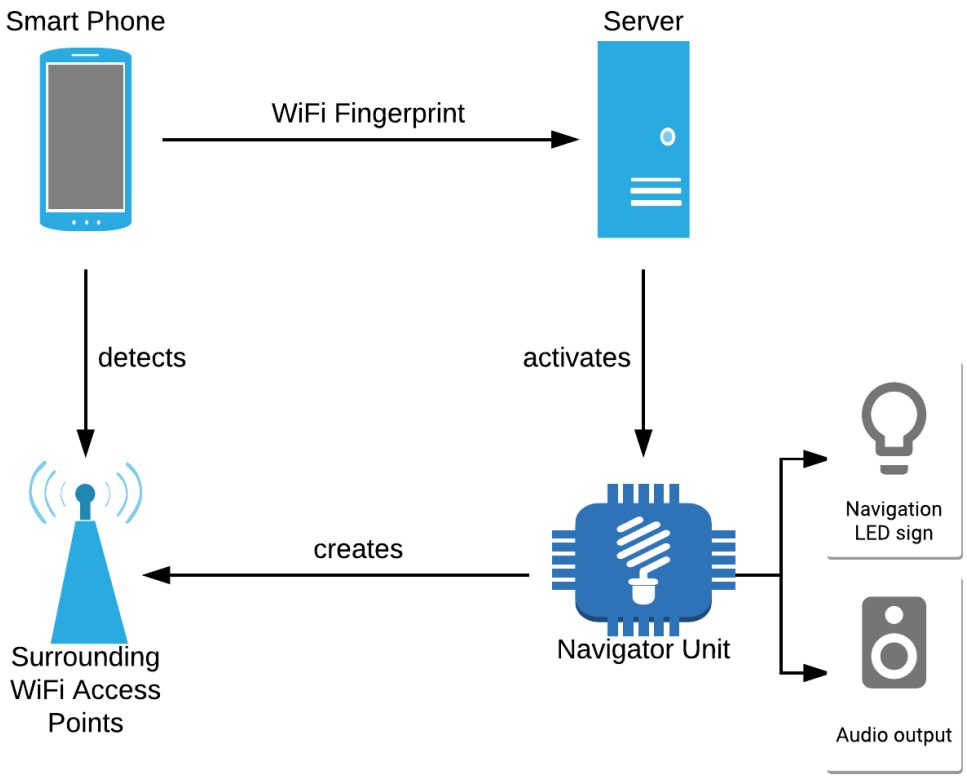

Figure 3: Conceptual System Architecture

3 IMPLEMENTATION

In this chapter will be presented information about how the system was developed. First is a summary of the components which makes up this system and how they relate. I'd like to detail how are all of these components deployed, and how they interact with each other. After that a detailed description of each component will follow.

3.1 SELECTION OF PLATFORM AND LANGUAGE

Most of the components of this system were written in JavaScript, including the backend server and navigator control software.

While JS is for most the language of choice for web applications, backend and embedded software is today still being mostly written in different popular languages like C, Java, Python or Ruby. However, since the Node.js - JavaScript runtime which will be later discussed further - runs now not only on x86, but on ARM processors as well, it is possible to use JS in all parts of the technology stack, including the embedded hardware.

Thanks to React Native - a JS mobile development cross platform framework - it would have been possible to write the Android application in JavaScript as well. For now, the choice was made not to, so the Android client is the only component written in a different language, which is its native Java.

Though if the need for iOS application would arise in the future, as discussed in the concluding part of this work, it might make sense to rewrite the mobile client using React Native, and have all of the components which makes up this system written in JavaScript, a feat impossible not a long ago.

3.2 SYSTEM ARCHITECTURE

The generalized way how the system should operate has been described in the previous chapter. Essentially it is a client – server architecture, where communication is being made using REST interfaces.

3.2.1 Components

There are five key components which makes up this system, titles in braces being their corresponding GitHub repository names:

Backend server (navsys-backend): Backbone of the whole system, Express.JS based server acting as an intermediary for all the other components.

FIND server: Partner of the backend server, open source service providing machine learning based API for WiFi fingerprint identification.

Navigators (navsys-navigator): Navigation units powered by Raspberry Zero W, equipped with digital LED strips. Based on Express.JS as well, they are exposing a REST interface for accepting commands.

Android Client App (navsys-client-android): The user facing component of this system, which provides interface for selecting desired destination, and collecting WiFi fingerprints for localization needs.

Management frontend (navsys-manager): Simple web application written in React, created for purpose of easy system administration.

3.2.2 Use Cases

Following is the diagram showing use cases which the system supports:

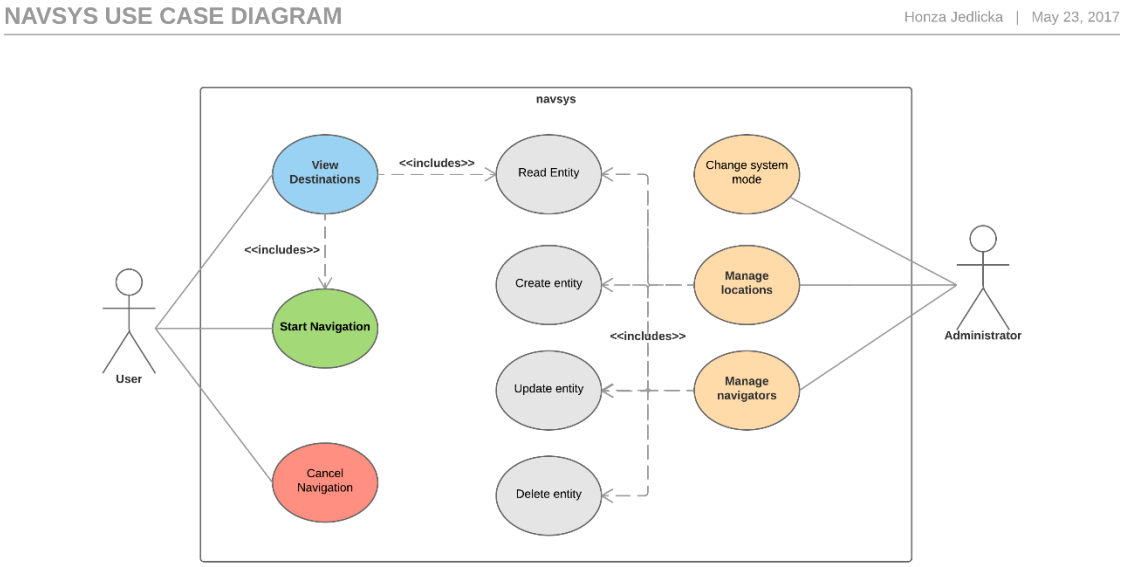

Figure 4: Use case diagram

To illustrate how does the system operate, I'd like to explain it on the use cases of how users register to the system, and how does the system navigate them.

3.2.2.1 User session registration flow

1. When user wants to be navigated somewhere, he first has to choose a desired destination in the client application. When that happens, the client application sends a registration request to the backend server with a user's ID, the chosen destination and WiFi fingerprint of his current position.

2. When server receives the registration request, it sends the WiFi fingerprint to FIND service for location identification.

3. FIND service runs Naive Bayes analysis on the received fingerprint, and sends to the backend the closest match.

4. When backend gets back the location analysis results, it then uses returned location for calculating a shortest path between that location and requested destination.

5. Backend then activates the navigator unit corresponding to the user's current location, navigating in the heading of next location in the calculated path. At the same time, it activates the navigator at the next location as well, which will display way to the third location.

6. At last the backend assigns available color to this user session, and returns it to client application with the calculated current location

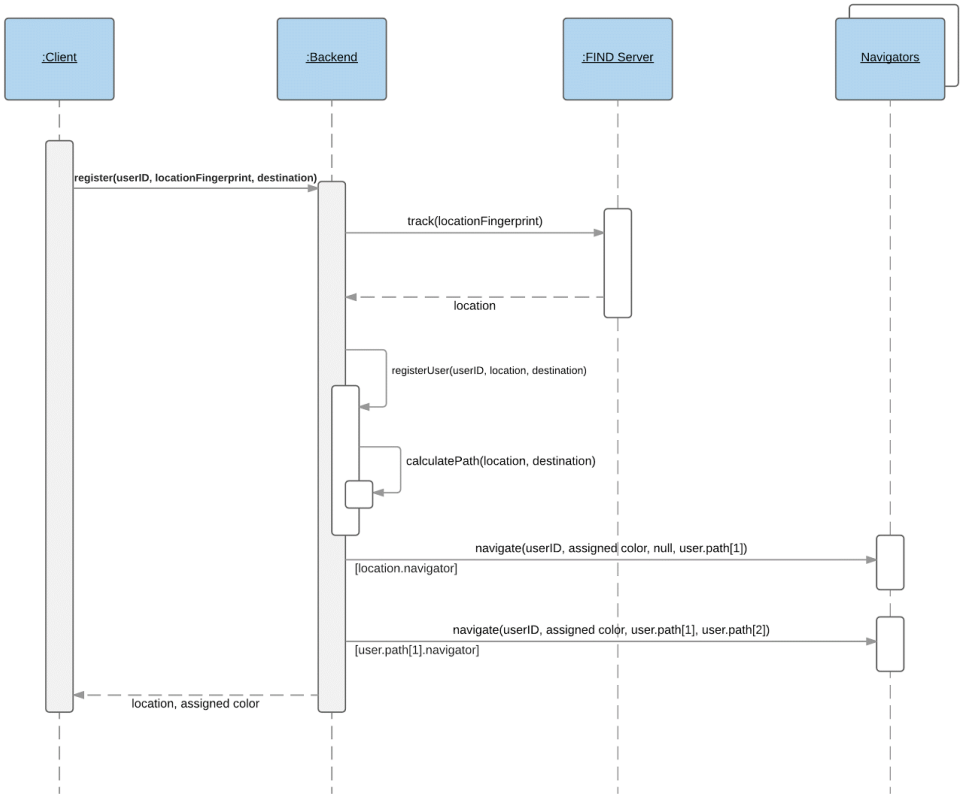

Figure 5: User registration sequence diagram

3.2.2.2 User tracking flow

1. Mobile application starts scan for surrounding WiFi networks. Resulting WiFi fingerprint is sent to backend server's 'track' endpoint.
2. When server receives track request, it resends it to FIND service for location identification.
3. FIND service runs analysis on the received fingerprint, and sends back the closest match.
4. When backend gets back the identified location, it sends it to its track handling method, which decides if there should be any change in active navigators or not

a. If yes then, according navigators get activated or deactivated.

b. If not, then there is no change, location only gets pushed to user's location history.

5. Calculated location is send back to the mobile client

Sequence diagram of tracking flow would follow the same pattern as the registration.

3.3 DEPLOYMENT

All the components communicate using REST interfaces. The following deployment diagram illustrates how all of the system components are deployed:

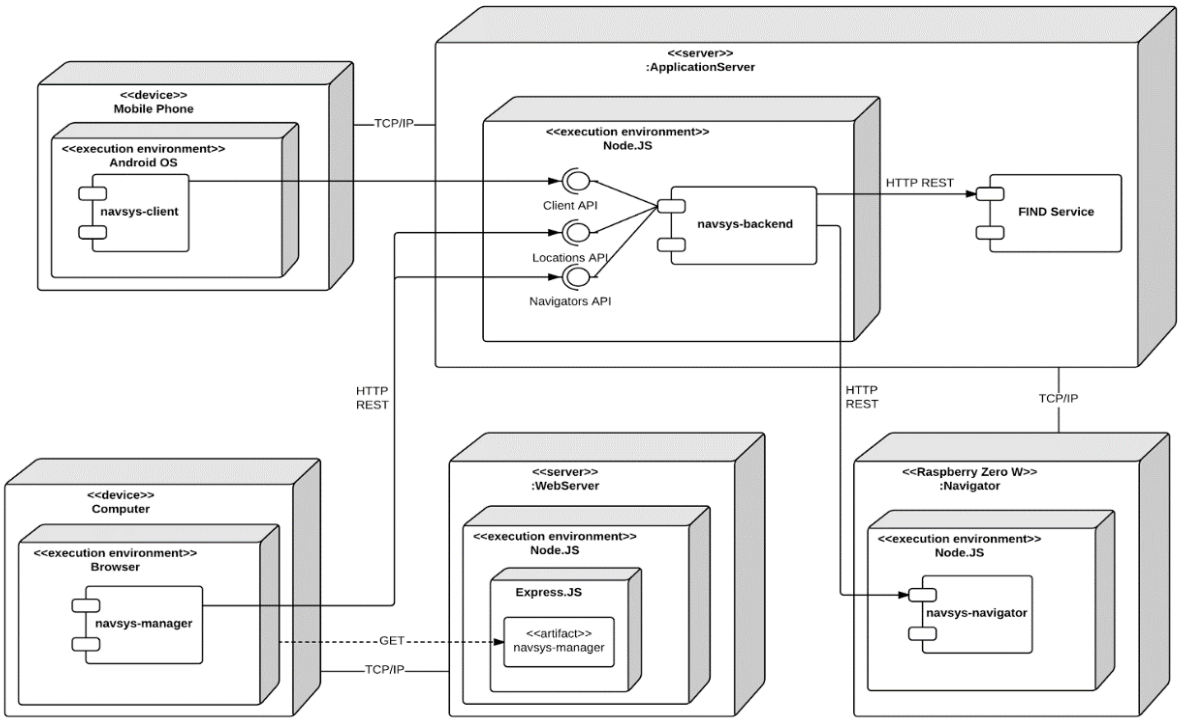

Figure 6: Deployment Diagram

3.4 SERVER BACKEND

Primary purpose of server in this system for supporting the user navigation - tracking users along their way, managing their location and activating corresponding navigator units.

It is written in JavaScript for Node.js platform using Express.js framework.

It provides REST API for mobile client and CRUD REST API for the management server.

3.4.1 System entities

3.4.1.1 Navigator entity

Navigator entity holds only information needed to control the navigator units – their IP address and mapping of the endings of their LED signs to the location they are neighboring with. Additionally, system stores a MAC address of the AP they are broadcasting, so e.g. localization using only the navigator AP RSSI's can be tested.

3.4.1.2 Location entity

Location holds the information needed for identifying the given location – its name and ID, flag if the given location is a destination or not, and the list of neighbors with real world distance to them.

Because of the neighbors list, the locations act as a double linked list, and can be used for graph searching.

3.4.1.3 User entity schema

User entity does not hold information of a user profile, but rather a single user navigation session. It stores information about the origin and destination of that session, shortest calculated path for the user to follow, and a host of data used by the system for tracking – previous location history, current progress in the path and list of navigators which are active for that user.

3.4.2 Server REST API

3.4.2.1 REST

Representational state transfer is today one of the most common ways of providing interoperability between systems over computer networks. It is used for publication of Web Services, which it maps to HTTP methods as are GET, PUT, POST or DELETE.

3.4.2.2 Client facing REST API

Method	Endpoint	Description
POST	/track	The primary endpoint for user navigation. Client posts through this endpoint WiFi fingerprints
POST	/learn	Endpoint for learning new locations. Currently only resending requests to FIND service
POST	/register	Using this endpoint, the mobile client registers for tracking session
POST	/cancel	For cancelling of currently running tracking session. Shutdowns active navigators and clears the session

Table 1: Server client API

3.4.2.3 Entity REST API

This API is used primarily for the management application, but client use it as well for getting the list of destinations.

3.4.2.3.1 Navigators

Method	Endpoint	Description
PUT	/navigators/:id	Creates a new navigator entity or rewrites existing with the given ID. The IDs of navigators are constructed from their corresponding location IDs.
GET	/navigators/active	Gets the list of all active navigators. For dashboard purposes.
GET	/navigators/:id	Gets a navigator by its ID
GET	/navigators/	Gets an array of all navigator entities
DELETE	/navigators/:id	Deletes a navigator entity by its ID

Table 2: Client Navigators API

3.4.2.3.2 Locations

Method	Endpoint	Description
PUT	/locations/:id	Creates a new location entity or rewrites existing with the given ID. The IDs of locations are created from the location name.
GET	/locations/destinations	Gets an array of locations which are marked as destinations. Used by client for displaying the destination list.
GET	/locations/:id	Gets a location by its ID
GET	/locations/	Gets an array of all location entities
DELETE	/locations/:id	Deletes a location entity by its ID

Table 3: Client Locations API

3.4.3 Development environment and libraries

3.4.3.1 Node.js

Node.JS is a popular open-source cross-platform runtime environment for JavaScript, built on V8. Thanks to V8, it is performant enough to be used even as a base for production servers. Thanks to this runtime, it is now possible to run JavaScript code outside of the browser, making the way for servers or desktop applications written in JS.

3.4.3.2 Chrome V8

V8 is a JavaScript execution engine, developed by The Chromium Project primarily for use in Google's Chrome internet browser. V8 compiles the JavaScript source code to native machine code, rather than interpreting it in real time, and then additionally optimize it.

3.4.3.3 Express.js

Express.JS is a web application framework for Node.js. It's still today the most popular framework for building of REST APIs in Node, even though there are other popular as well as is Connect, Koa or Restify.

3.4.4 Server Configuration

To be able to configure the server without changing the source code itself, the system configuration is stored separately. The environment variables like host addresses, ports and other settings are stored in a '.env' file at the root folder of project. This makes it easy then for example to deploy the server code onto a EC2 instance in AWS, and for each instance provide

different '.env' file changing the server setup programmatically, and at the same time without any other dependencies.

3.4.4.1 Dotenv-safe

Dot-env safe is a JavaScript library building upon dotenv plugin to provide a safe way of loading configuration / environment variables during runtime.

When dot-env plugin's load method is called, it tries to locate '.env' file in the project folder, and inject all the variables it contains into node process environment object.

Dot-env safe then build upon this, which tries to load '.env.example' file, which acts as a template for the '.env' file, and if it detects that some of the variables declared in the template are missing in the '.env' file, it throws an error, preventing the app from getting into undefined state which could occur when continue to run with some of the variables being actually undefined

3.4.4.2 TrivialDB

For the storage of system entities like locations and navigators I chose a different approach. While they are basically a part of server configuration as well, it is useful to be able to edit them during runtime, remotely through a REST API. For that purpose, other system component - the management frontend - was created. This way the location and navigator configuration can be managed without the need of rebooting the server, which is definitely more convenient for a possible sys admin.

At the same time, it is needed for calculating user navigation path, that the entire location graph is present in memory. As a solution it was decided to basically store this data serialized into a file, same as it is with the '.env' file, but make the file follow a JSON syntax, and have a CRUD interface available for its editing. For this task a TrivialDB library was used, which acts as a simple in-memory database, following exactly the principle stated.

3.4.4.3 Babel

JavaScript is very dynamically progressing language, and its ecosystem is evolving even more rapidly. It is not uncommon to find in production codebase language constructs which are only in proposal stage, and as such no JavaScript interpreter will understand them, nor is it uncommon to code in languages only derived from JavaScript as is Typescript or CoffeeScript.

Common aspect of this is, that JavaScript interpreter does not understand this syntax, so it must be transpiled first into common JavaScript to be able to run in regular runtime like browser or Node.js.

The most commonly used transpiler is Babel. It must be configured to know which ruleset to apply to our code, which is usually provided in a form of '.babelrc' file placed in project root folder. There are several options of how to execute Babel. While being a CLI application, it can be run manually from command line, but more usual is to somehow integrate it into the project build process. For simple project it might be viable to execute the project code using babel-node, which is a tool which will first transpile the code, and then automatically feed it into a node process, which is a feat similar of piping the output of babel transpiler into node.

3.5 FIND SERVICE

Crucial element of the system is identification of user's actual location. For the primary method of doing this, an open source solution called 'FIND' was chosen. FIND is a web service written in Go, directly meant to be used as an indoor positioning framework.

It's working on a client based fingerprinting principle, which was chosen to be the localization method used in this system. FIND provides a simple REST API with two main endpoints - 'learn' and 'track'. These endpoints accept WiFi fingerprints, and uses machine learning classificators - either Naive Bayes or Random Forests, to first learn getting fed data via the learn endpoint, and then identify WiFi fingerprints which are sent to it with the track endpoint.

For development and testing purposes a publicly available cloud instance of this server was used, but it can be installed and run locally for production release.

3.6 NAVIGATOR HARDWARE

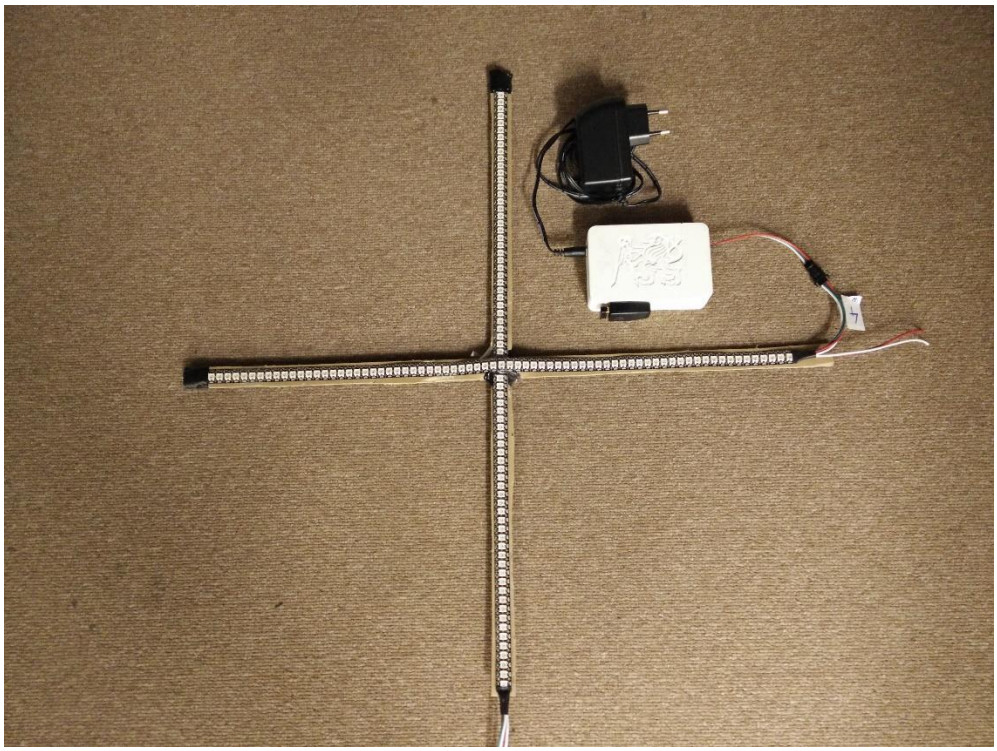

Figure 7: Completed prototype

3.6.1 Controller

There were many candidates to choose from for use as a hardware platform of navigators. With regard to the assignment requirements, my previous experience, capabilities and price the following two candidates has been considered:

3.6.1.1 NodeMCU

NodeMCU is a development board based on an Espressif ESP8266 SoC.

Espressif ESP8266 (Espresiff, 2017) is a highly integrated SoC, containing 32bit MCU and WiFi stack, which quickly gained a massive traction when it became widely available, because of its high versatility, IoT application potential, and very low asking price. ESP8266 can be programmed natively in C with Espressif SDK, or using Arduino Core[5], which has the added benefit that most Arduino libraries can be used as well.

[5] https://github.com/esp8266/Arduino

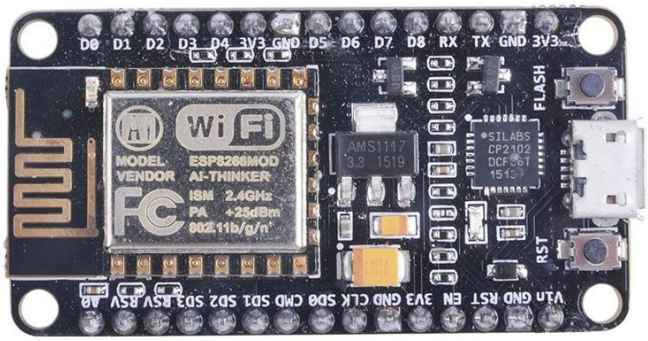

Figure 8: ESP8266 based NodeMCU board

3.6.1.2 Raspberry Zero W

Raspberry Zero W (Raspberry Pi Foundation, 2017) is a successor to Raspberry Zero, a single board computer which caused a disruption when it launched with a price of mere $5. It is almost the same board as Zero, but it additionally has a BCM43143 WiFi and Bluetooth chip.

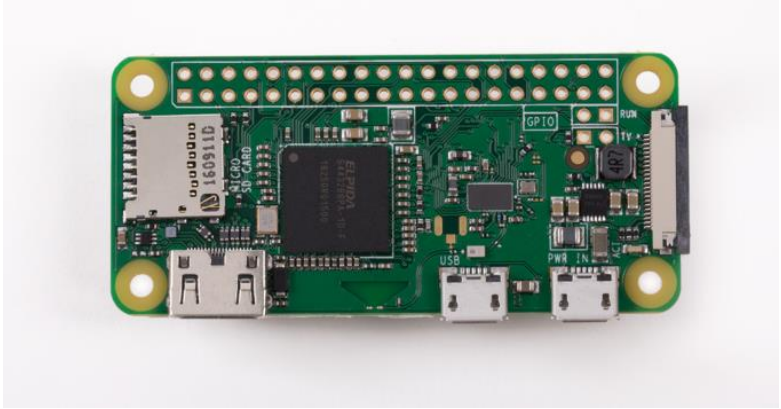

Figure 9: Raspberry Zero W (Raspberry Pi Foundation, 2017)

3.6.1.3 Controller Selection

Both of these boards were considered because of having integrated WiFi, low asking price and vast community support, but otherwise they are fundamentally different.

While ESP8266 is a microcontroller, Raspberry is actually a tiny PC running Linux OS. How that affects our use-case mostly is in the way these are configured and programmed.

ESP8266 as a microcontroller, and as such is controlled by a firmware - single compiled C program. Whatever functionality we might want to provide, and configuration which should be done has to be specified in this program. This is equally limiting and liberating in a sense, as when all what it is being done is specified in a single place, it gets quite easy to study, change and manage. However, when some new functionality should be provided, it is always necessary to reflash the firmware of this device, and hardware based additions have to have their circuitry integrated.

Raspberry as a PC is much more flexible, as new hardware capabilities can be as easy as plugging another peripheral into USB port, and plethora of different functionalities can be managed and

configured without changing the scripts. At the same time, since it is running regular Linux operating system, lot has to be configured to run the way is needed.

If the choice were to use the ESP8266 instead, it would always run the same firmware after booting, so it's much easier to be sure everything is working the way it should, and keep on doing that.

In the end, after discussion with supervisor of this work Raspberry was picked to be used, mainly because of its flexibility being in line with the non-functional requirement of building an extensible platform.

3.6.2 LED signs

While it was obvious, that the best fit for this job will be digital LED strips which offers the ability of controlling each RGB LED individually, there are some key differences between various types of them.

3.6.2.1 Three wire and four wire strips

There are many types of LED control ICs. The most general two categories of them could be defined as the ones which have only a single data input, which are represented by WS2812 IC (WorldSemi Co.), and the ones which have data and clock input. One of those drivers is WS2801 (WorldSemi Co., 2008). Difference is that those which have a clock input use it to know, that data is available, and read from data input only when the clock is running. This fact can have a direct impact in application of strips which uses these ICs.

LED strips which are not slaved by clock input have to be controlled by signal with a very tight timing, and when the signal is being transmitted it cannot be interrupted. This make these ICs hard to control directly from a high-level operating system, as they might interrupt processes at any time. Usual way of controlling these from regular OS is with additional microcontroller.

Fortunately, there is a way how to control WS2812 IC from Raspberry Pi, even though it's not quite straightforward. Raspberry Pi is equipped with a PWM module, which coupled with reading data via Direct Memory Access (DMA) provides a way how to create signal precise enough to control WS2812 drivers while not being interrupted by OS task manager (Garff, 2017).

3.6.2.2 Connection layouts

From the different shapes of intersections arises need for different shapes of navigator's LED strip formations. Most common ones are X and T shaped intersections, but we can come across Y shaped ones as well, as well as open lobbies across which there is not a single definitive shape to use. For testing purposes, three types of LED signs were built – X, T, and Y shaped. Some of them were built using high density WS2812 strip, while the other are using low density WS2801

based strips. To build the different shapes, strips had to be cut to size, and then reconnected using cable interconnects.

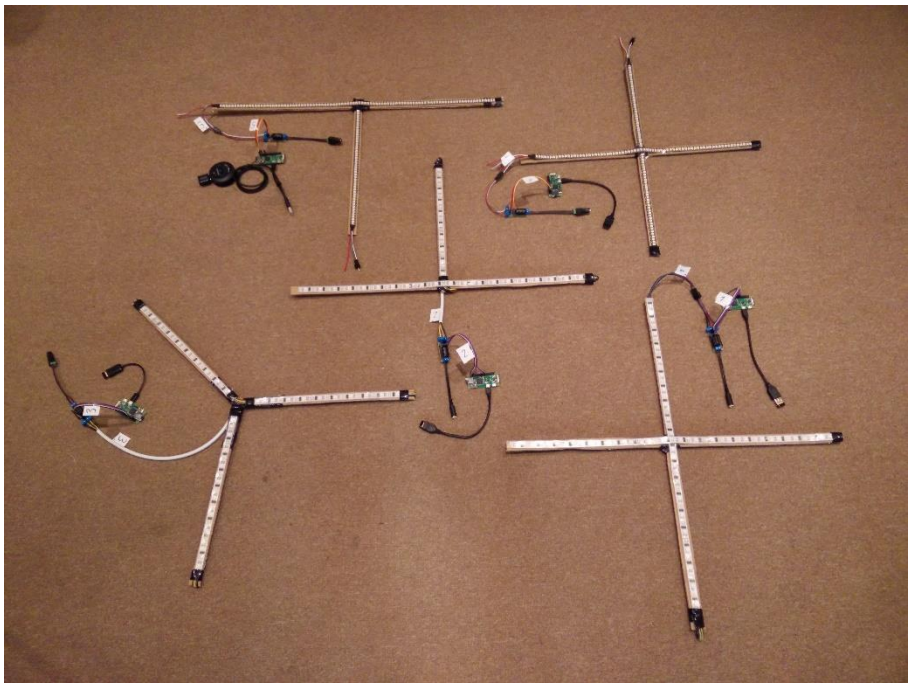

Figure 10: LED sign shapes

3.6.3 Daughter board

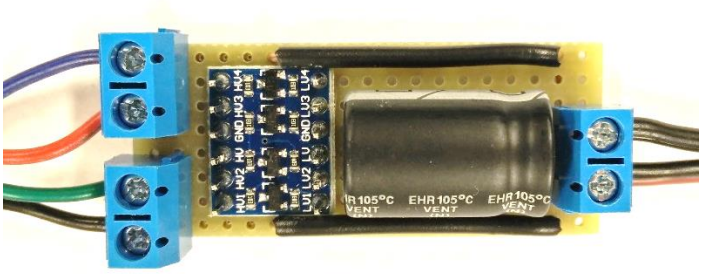

Figure 11: Daughter board detail

After testing of the LED strip when connections were made using several WAGO connectors, it was obvious that some sort of more refined solution is needed. After a quick sketching of a schema, I utilized Fritzing application to layout the circuit onto a pre-drilled stripboard. While printed PCB would be nicer, it would be more expensive to manufacture, took longer to manufacture, and cost more time to develop as well. Though if larger batches of units will have to get manufactured in the future, printing PC will be way to go.

The task the daughter board is fulfilling is interconnection of the power source with Raspberry and LED strip, while also employing a blocking capacitor to prevent voltage drops which might occur during sudden LED strip activation, and as well level shifter, to enable communication between raspberry GPIO voltage and LED strip voltage within their specification.

3.6.3.1 Blocking cap

A prominent feature of the daughter board is a 2200µF capacitor connected across the voltage input screw terminals. Its role is to block any voltage drops, which could occur in situation when s long part of the LED strip would get suddenly lit up.

3.6.3.2 Power source

The power source used for the navigator units are switching 5V DC adapters, which should be dimensioned accordingly depending on the wattage of connected LED strip. For testing purposes 3.5A adapters were used.

3.6.3.3 Level shifting

An important aspect which had to be taken into account is, that the Raspberry Pi GPIO has a voltage of 3.3V, while most of the regular LED chip drivers are running on 5V. While it doesn't create any hassle from the power source point of view, as Raspberry is being powered by 5V as well, and such we can utilize only one PSU to power them both, it can interact with the control inputs of the LED drivers.

The WS2801 datasheet specifies, that the logic voltage should be at least 80% of input voltage to be registered as a high signal. Similarly, WS2812 datasheet specifies that high signal level as at least 70% of input voltage. So since the input voltage is 5V, neither driver should be able to receive commands from Raspberry Pi's 3.3V GPIO.

Paradoxically when tested, the WS2801 even though it's logic voltage threshold should be higher than the WS2812 could have been controlled by Raspberry without any problem, but the WS2812 couldn't.

To remedy this situation, level shifters are being widely used. Level shifters are usually a quite simple circuit, whose operation is based on a mosfet transistor. This transistor allows us to send lower voltage signal to its gate terminal, which will open a channel between the drain and source terminals, and such is able to control a higher voltage circuit without any mechanical parts, opposite to when relays are being used.

3.6.4 Enclosure

One of the necessary steps needed for the prototype to become a product, is to create an enclosure for the electronic parts to fit into. One of the options for a quick solution is to use one of the universal boxes available in electronic component stores. Drawbacks are that one has to drill all the holes into the enclosure manually, attach the components inside freehand, and the box itself is usually not a perfect fit. That is acceptable for a single prototype, but would be too time consuming for manufacturing of more units.

For navigator unit a model for 3D printed enclosure has been created, to have a nice engraved enclosure, into which the parts fit spot on and minimal work is required for assembly of each unit.

3.6.4.1 3D Printing

In regular manufacturing when in need of an enclosure, the usual way to go is to create an injection form, with the help of which custom shaped cases can be manufactured cheaply and in great quantities. However, the initial investment is very high, too high for small scale projects like this one.

In the last decade a new trend has emerged in a form of 3D printers. With help of these, anyone can create a custom made enclosure exactly up to his specifications, without any initial investment for the injection form. Each product can also differ one to another. Drawback of this approach is, that compared to injection molding it takes several orders of magnitude more time to print out a single product and the material used cost more as well.

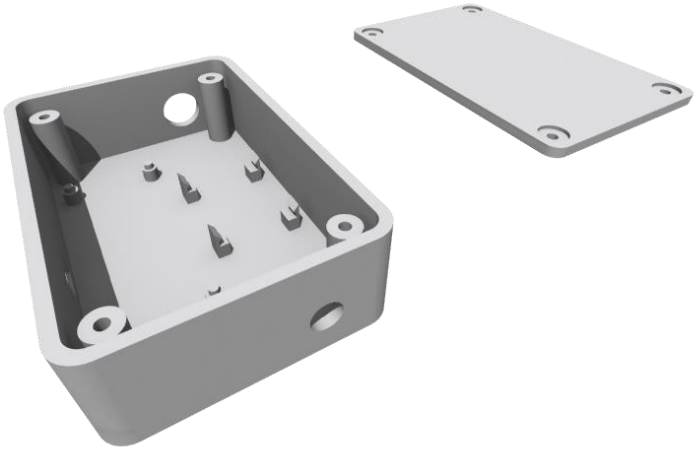

Figure 12: Enclosure rendering

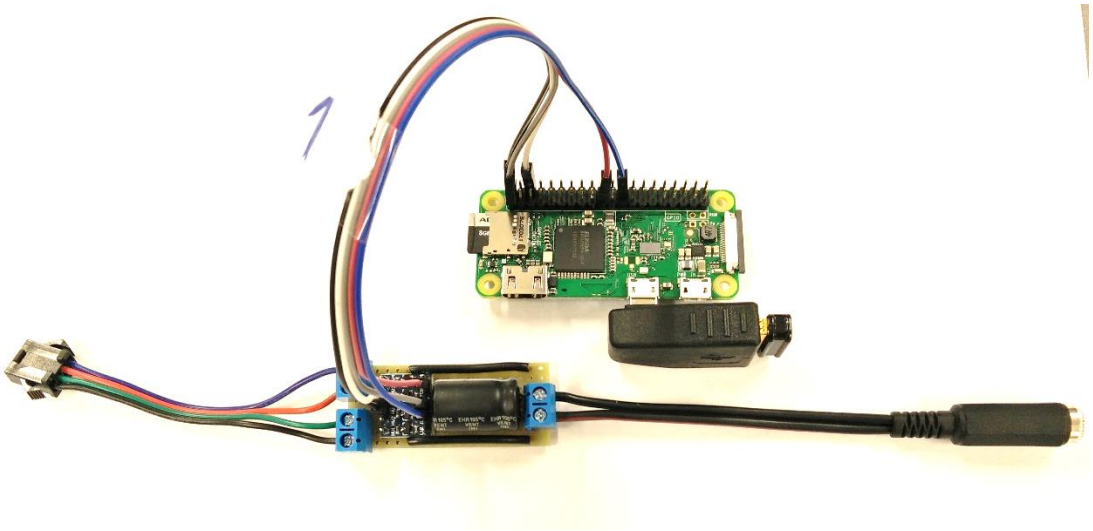

Figure 13: Navigator unit without enclosure

Figure 15: Navigator unit during assembly

Figure 14: Assembled navigator unit in printed enclosure

3.6.5 External WiFi dongle

Additional peripheral which can be seen in the figures is an external WiFi dongle. Even though the Raspberry Zero W already has a built in WiFi, an external one with a longer range is added to be used for the upstream connection, and the built in one is set up to provide AP signal for aiding uniqueness of the fingerprint of given location. More on this topic is explained in the navigator software section.

3.6.6 Audio Support

For the navigator to be capable of audio playback, it was necessary to utilize some external solution. While the chipset of Raspberry is able itself of audio output, it is only a PWM output, not a true DAC, without any physical connector on board. Another issue is, that because of the way how WS2812 strips are being controlled – using the PWM as well, they cannot be used together with the internal sound output (Garff, 2017).

To resolve this, an external USB sound was added, with little to no configuration needed. There are also specialized modules which fit onto the Raspberry called pHAT form factor. One of them is pHAT DAC[6], which could have been used for audio output as well, but unfortunately is using the same GPIO pin as the one being used for control of WS2812. That is because the Inter-IC Sound Bus (I2S) interface (Cypress Semiconductor Corporation, 2016) which that DAC utilize is assigned to the same pin as PWM output.

[6] https://shop.pimoroni.com/products/phat-dac

3.7 NAVIGATOR SOFTWARE

Even though it's role and functionality is quite different, the software running on navigator follows the same pattern as the backend server, being a Node.js application based on an Express server.

3.7.1 Navigator REST API

Method	Endpoint	Description
POST	/navigate	The primary endpoint for user navigation. This endpoint accepts requests for running navigation animations
POST	/stop/audio	Endpoint specifically created for stopping of audio playback.
POST	/stop/:id	Using this endpoint, the animation with given ID is stopped.
POST	/show	This endpoint can be used for showing arbitrary animations.
POST	/play	This endpoint accepts requests for playing audio. A filename is being specified in the body of request.

Table 4: Navigator REST API

3.7.2 Operating System

The officially supported distro – the Raspbian (Raspberry Pi Foundation, 2017) was chosen to be used. Latest stable release to this date is Raspbian Jessie, which is the one running in the navigator units.

3.7.2.1 System Configuration

As noted in the 3.6.1.3 Controller Selection article, since a regular Linux OS is running the navigators, a lot has to be configured the way it is needed. Most of this configuration relates to the WiFi setup described later on. Another issue to deal with though, was the need of automatic startup of the navigator software, which is being executed by the npm command. That was solved by calling a startup script from '/etc/rc.local' file, which is being executed by init process when the system reaches multiuser runlevel upon startup (Raspberry Pi Foundation, 2016).

3.7.2.2 Image cloning

After the initial configuration of first Raspberry was done, it was needed to duplicate its system image to all other units.

For this task I used a command line tool 'dd'. 'dd' works very low level, as it simply takes the stream of data from a block device configured as its input, and saves it into output file or block device (The IEEE and The Open Group, 2016). Using 'dd' I first copied the whole SD card into an image file, and then replicated it onto the other SD cards.

Problem has arisen, stemming from the fact, that for the first Raspberry on which the image was developed, a different SD card manufactured by Kingston has been bought. For all the other navigator units, an ADATA brand cards has been bought later. Though they were both advertised as having the same size, the ADATA cards were actually 75 MiB smaller than their Kingston counterpart. Were it to happen the other way, then there would have been no problem whatsoever. But since 'dd' created an exact copy of the Kingston card, the resulting image was larger than the target card could accept.

This issue was resolved by shrinking the last partition on the master card, leaving some space unallocated. Knowing the last sector used by the last partition, 'dd' was set to copy only until this sector. In this manner still an exact copy of the card was created, but without its last part, which contained only unallocated space. Thus the image then fitted the ADATA cards.

3.7.3 WiFi configuration

One of the reasons I chose Raspberry Zero W as a platform for navigator is that it has integrated WiFi adapter.

For making the fingerprinting more precise, the integrated Raspberry's WiFi is configured as an AP, so at every intersection there is present a strong signal from the corresponding navigator, which makes the fingerprint more unique to that intersection.

At the same time though is needed for the Raspberry to be connected to the infrastructure network, to be able to communicate with server.

These contradicting requirements created a problem, that the Raspberry needs to be present in two networks at the same time, even though the other network is just a dummy AP.

There are two solutions to this problem. The first and simpler one is to just add another WiFi adapter. The other is to set the built-in adapter to concurrent station and AP mode.

While the STA+AP mode will solve this issue without any additional hardware, it is not without drawbacks. For one, when running these two modes parallel, the adapter has to operate them on the same channel. But the AP service does not automatically tune itself onto the same channel the station service is running on, it has to be configured manually like that upon start up. Another problem is that the infrastructure AP the Raspberry is connected to might change channels over time, to which the AP service would have to react.

Another problem is that transmitting and receiving at the same channel will create a noise, which will decrease the throughput and range.

I've decided to add another adapter, though not really for the cited reasons, but because the WiFi range of the built-in WiFi of Raspberry Zero W is quite limited, and I have the need to be able to connect to the infrastructure APs which while in a small office might be close enough, but in the lengthy corridors of our faculty could get easily out of range.

There are several kinds of antenna technologies which are used with PCB circuits. There are external antennas which could be connected through a U.FL connector, onboard soldered ceramic antennas, or traces directly on the PCB designed to act as an antenna. Raspberry Zero W uses a new kind of antenna, which uses a hollowed out cavity in the ground plane, which has a resonant frequency of 2.4Ghz - the one which WiFi and Bluetooth operate on.

3.7.4 LED driver bindings

As it was already explained, the WS2801 and WS2812 have a different implementation of their communication protocol. While with WS2801 the Raspberry is able to communicate natively using the SPI pins, to communicate with WS2812 a more difficult approach combining PWM output and direct memory access has to be used. That also means using a different library. Fortunately, both[7][8] already had a JavaScript binding available. The only problem in using them,

[7] https://github.com/n-johnson/node-rpi-sk6812-native
[8] https://github.com/Jorgen-VikingGod/node-rpi-ws2801

was that the WS2812 didn't have an up to date dependency to the native module. The way of learning this fact was complicated though, as it required debugging in the binding library's underlying C code.

3.7.5 Animations in JS

For creating the LED strip animations, I had to write my own animation code, as Node.js does not have any animation library of its own. Collection of methods in module 'animator' has been written, which for running animations accepts instances of objects having method 'tick'. By using a setInterval Node.js method, the 'tick' method of animations is getting repeatedly called after a set delay for each frame it should render, thus resulting in an animation.

3.7.6 Strip endings addressing

It was needed for the server to be able to specify for the navigator from which direction is user approaching it, and to which next location is he heading to, so the navigator could run the animation oriented accordingly.

An idea was drafted, that navigators could be saving which LED strip ending correspond to different locations they are pointing to, so then it could be possible to send to navigator command only with the strip ending codes of the locations, and the navigator itself would infer which LEDs it needs to light up, and in which direction should the animation run.

Problem of this idea was, that saving only addresses of endings on the navigator side is not enough, as it is then with varying LED sign shapes and sizes impossible to decide which LEDs create a path between those two endings.

The final solution which is being used, is that navigator is persisting a JSON object which for each ending keeps two values - lowest and highest index of LEDs which are contained in the arm of LED sign for the given index. These numbers are always stored in the order from the center to the endings, which resulted in a definitive addressing system, with which it is enough for the server to send only the 'from' and 'to' strip ending IDs, and navigator can infer the path between them.

3.7.7 Audio Playback

For audio playback a free CLI player mpg321[9] is being used. The playback is being handled by a JavaScript wrapper module play-sound[10].

[9] http://mpg321.sourceforge.net/
[10] https://github.com/shime/play-sound

3.8 ANDROID CLIENT APP

The android client application is beside the navigational lights the only part of system which the user sees and interacts with. Its purpose is to provide for users a way of choosing desired destination and a form of tracking progress feedback, and for the system a way to gather WiFi fingerprints for localization purposes.

3.8.1 User Interface

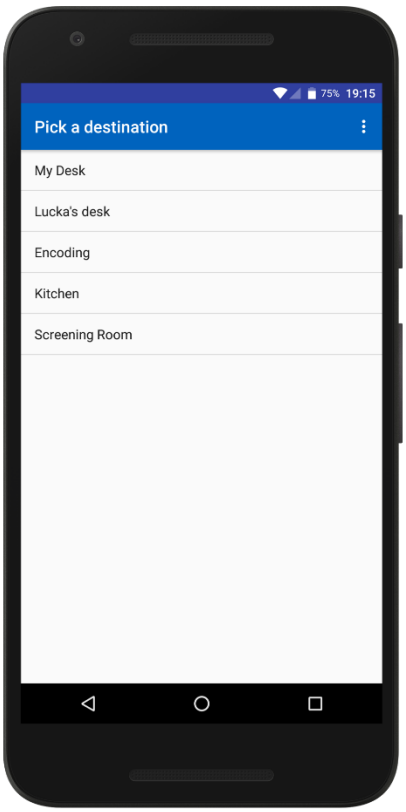

Figure 17: Destination picker screen

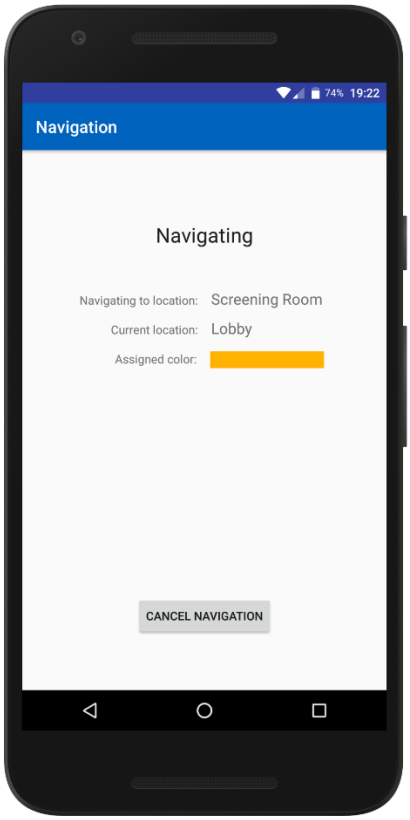

Figure 16: Navigation screen

According to the requirements the app should be easy to use, and be least intrusive as possible. That was reflected in the UI design, which is based on only two screens. First screen lists possible destinations which user can choose from. After user selects his desired destination, applications transitions to the Navigation activity, which shows the current status, user's current location, and his assigned color which he should follow on the Navigator units.

3.8.2 Tracking Service

In the background a tracking service runs, which every five seconds polls Androids systems's WifiManager class to start a new scan of WiFi networks. When the scans finished, the tracking service is informed about it in a form of system intent. It then gets the new WiFi fingerprint, and sends it to the backend server.

3.9 MANAGEMENT APP

The management frontend is a CRUD web app meant for making the system configuration changes easier - it can display the system entities, create new ones and alter existing. It's built using React components and styled using Material-UI style library.

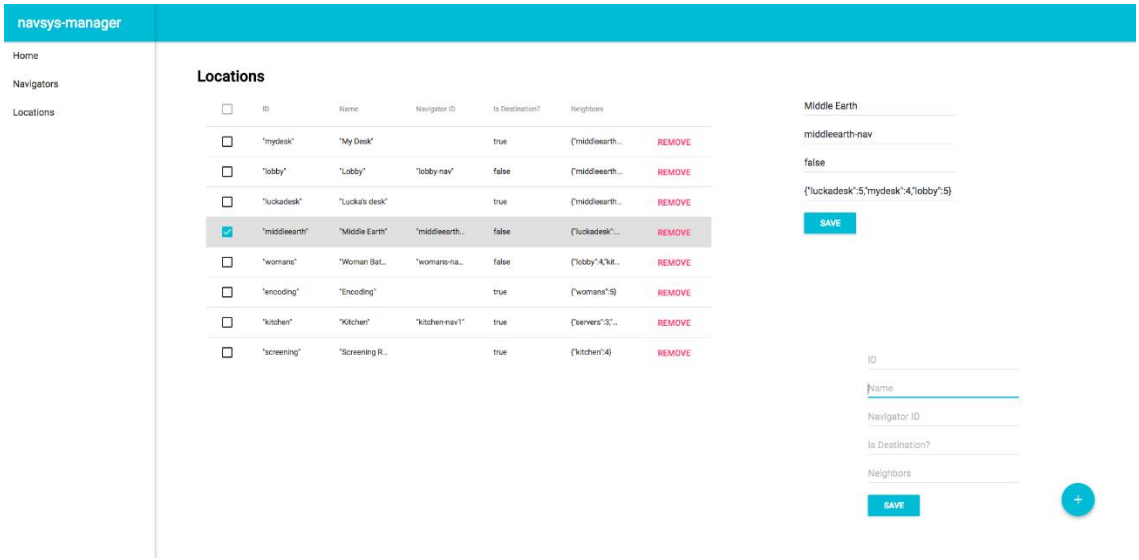

Figure 18: navsys-manager interface

3.9.1 React

React is a popular frontend JS framework, create by Facebook. Originally open sourced in 2013, in quite a short time has acclaimed status of one of the most used frontend frameworks .

3.9.1.1 JSX

JSX is a preprocessor usually used in conjunction with React, extending the JavaScript syntax by allowing of writing HTML or XML like tags, which are then transpiled into React components.

3.9.2 Material-UI

Material-UI is a set of React components that implement Google's material design specification[11].

3.9.2.1 React Grid System

Material-UI in its current stable version does not include any grid system, which is a popular feature in a competing Bootstrap web UI framework. This drawback was solved by using react-grid-system library, which does even markets itself as being Bootstrap-like (Meyer, 2017).

[11] http://www.material-ui.com/

4 EVALUATION

The navigation system which has been developed is supposed to help find visitors their destinations faster and more easily, than if the visitor would have to follow some other existing conventional methods. To support this claim, in this chapter the results of implementation will be evaluated. The system was tested in several ways, to prove that it is capable of fulfilling its role, and it is verified that all the defined requirements have been met.

4.1 DEVELOPMENT TESTING

During the development single components has been tested to verify functionality they are correspondingly responsible for. After all of the components have reached an operating level of capability, integration testing started taking place overtime, to verify that each of the components is able to communicate with the rest of the system. When the integration testing was done, the system has proven to be functional.

4.2 VALIDATION TESTING

When the whole system was tested to be operational, to prove that it serves its purpose validation testing has taken place.

The testing session has taken in place in the office of Blue Sky film distribution company.

4.2.1 Navigation speed testing

The system is supposed to make users find their way fast. While the time it takes user to find a way to the place he hasn't been yet varies, the time taken if he already knows the way is known - the shortest path between the origin and the destination at a walking pace.

Testers were asked to first walk the path to the destination at their standard walking pace. This time sets the reference. Then they were asked to walk the same path with the system navigating them with following conditions:

- At any intersection they would wait until the system would show them where to go next.
- At any time, the system would fail to navigate them – e.g. a navigator would fail to light up, or show the wrong way, the test fails.

Durations of the test runs were then compared to the reference duration.

Participant	Reference duration	Tested duration
Participant A	0:26.42	0:31.24
		0:25.44
		0:33.67
Participant B	0:19.74	0:21:43
		0:26.80
		0:22.35

Table 5: Navigation speed testing

4.2.2 Error correction testing

The system has to be able to correct path of users which have strayed from the determined path. For this test the users were asked to follow the path to their destination being navigated by the system, but at one randomly chosen intersection intentionally take a wrong turn.

System had to give user an alert feedback to try to correct the path, otherwise it fails the test.

Test run	Alert fired
1	YES
2	YES
3	YES
4	YES
5	YES

Table 6: Error correction testing

4.2.3 Multiple users testing

One of the requirements for the system was, that the system should be able to navigate multiple persons at once.

- It was tested that having multiple active sessions does not affect the performance of the system.
- It was tested that multiple users can be simultaneously navigated on non-conflicting paths
- It was tested that multiple users can be simultaneously navigated on a conflict path, which results in their visual cues being alternated on the conflicting navigation sign

4.3 GOALS FULFILMENT

Research tasks has been covered in second chapter of this work, and based on that a proposal of the system has been drafted. When implemented, the system has proven to be functional. It has been designed and implemented in such a way, that all the functional and non-functional requirements have been taken into account. Localization using FIND server coupled with the constrains set in place by location graph has been tested to provide stable results for the tracking to be robust enough.

4.3.1 Functional Requirements Fulfilment

- System will guide users along the shortest path to their destination
  - ✓ System is using Dijkstra's algorithm to calculate the shortest path
- System will guide users using visual cues present in the physical world
  - ✓ System is using LED signs made of individually controllable LED strips for user guidance
- System will enable users to choose any destination available
  - ✓ System integrates an Android application which
- System will be capable of using audio output to provide navigational cues and entertain users
  - ✓ Navigator units support connection of USB audio peripherals, and its API contains an endpoint for triggering a playback of audio files.

- ✓ Backend asks for playback of an alert audio file, when user goes a wrong way

- System will be able to guide users back to their correct route should they take a wrong turn
  - ✓ When backend evaluates that user has taken a wrong turn, it can send to Navigator an alert flag along with the navigation request, which will make the navigation animation to be interspersed with an alert animation

4.3.2 Non-functional requirements

- System will be capable of guiding multiple users at the same time
  - ✓ System has no problem running several user session performance wise
  - ✓ Each user has his unique color assigned to be shown on the Navigator units
  - ✓ Shall more users meet at one Navigator at the same time, their animations will be alternated
- System will be remotely configurable
  - ✓ System integrates a management web application
- System will be horizontally scalable
  - ✓ System can accept new Navigator units, and puts no limit on their total number
- System will be extensible
  - ✓ System is open source
  - ✓ Each component is modularized
  - ✓ All components communicate using a defined REST API
- System have to be capable of continuous operation
  - ✓ There is no limit on how long the system could be run, during testing no unit failed by itself
- Deployed hardware will use wireless communication
  - ✓ All navigator units communicate with server using WiFi network

5 CONCLUSION

In this work a current state of indoor navigation systems has been analyzed. While there are positioning technologies capable of providing working indoor localization, in most of the buildings there isn't a system available using any of them, and old school ways of navigation continue to being used. Reason for that might be the design fragmentation of current available systems, and primarily lack of any one quality free generic system, which would be widespread enough so the administrators would be willing to integrate it into their building. Most users won't be bothered to use these systems, if they usage would require installation of a separate application for each building they visit. However, there are several hopeful projects, especially the Google Indoor Maps, which might fulfil this role.

Technologies capable of providing the positioning data has been researched, which are foremost the technologies which the current generation of smartphones are capable of working with, as smartphones seems to be the best target for the positioning systems, not the user himself.

A navigation system using WiFi fingerprinting positioning has been developed, which is fully capable of navigating visitors in the building it would be deployed in, as long as they have an Android smartphone, and are willing to install the fingerprinting application. A passive tracking possibility has been researched as well, which would be independent on the smartphone's OS and wouldn't require installation of any application. It was then concluded however, that such techniques could be used maliciously without any consent of the smartphone owners, and mobile OS developers themselves are trying to prevent deployment and usage of such systems.

In the future perhaps some indoor navigation system will gain enough of traction, that it will get integrated into the smartphone OS itself, which could then enable the possibility to get user's consent and tracking data without any application needed, and provide the most streamlined and least intrusive user experience.

5.1 RESULTS

The resulting implementation of navigation system has been proved to be able to navigate users directly to the destination they desire to visit, and in most cases even at their regular walking pace. Thanks to that, it aids the users by saving their time, and relieve them of the stress related to searching for a location never visited before.

The developed system makes sense to deploy in a building where following conditions are met:

- User visits this building repeatedly, so he might be willing to install the tracking application
- In this building is a large number of possible destinations, some of which even regular user might not know location of
- Users regularly visiting this building might have the need of visiting locations not previously visited before

A school campus is a perfect fit for these criteria, as the students:

- Visit it almost on a daily basis
- There is a large quantity of classrooms, lecture halls or study rooms
- Every semester need to find a location of classrooms for the new schedule, and sometimes might have to visit teacher's office

Because of that it made sense to aim this implementation toward deployment in the Faculty of electrical engineering, where testing of this system in limited deployment will occur in weeks following delivery of this work.

5.2 FURTHER WORK

Even though most of the defined goals have been met, there is still work needed to get from a prototype state to a releasable product.

At the hardware side, it is planned to fit the LED signs into aluminum frames, as the prototypes were tested only with a cardboard backing. All of the navigators should be also equipped with a 3D printed enclosure and standardized 4 pin connector for the LED sign, which has been tested on only one of the units.

At the software side there are few issues which were found during testing which still has to be fixed. There are also several features already planned which might be included in the future. It would be nice to have a WebSocket[12] based dashboard in the management server, which would provide a live view of the current system activity. Another one is, that automatic triggering of the audio cues is now limited only to alert users going the wrong way. It would be nice to get more types of cues voice acted and implemented into the system.

5.3 PARTING NOTE

This project has been a great opportunity to design and develop from scratch a real applicable system with interesting functionality. It was developed using a combination of technologies that I have used in the several past years at the university, in my regular work and other freelance projects, as well technologies I wanted to gain experience with myself, so it was a perfect fit. It tested all the skills I have gained so far, and gave me a valuable experience which I'll make use of on a daily basis.

[12] https://developer.mozilla.org/en-US/docs/Glossary/WebSockets

REFERENCES

Altman, N. S. (1992). "An introduction to kernel and nearest-neighbor nonparametric regression". *The American Statistician*, 175–185.

Britannica. (n.d.). *Trilateration*. Retrieved from https://www.britannica.com/science/trilateration

Chrysostomos, N. (n.d.). mpg321. Retrieved from http://mpg321.sourceforge.net/

Chui, C. K., & Chen, G. (2009). *Kalman Filtering with Real-Time Applications.* New York: Springer.

Cypress Semiconductor Corporation. (2016). Inter-IC Sound Bus (I2S). Retrieved from http://www.cypress.com/file/133906/download

Espresiff. (2017). *ESP8266 overview*. Retrieved from https://espressif.com/en/products/hardware/esp8266ex/overview

European Global Navigation Satellite Systems Agency. (2017). *Galileo Increases the Accuracy of Location Based Services*. Retrieved from https://www.gsa.europa.eu/news/results-are-galileo-increases-accuracy-location-based-services

Garff, J. (2017). *rpi_ws281x GitHub repository*. Retrieved from GitHub: https://github.com/jgarff/rpi_ws281x

Green-Armytage, P. (2010). A Colour Alphabet and the Limits of Colour Coding. *Colour Journal*.

Han-Sol Kim, W. S.-R. (2017). Indoor Positioning System Using Magnetic Field Map Navigation and an Encoder System. (V. M. Passaro, Ed.)

infsoft GmbH. (2016). *Indoor Positioning & Navigation.* Retrieved from https://www.infsoft.com/portals/0/images/solutions/basics/whitepaper/en-indoor-navigation-indoor-positioning-infsoft-ebook.pdf

infsoft GmbH. (n.d.). *Indoor Navigation for Zurich Main Railway Station*. Retrieved from https://www.infsoft.com/industries/railway/success-story

Jedlička, J. (2013). Light Chain Control. Retrieved from https://dip.felk.cvut.cz/browse/details.php?f=F3&d=K13139&y=2013&a=jedlija5&t=bach

Mathieu Cunche, C. M. (n.d.). On Wi-Fi Tracking and the Pitfalls of MAC Address Randomization. Retrieved from https://ido2016.sciencesconf.org/122873/document

Meyer, D. G. (2017). *React Grid System*. Retrieved from https://github.com/JSxMachina/react-grid-system

Pourhomayoun, Jin, & Fowler. (2012). Spatial Sparsity Based Indoor Localization in Wireless Sensor Network for Assistive Healthcare Systems. *EMBC 2012*.

Raspberry Pi Foundation. (2016). *rc.local*. Retrieved from https://github.com/raspberrypi/documentation/blob/master/linux/usage/rc-local.md

Raspberry Pi Foundation. (2017). *Raspberry Pi Zero*. Retrieved from https://www.raspberrypi.org/products/pi-zero-w/

Raspberry Pi Foundation. (2017). *Raspbian*. Retrieved from https://www.raspberrypi.org/downloads/raspbian/

Rish, I. (2001). An empirical study of the naive Bayes classifier. *IJCAI Workshop on Empirical Methods in AI.*

The IEEE and The Open Group. (2016). *dd.* Retrieved from The Open Group Base Specifications Issue 7: http://pubs.opengroup.org/onlinepubs/9699919799/utilities/dd.html

WorldSemi Co. (2008). WS2801 3-Channel Constant Current LED Driver With Programmable PWM Outputs. Retrieved from https://cdn-shop.adafruit.com/datasheets/WS2801.pdf

WorldSemi Co. (n.d.). WS2812 Intelligent control LED integrated light source. Shenzhen. Retrieved from https://cdn-shop.adafruit.com/datasheets/WS2812.pdf

Wutjanun Muttitanon, N. K. (2007). An Indoor Positioning System (IPS) using Grid Model. *Journal of Computer Science 3 (12)*, 907-913.

A LIST OF ABBREVIATIONS

AP	Access Point
AWS	Amazon Web Services
CLI	Command Line Interface
CRUD	Create Read Update Delete
FPS	Frames Per Second
GPIO	General Purpose Input / Output
GPS	Global Positioning System
ID	Identity Document
I2S	Inter-IC Sound Bus
IoT	Internet of Things
LED	Light Emitting Diode
MCU	Microcontroller Unit
PIR	Passive Infrared Sensor
PSU	Power Source Unit
PWM	Pulse Width Modulation
REST	Representational State Transfer
RGB	Red Green Blue
RSSI	Received Signal Strength Indicator
SD	Secure Digital
SoC	System on Chip
SPI	Serial Peripheral Interface bus
UI	User Interface
URL	Uniform Resource Locator
USB	Universal Serial Bus

B INSTALLATION MANUAL

1) Acquire source codes for all of the navsys components. It is recommended to pull latest versions from GitHub repositories. Otherwise the version 1.0.0 of all components is included on the CD with this work.
2) All the components are set up as npm packages. After verifying you are connected to internet, execute `npm install` in the root folder of every component.
3) All components should now be ready to run. See '.env' files for configurations and 'package.json' file for startup scripts.

C USER MANUAL (ANDROID APP)

This manual assumes, that the navsys-backend server is running, and that the Android application has been compiled with the correct server's address set in its constants!

The server address is not configurable from the app UI, as the regular user should not change it.

1) Install the apk file onto an Android smartphone. Minimum supported version is Android 4.1 Jelly Bean. If not already set, it will be needed to enable installation from unknown sources in the Security menu in Settings first.
2) Launch the application, it should show a list of possible destinations
3) Choose a destination
4) After initialization, app should display the current location, selected destination, and color assigned to the user
5) Follow the assigned color on navigator signs. They should lead you to the selected destination.

D BILL OF MATERIALS

For a single navigator unit with one meter worth of high resolution LED strip and dual WiFi setup, following materials were used.

Listed costs are calculated at the used quantity, valid at the time of purchase and without shipping fees.

Item	Quantity	Seller	Original price	Price in USD
Raspberry Zero W	1pc	https://thepihut.com/	£9.6	$12.45
WS2812 LED strip 144/m density	1m	https://www.aliexpress.com/	$14	$14
microSD card 8GB Class 10	1 pc	https://www.alza.cz/	169 CZK	$7,15
Edimax EW-7811Un WiFi dongle	1 pc	https://www.alza.cz/	199 CZK	$8,41
microUSB - A/F USB adapter	1 pc	https://www.alza.cz/	50 CZK	$2,11
Vigan 5V 3.5A power source	1pc	https://www.gme.cz/	278 CZK	$11.76
Universal pre-drilled stripboard 3 holes per strip	¼ pc	https://www.gme.cz/	75 CZK	$3,17
Electrolytic capacitor 2200uF 16V	1pc	https://www.gme.cz/	8 CZK	$0,34
2-pin PCB screw terminal, 5mm pitch	3 pcs	https://www.gme.cz/	15 CZK	$0,63
Logic Level Shifter Bi-Directional Four-way	1 pc	https://arduino-shop.cz/	29 CZK	$1,23
Barrel connector on cable	1 pc	https://www.gme.cz/	32 CZK	$1,35
Dupont F-F Cable	5 pc	https://arduino-shop.cz/	12 CZK	$0,51
Solid insulated wire Ø0,8	0,1m	https://www.gme.cz/	3 CZK	$0,13
			TOTAL	$63.24

Table 7: Bill of Materials

E DAUGHTER BOARD SCHEMA

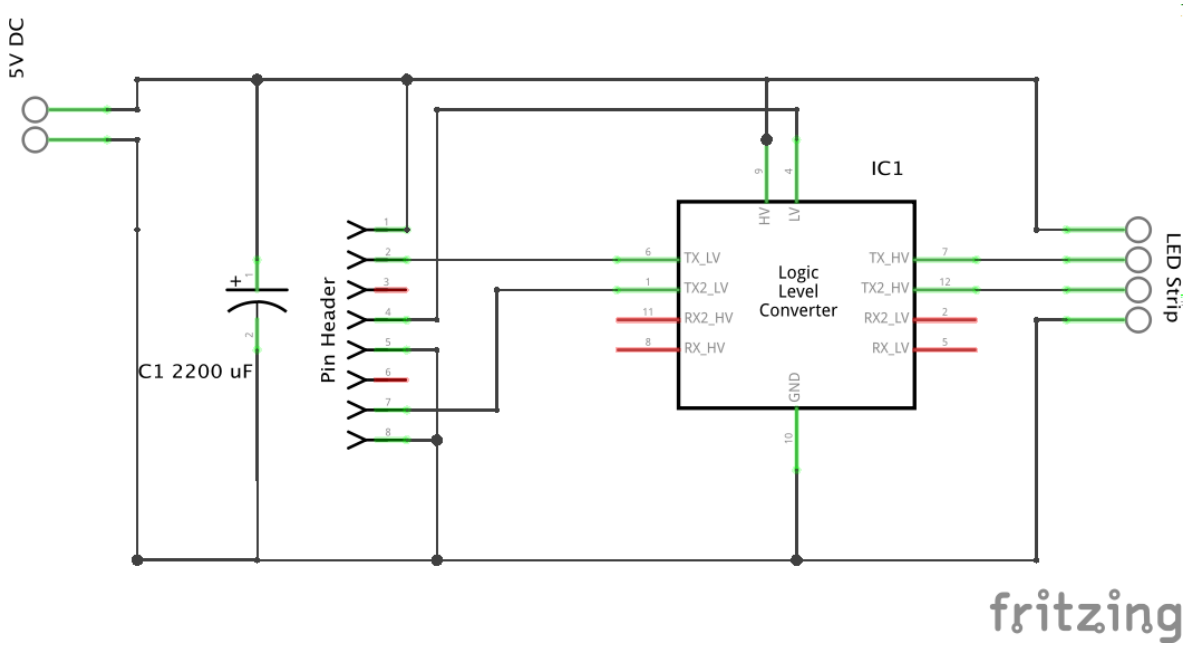

Figure 19: Daughter board schema

F SOURCE FILES

For the most up to date version it is recommended to pull the latest code from the GitHub repositories.

Source files on CD included with this work will be marked on the GitHub repositories as release 1.0.0.

GITHUB REPOSITORIES

Backend	https://github.com/yedlosh/navsys-backend
Android Client	https://github.com/yedlosh/navsys-client-android
Navigator	https://github.com/yedlosh/navsys-navigator
Manager	https://github.com/yedlosh/navsys-manager
Documentation	https://github.com/yedlosh/navsys-docs

CONTENTS OF THE CD

Name	Description
navsys.docx	This document in a Microsoft Word format
navsys.pdf	This document in a portable document format
navsys-navigator	Source files of the navigator component
navsys-backend	Source files of the backend component
navsys-client-android	Source files of the android client component
navsys-manager	Source files of the management frontend component
daughter_board	Folder containing images and source code of the daughter board schema
enclosure	Folder containing 3d model and renders of the enclosure for navigator units

Table 8: Contents of the CD