ETH zürich

Soft Robotics Laboratory

**Semester Thesis**

# LiDAR-based Haptic Feedback Wristband for the Visually Impaired



Maximilian Eberlein

January 2025

Supervision

Prof. Dr. Robert Katzschmann
Dr. Ronan Hinchet

**Abstract**

Approximately 285 million people worldwide are either affected by complete blindness or by moderate to sever visual impairments. It is estimated that every 5 seconds, another person in the world loses their sight. Many individuals with visual impairments rely on their remaining senses (hearing, touch, and even smell) as well as tools like walking sticks and assistance from guide dogs to navigate their environments and avoid obstacles. White canes are a simple and reliable solution for local navigation and obstacle detection. However, they can only detect obstacles up to chest height and within a limited range. This semester thesis seeks to address this problem by developing a system that combines a portable, untethered, easy-to-use and low-cost physical device with a custom-made smartphone app. The device introduced in this thesis is a discreet, compact wristband designed for intuitive, hands-free use. It measures the distance to nearby objects and provides haptic feedback to indicate their proximity, assisting users in avoiding obstacles and performing local navigation tasks. Additionally, it supports global navigation by delivering haptic cues if the user deviates from a route defined in the app. By applying this system as an assistive addition to the white cane, visually impaired individuals will hopefully be able to extract more information about their surroundings if needed, helping them in the difficult task of navigating unfamiliar environments and alerting them of obstacles beyond the reach of their canes.

# Contents

**1 Introduction** **1**
1.1 Motivation . . . 1
1.2 Related Work . . . 1
1.3 Outline . . . 3

**2 Initial Stages & Prototyping** **3**
2.1 The Electronics Ecosystem . . . 3
2.2 Haptic Feedback Implementation . . . 4
2.3 Distance Measurement Implementation . . . 5
2.4 First Wristband Prototype . . . 7

**3 Hardware Optimization & Integration** **10**
3.1 Distance Sensor Integration . . . 10
3.2 Use of Flexible PCBs . . . 12
3.3 Protection of Electronics . . . 14
3.4 The Final Hardware Design . . . 15

**4 Software Implementation & Integration** **19**
4.1 Why build a Phone Application? . . . 19
4.2 How to build a Phone Application? . . . 19
4.3 The Final App . . . 22

**5 Conclusion & Further Work** **28**

**6 Acknowledgements** **29**

# 1 Introduction

There exist many assistive technologies for visually impaired individuals, ranging from smart canes with integrated navigation systems to smart glasses that read texts aloud. Significant research has gone into creating diverse products that enhance the quality of life for this community in different ways. However, there is to this day no compact, affordable and easy-to-use solution that enables visually impaired individuals to obtain distance information about their surroundings and therefore a better understanding of their environment.

## 1.1 Motivation

The goal of this semester thesis is to address this gap and develop a wristband that utilizes a LiDAR sensor to measure the distance in a specific direction and converting that information into a haptic feedback response on the user's wrist. The advantages of such a system are manifold: First, it leaves the user's hands free, which is crucial since many visually impaired individuals often have only one hand available due to the need to hold a cane. Second, it is discreet, compact and easy to put on and operate. Third, it allows the user to point in any direction of interest and receive distance information about obstacles or objects in that specific area. This functionality can also be useful for gaining a general, low-grained understanding of a room's dimensions and the placement of objects within it. Forth, integrating the wristband with a custom-made smartphone app may unlock even more features such as global navigation assistance. To the best of my knowledge, no existing assistive devices provide this unique combination of features for the visually impaired.

## 1.2 Related Work

This thesis is inspired by and draws upon the research of Katzschmann et al. (2018), who developed an intuitive, hands-free, and discreet wearable device that enables visually impaired users to detect both low- and high-hanging obstacles, as well as physical boundaries in their immediate environment (Fig. 1). My goal is to apply the same core concept in my thesis, but making it more compact and giving users the possibility to strategically scan their surroundings or areas of interest by simply pointing there with their hand.

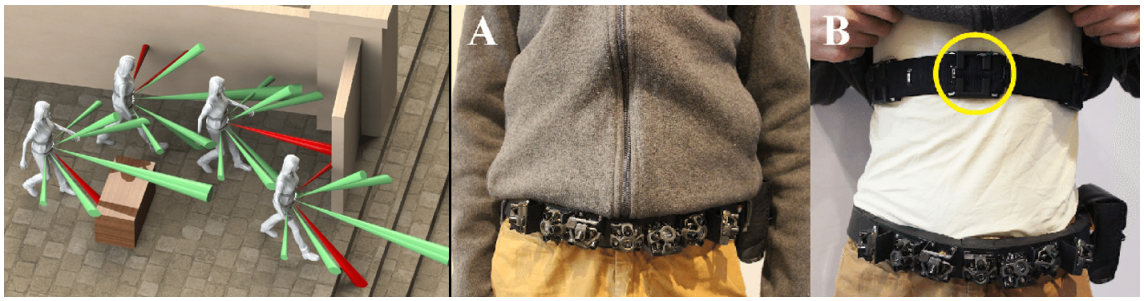



Figure 1: Safe Local Navigation for Visually Impaired Users With a Time-of-Flight and Haptic Feedback Device

Additionally, the idea of providing haptic feedback based on spatial distance information for visually impaired individuals has also been explored by Zahn and Khan (2022). They used an Intel RealSense camera mounted on custom-made glasses to deliver multidimensional distance feedback via an arm sleeve under varying lighting conditions (Fig. 2). While my approach using a miniature distance sensor does not achieve the same level of multidimensional feedback and accuracy, the wristband introduced in this thesis is significantly more discreet and inexpensive.

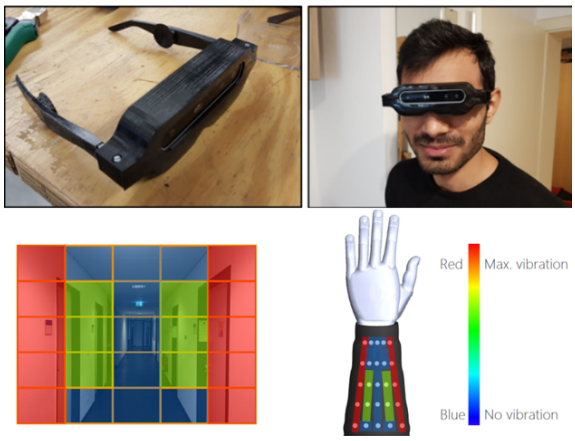



Figure 2: Obstacle avoidance for blind people using a 3D camera and a haptic feedback sleeve

Quick et al. (2023) were the only ones to attempt applying the "distance measurement to haptic feedback" concept from Katzschmann et al. (2018) to a wrist-mounted device. They developed a wristband coined OptiBand featuring a LiDAR sensor and a single haptic feedback motor that allows visually impaired people to achieve scene awareness for distances up to 12 meters (Fig. 3). However, their device can only provide distance information with the user's hand occupied, is rather bulky and relatively expensive.

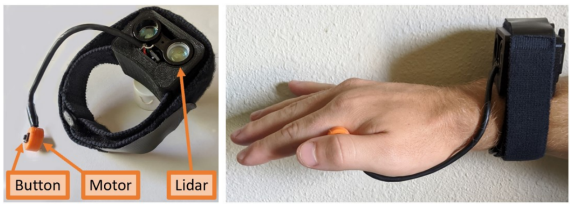
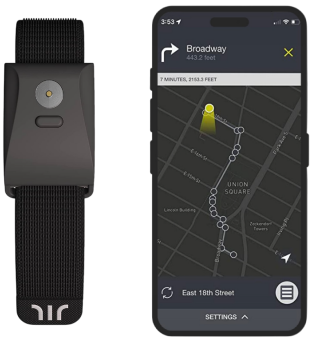



Figure 3: Using the OptiBand to Increase the Long-Range Spatial Perception of People with Vision Disabilities

A prominent commercial product used by visually impaired individuals for global navigation is the Wayband®[1]. This haptic wristband provides hands-free, eyes-free, and ears-free guidance by emitting vibration patterns to direct users to their destinations. The device connects to the HapticNav® app via Bluetooth, utilizing GPS, magnetometer, and compass sensors, along with data from OpenStreetMap, to offer real-time navigation assistance. However, the app is currently only available for Apple devices, limiting its accessibility for Android users. Additionally, the app is designed primarily for sighted users, which may present challenges for visually impaired individuals in terms of ease of use. The global navigation functionality proposed in this thesis is primarily inspired by the WayBand-HapticNav system (Fig. 4), but it incorporates an Android app specifically designed for visually impaired users, with a focus on simplicity and ease of use.




[1] https://haptic.works/wayband, 2025.

Figure 4: WayBand® and HapticNav® iOS App

### 1.3 Outline

The approximate timeline for this thesis, completed part-time during the fall semester of 2024, is illustrated in Fig. (Fig. 5).

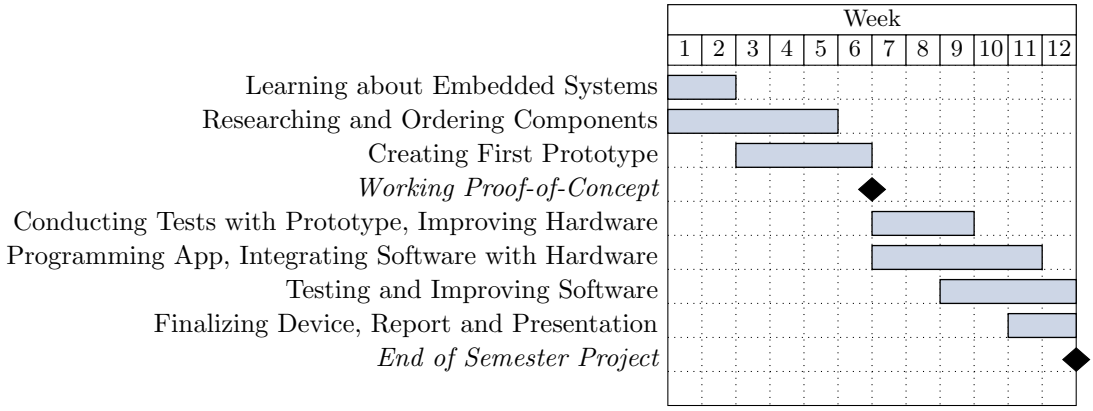



Figure 5: Timetable of the 12 weeks part-time thesis.

## 2 Initial Stages & Prototyping

### 2.1 The Electronics Ecosystem

The main objective of this thesis was to design, develop, and test an assistive wristband for visually impaired individuals under multiple size, price and performance limitations and conditions. The goal was to find a low-cost, easy-to-use solution for creating a discreet and comfortable device that could fit around a human's wrist. According to Karki et al., 2014, wrist circumferences were found to be 16.44 ± 0.97 cm for men and 15.71 ± 0.93 cm for women. Based on these measurements, I aimed to design a wristband that would comfortably fit individuals with wrist circumferences ranging from 14 cm to 18 cm. However, this constraint meant that I did not have much space to work with. As I am not a trained electrical engineer and designing a custom PCB was beyond the scope of this thesis, I had to come up with a way to electronically connect and control the haptic feedback components and the distance sensor using off-the-shelf components. This meant, that I had to analyze which sensors and haptic feedback solutions were suitable for my specific use case (e.g. components that are miniature, energy-efficient, cost-effective, and robust) and to find a compact and practical approach to control the device.

I initially considered using a Teensy 3.2 microcontroller or an Arduino Nano Every. However, my supervisor, Dr. Ronan Hinchet, introduced me to the [2] ecosystem, which offered several significant advantages over my original options. TinyCircuits specializes in creating small, modular, and user-friendly electronic components and development boards. Their products are specifically designed to be space-efficient and lightweight, making them ideal for wearable devices and other compact applications. Moreover, all their boards are fully compatible with Arduino and Raspberry Pi development platforms, which makes them highly suitable for prototyping and experimentation.

The TinyZero processor board (Fig. 6A), for example, features the same 32-bit ARM Cortex M0+ processor as the Arduino Zero but measures just 20 x 20 x 4.4 mm. Furthermore, its modular design enables additional TinyCircuits modules to be stacked directly onto the board without the need for cables (Fig. 7B), which helps keep the device's electronics compact (as most people know, cables can take up a lot of space). This feature allowed me to easily stack a TinyCircuits Bluetooth module (Fig. 6B) onto the TinyZero, saving significant space and enabling the connection to a smartphone later in the project.

[2] https://tinycircuits.com/, 2025.

Another significant feature of the TinyCircuits ecosystem is Wireling, or more specifically the Wireling Adapter TinyShield (Fig. 6D). This module enables the connection of up to four $I^2C$ devices simultaneously using a 4-channel multiplexer. With the ability to stack up to eight Wireling TinyShields, the TinyZero can theoretically connect with up to 32 $I^2C$ devices. Many commonly used $I^2C$ components, such as sensors and displays (all of which draw 3.3 volts), have been integrated into compact (10 x 10 x 3.5 mm), low-cost boards within the Wireling platform. These boards can be easily plugged into the Wireling Adapter TinyShield using relatively thin cables of various lengths (Fig. 7D).

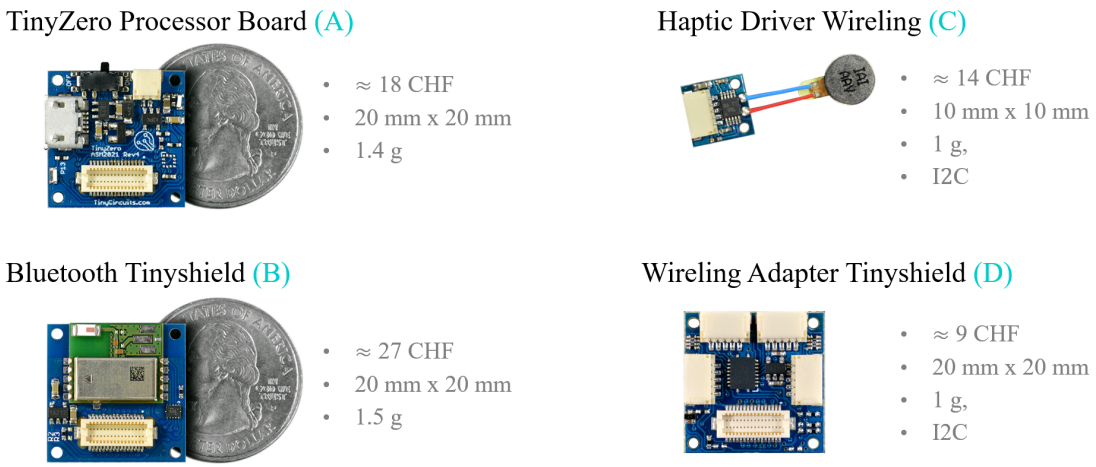



Figure 6: TinyCircuits Components Used in This Thesis: The 20-pin connector port at the bottom of the boards in A, B, and C enables vertical stacking. The four smaller 5-pin ports on the Wireling Adapter Tinyshield (D) can be connected via thin cables to a variety of Wireling boards with attached $I^2C$ devices (C). All components are compact, cost-effective, and lightweight.

## 2.2 Haptic Feedback Implementation

The next challenge was implementing the haptic feedback capabilities of the device. Fortunately, TinyCircuits' Wireling platform offers a ready-made solution with the Haptic Driver Wireling board (Fig. 6C), which features a Linear Resonant Actuator (LRA) that can be controlled via the $I^2C$ bus to create haptic vibration type motion. Coincidentally, I had already experimented with these very motors in the early stages of the thesis. There are few alternatives on the market that are as cost-effective, easy to control, and capable of producing fine-grained haptic effects (These motors can generate over 100 distinct vibration patterns). Given these advantages, it was clear that Haptic Driver Wirelings would be the most effective and intuitive choice for providing haptic feedback to the user.

To test their functionality, I connected four of them to the Wireling Adapter TinyShield and uploaded the same code I had previously used in my Arduino Uno experiments to the TinyZero processor board. As a proof of concept, I wanted to test how well I could feel the vibration of the haptic motors on my wrist. I connected the four Wirelings to my TinyZero-Wireling system and, although it was difficult to gauge the effect without an actual wristband to house the motors, I was still able to make the following observations by pressing them against different spots on my wrist with varying levels of force:

1. It was not possible to actuate more than two motors at the same time due to the power supply being a bottleneck, whether using a 3.7V LiPo battery or connecting the system to a laptop. This meant that only vibration patterns involving a maximum of two motors actuated simultaneously were feasible. In fact, when I later added the distance sensor to the system, this power bottleneck limited me to actuating only one haptic motor at a time.
2. The longer one spot on the wrist was exposed to vibration, the less accurately I could feel changes in vibration intensity. However, if the vibration ceased for a few seconds and then

started again, the sensitivity was noticeably better. From this, I learned that continuous motor actuation, even with low vibration, should be avoided. Instead, the user would perceive changes in vibration more clearly if the motors were mostly off.

3. It was possible to distinct between different vibration source locations on the wrist, but not very precisely. When distributing the four motors around my wrist and letting them vibrate sequentially in 1-second intervals, I could feel the position of the vibration source change. However, to accurately identify whether the vibration was coming, for instance, from the back of my wrist or its right side, the motors needed to be pressed firmly against my skin. If they were only lightly touching my skin, I felt their vibration less clearly and could not pinpoint their exact location. Moreover, I could perceive changes in vibration more effectively when the motors were in direct contact with my skin, rather than separated by any material.

4. I was much better at recognizing patterns than at locating vibration sources on my wrist. Even with the motors loosely attached, I could confidently identify whether the vibration sequence was moving clockwise or counterclockwise. This also worked with other, slightly more complex patterns. My key takeaway was that, in addition to vibration intensity, I could experiment with patterns and, more importantly, with the frequency of turning the motors on and off. This insight proved crucial when fine-tuning the distance-vibration mapping later in the project.

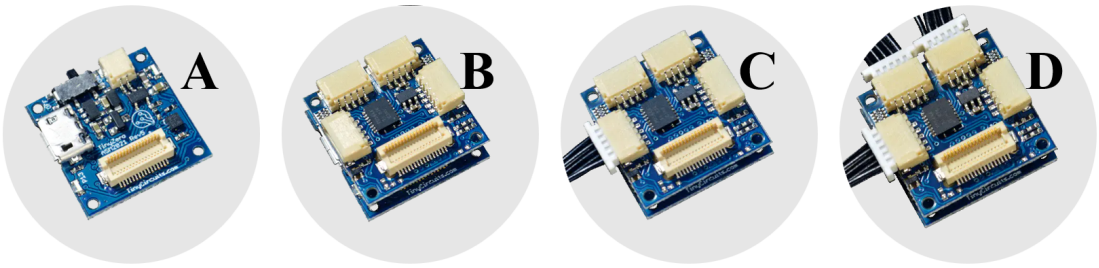



Figure 7: (A) TinyZero Processor Board, (B) Vertical stacking of the Wireling Adapter TinyShield onto the TinyZero, (C) Connection of an individual Wireling board via thin cable, (D) Connection of multiple Wireling boards, with a 4-channel multiplexer managing $I^2C$ signals for each connected board.

## 2.3 Distance Measurement Implementation

When conceptualizing this thesis, I planned to integrate the VL53L1X Time-of-Flight (ToF) sensor from STMicroelectronics (Fig. 10A) as a means to measure distances due to its 4-meter range, widespread availability, compact size, $I^2C$ compatibility, and low cost. Similar to LiDAR sensors, ToF sensors measure the time it takes for an emitted light signal to return and use that data to calculate the distance.[3] I initially tested both the VL53L0X and VL53L1X sensors but found that they performed poorly in ambient light conditions. The wavelengths from sunlight introduced so much noise into the distance measurements of both sensors that their inaccuracies increased by as much as ± meter, which was unacceptable for my application. Given that the device I intended to build needed to function even on a bright, sunny summer day, it was clear that I needed to find a more reliable distance sensor. I began searching for alternatives and avoided sensors that cost more than 50 CHF, exluding for example the Garmin Lidar Lite v3 sensor, which was used by Quick et al., 2023. After intensive research, I came across the TF-Luna LiDAR sensor from Benewake (Fig. 10B), which on first glance met my criteria. The TF-Luna had a range of up to 8 meters, was accurate even in bright lighting conditions, and cost less than 25 CHF. However, it had a field of view (FOV) of only 2 degrees, which was much smaller than the 26-degree FOV of the VL53L1X ToF sensor. This smaller FOV presented both advantages and challenges: On one hand, the TF-Luna sensor provided the user with a more fine-grained understanding of their surroundings due to its narrower FOV. On the other hand, it required the user to move the sensor in various directions to detect potential obstacles. This is an advantage in the sense that objects like a walking cane or the walls of a narrow corridor do not interfere with the detection of objects

[3] https://en.wikipedia.org/wiki/Time-of-flight_camera, 2025.

directly in front of the user. However, it also means that the user must point the sensor precisely at an obstacle in order to detect it. These concepts are visualized and explained in Fig. 8 & 9.

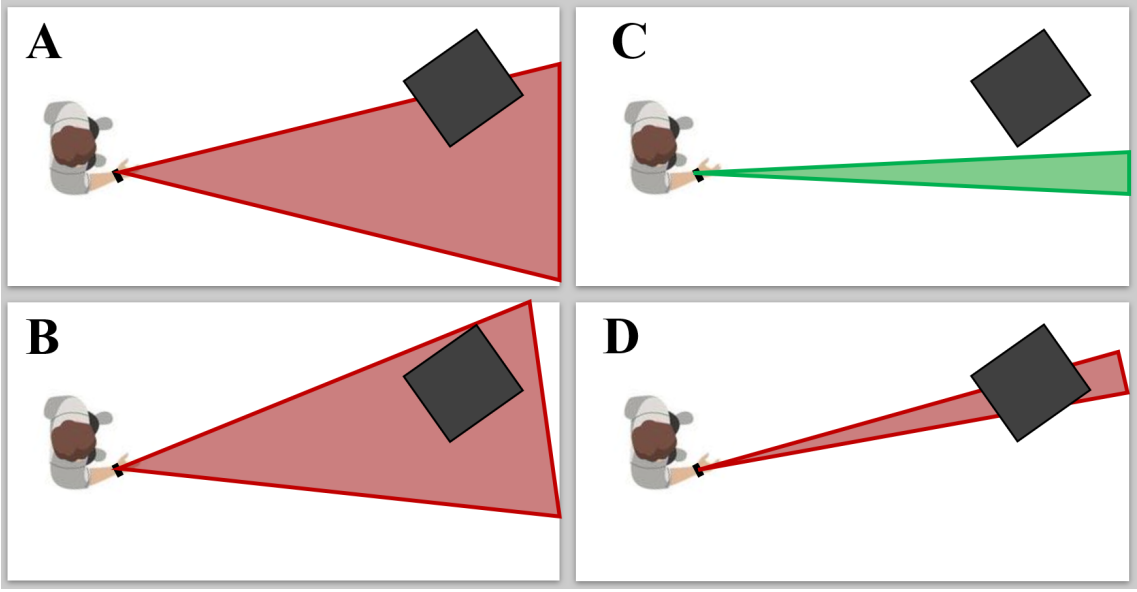



Figure 8: Comparison of sensor field of view (FOV) impact on obstacle detection. The wider FOV on the left increases the likelihood of detecting obstacles regardless of sensor orientation (A & B). In contrast, the narrower FOV on the right may miss obstacles directly ahead (C), requiring frequent sensor reorientation to ensure timely detection (D).

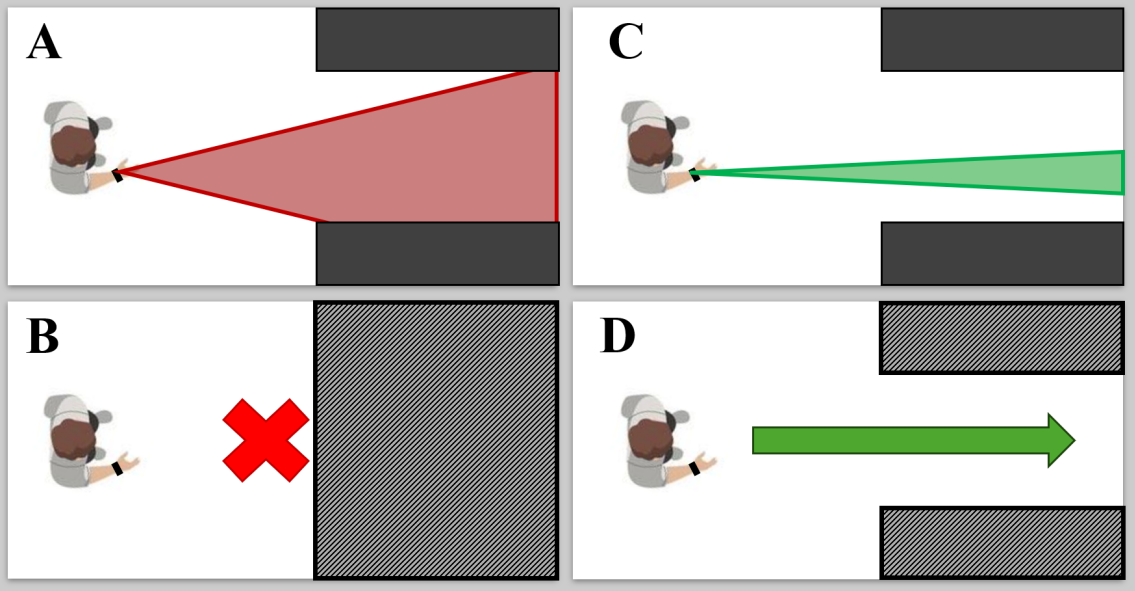



Figure 9: Comparison of sensor field of view (FOV) impact on passage detection. The wider FOV on the left is unable to detect the passage, as its range consistently intersects with one of the side walls (A), leaving the person unaware of the passage's presence (B). Conversely, the narrower FOV on the right allows the person to identify the passage (C) and navigate through it by swinging the sensor to approximate the walls' positions (D).

It was not entirely clear how well visually impaired people would adapt to the fact that they needed to swing the sensor around, but considering that they already do this with their white canes and that the sensor was lightweight, I believed that the narrower FOV would offer more benefits if used optimally compared to a wider FOV. Given this and the fact that no other viable options on the market met all of the following criteria—low cost, small size, $I^2C$ compatibility, accuracy,

resistance to ambient light, and sufficient range—I decided to use the TF-Luna as the distance sensor for my device.

However, connecting the TF-Luna to the Wireling Adapter TinyShield proved to be challenging. The TF-Luna requires 5V to function properly, but the 3.7V to 5V converter in the TinyZero does not supply current to the stack connectors on the TinyShield, a limitation I had not noticed while reviewing the schematics during the planning phase. As a result, I could not provide the TF-Luna with the necessary 5V while simultaneously connecting it via the Wireling interface. While connecting it directly to the processor board was an option, it would have been complicated and somewhat defeated the purpose of using the Wireling interface.

During the week I was testing the TF-Luna, Benewake released another LiDAR sensor (Fig. 10C) called the TFS20-L. This brand-new sensor offered improved range, form factor, accuracy, and better resistance to light conditions (100 Klux) compared to the TF-Luna (70 Klux), and most importantly, it operated at 3.3V. However, at the time, there was no detailed manual or schematic available for this new sensor, and I couldn't know if any of the Benewake Arduino libraries would be compatible with it. At the time of ordering, the TFS20-L cost around 40 CHF. Fortunately, it worked perfectly with the same TFLI2C[4] Arduino library that I used for the TF-Luna sensor. It was impressive how well it performed in bright light conditions and how accurate the distance measurements were, even at distances up to 12 meters. To verify its accuracy, I used a measuring tape to align the sensor and quantitatively check the readings. The key advantages and disadvantages of each tested sensor within the context of my application requirements are illustrated in Fig. 10.




Figure 10: Comparison of distance sensors, highlighting their advantages (green) and disadvantages (red) relative to the requirements of the device.

## 2.4 First Wristband Prototype

With the distance sensor and haptic motors connected to a central processor board, the core electronics for this thesis were complete. The next step involved writing the code to translate the distances measured by the TFS20-L sensor into haptic feedback patterns for the LRA motors. I utilized the Adafruit DRV2605[5] library to interface with the DRV2605 chip on the Haptic Driver Wireling. The complete code implementation for the TinyZero, along with explanatory comments, is available for download here.[6]

To evaluate how my code changes translated into user experience, I created the first prototype of the wristband. As depicted in Fig. 11, I used double-sided Velcro for the wristband's base.

[4] https://github.com/budryerson/TFLuna-I2C, 2025.
[5] https://github.com/adafruit/Adafruit_DRV2605_Library, 2025.
[6] https://drive.google.com/drive/folders/1L98Av-krSZv1qToZuPvuumLcJm3gA-a0?usp=sharing, 2025.

Using a soldering iron, I melted holes into the Velcro to position the haptic motors for direct skin contact. To secure the motors more robustly, I attached their Wireling boards on top with strong adhesive-backed Velcro. This setup ensured that the motors pushed against the Velcro during operation, which improved the transmission of vibration intensity to the wearer's skin. Using the same adhesive-backed Velcro, I attached a battery and the TinyCircuits stack — including a TinyZero processor, two Wireling Adapter TinyShields, and a Bluetooth TinyShield — to the wristband. This method allowed me to move the components along the wristband if needed and therefore to be more flexible in the prototyping process. At this stage, I did not integrate the sensor into the wristband. Instead, I held it between my fingers, pointing it in various directions to observe how the vibration intensities changed based on the distance-to-vibration mapping implemented in the code. With these initial prototyping steps complete, the device was ready for its first round of testing.

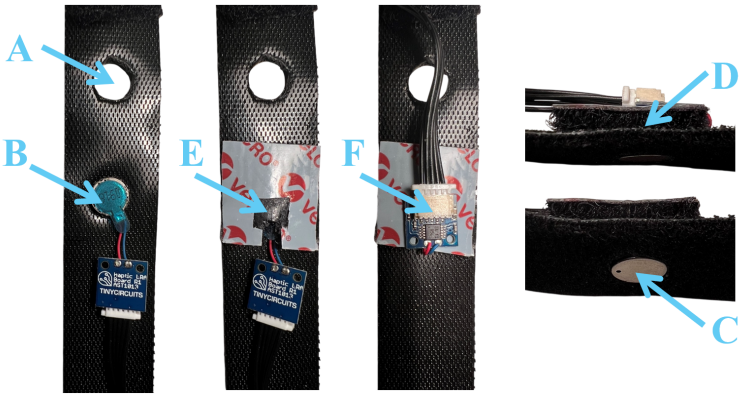



Figure 11: (A) Holes are melted into the double-sided Velcro using a soldering iron. (B) The motors fit perfectly into the holes and cannot fall out on their own. (C) The motors are slightly thicker than the Velcro, creating an "overhang" that ensures constant contact with the skin. (D) A second piece of Velcro is used to attach components to the base Velcro. The lower side of this Velcro is secured to the base Velcro, while the upper side has adhesive material. (E) The second Velcro piece is cut to allow cables to pass through, and a small square on the upper side is cut open to glue the Wireling board. (F) The Wireling board is securely glued onto the second Velcro, with cable tension further pushing the Velcro, and in turn, the motor down. The position of the second Velcro can be easily adjusted.

Again, I gathered several valuable insights:

1. Response Time: I observed how quickly the system responded to changes in distance measurements. The TFS20-L sensor, with its default frame rate of 100 Hz, combined with the lightweight code running on the TinyZero Processor — even with Bluetooth running — allowed the haptic motors to react almost instantaneously. This rapid response with low latency was crucial, as obstacles can move or approach quickly in daily life.

2. Number of Haptic Motors: Using more haptic motors did not significantly enhance my understanding of the environment. Since I could actuate only one motor at a time due to power constraints, the primary advantage of multiple motors was the potential for creating patterns that conveyed proximity. However, I found it more intuitive to emulate the familiar system used in car parking sensors: the closer an object gets, the faster and louder the parking system beeps. Similarly, as an obstacle approaches the TFS20-L sensor, the motor's vibration frequency and amplitude increase. This approach felt more natural than using multidirectional vibration patterns. Consequently, I decided to reduce the number of haptic motors in the device. This not only simplified the system but also allowed me to remove one Wireling Adapter TinyShield, reducing the wristband's height and freeing up space to improve the wristband's fit and usability.

3. Limitations of LiDAR Sensors: I discovered that LiDAR sensors struggle to detect transparent objects, such as windows. Since the light passes through these objects without returning, the sensor cannot measure their distance.

4. Protection of Electronics: During blindfolded tests where I navigated using only the wristband's haptic feedback, I sometimes bumped into obstacles with my hand. This highlighted the need for protecting the electronics on the wristband to ensure durability in real-world usage.

In general, this first proof of concept confirmed that the main idea was viable. By swinging the sensor in a pattern similar to how a white cane is used — but in a vertical motion — I could navigate through corridors, avoid chest-height obstacles, and detect tables and other objects. The increasing vibration frequency and amplitude effectively warned me of obstacles before collisions occurred, even smaller ones.

However, the narrow field of view (FOV) of the TFS20-L sensor required constant and precise scanning in multiple directions to locate obstacles in time. This proved challenging when holding the sensor between two fingers, as I struggled to ensure its orientation and know its direction while blindfolded. This experience made it clear that the next step would be to develop a mechanism that allowed the user to easily put on the sensor, reorient it, and determine its direction, all without visual feedback. But how could I possibly implement that?

# 3 Hardware Optimization & Integration

## 3.1 Distance Sensor Integration

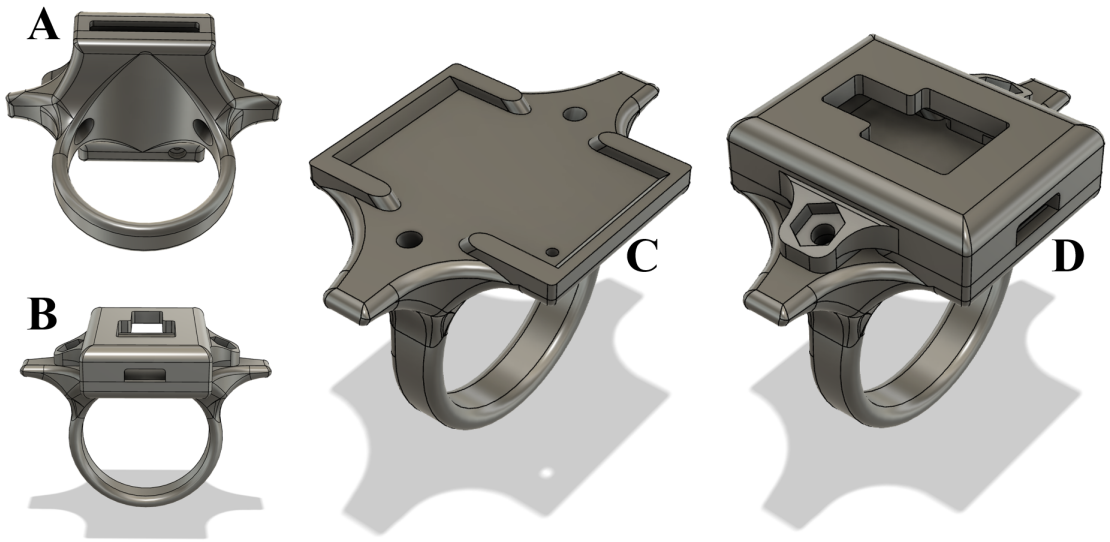



Figure 12: Visualization of the custom-designed sensor housing ring. (A) View from below, showing the long slit opening designed to prevent internal overheating. (B) Front view, showing the slit for the connection wires and the two half-arcs for reorienting the ring. (C) Ring part without the housing; the sensor can be secured by a small screw through the hole in one of the corners. (D) The housing part can be attached with two small screws and nuts, with the light emitter and receiver of the TFS20-L sensor (see Fig. 10) passing through the central opening.

In the early stages of this thesis, I planned to integrate the distance sensor directly onto the wristband, similar to the approach by Quick et al. (2023). However, their sensor placement requires the user to stretch their lower arm horizontally for obstacle detection. When walking with the arm hanging naturally, the sensor in their setup faces the ground, rendering it ineffective as it cannot be rotated. Furthermore, parts of the hand, such as the knuckles, may obstruct the sensor when the wrist is extended (Fig. 3).

To address this lack of flexibility, I decided to develop a custom ring to house the sensor. Rings offer two key advantages: they are easy to put on and remove, and they can be rotated around the finger for directional adjustments. I designed a custom ring[7] for the index finger using Autodesk Fusion, incorporating a housing on top to hold the sensor. This housing could be reoriented based on the user's needs. However, a significant challenge was enabling visually impaired users to determine and set the sensor's orientation without visual cues. To solve this, I designed two half-arcs on either side of the ring (Fig. 12A-D). These half-arcs allow users to intuitively reorient the ring using their middle finger and thumb (Fig. 14 & 15) without requiring visual feedback. I was pleasantly surprised by how natural and intuitive it felt to adjust the sensor's direction with the help of the half-arcs. Also, independent of any wrist movement, the sensor could only potentially be obstructed by the thumb, which gave the user more flexibility to use their hands in comparison to the device introduced by Quick et al. (2023).

The ring consists of two separate 3D-printed PLA components: the ring part and the housing part. Its assembly proceeds as follows: the TFS20-L sensor is first secured to the ring part using a small screw hole, as shown in Fig. 12C. The housing part is then attached to the ring part with two larger screws, protecting the sensor from collisions or damage. A small slit in the housing (Fig. 12A) ensures proper airflow to prevent overheating. An additional slit on the opposite side accommodates the cable connecting the sensor to the Wireling Tinyshield Adapter. Since the TFS20-L datasheet is incomplete, the correct wiring configuration is provided in Fig. 13 for readers who wish to replicate this design or work with the TFS20-L sensor in other projects.

[7] The .f3d file of the ring can be downloaded here.

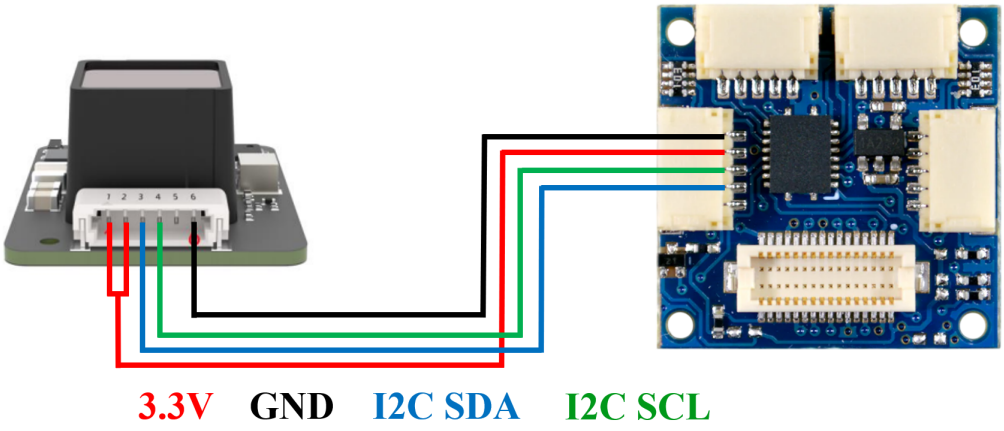



Figure 13: Illustration of correct wiring between the TFS20-L sensor and one of the Wireling Adapter Tinyshield's connection ports. Pin 5 of both the sensor and the Tinyshield should be left floating. Pins 1 and 2 of the sensor must be supplied with 3.3V.

The ring itself is composed of two separate 3D-printed PLA components, that are assembled together: the ring part (Fig. 12C) and the housing part(Fig. 12D). The assembly works as follows: the TFS20-L sensor is first screwed into the ring part via a tiny hole. The housing part is then attached to the ring part using two bigger screws. It protects the sensor from collisions or damage while also securing it to the ring. A small slit in the housing part (Fig. 12A) prevents overheating of the sensor by allowing some airflow. On the other side of the ring there is another slit, which is where the cable from the sensor to the Wireling Tinyshield Adapter goes through. As the sensor datasheet is incomplete, the proper way to wire it is shown in Fig. 13 for readers who wish to replicate this design or work with a TFS20-L sensor for other purposes.

Given the high flexibility of the ring design, I began exploring various ways to use the sensor effectively. I identified and tested two main configurations: First, the sensor can be used similarly to the OptiBand developed by Quick et al. (2023) to actively scan the environment ahead for potential obstacles (Fig. 14). By holding the hand in front, the user can freely choose the direction in which they want to measure the distance. However, the ring design offers greater comfort compared to the wrist-mounted sensor of the OptiBand, as the user only needs to move the wrist or even just the index finger, rather than the entire lower arm, to adjust the sensor's direction. For example, imagine wanting to scan the environment to your right. Using the OptiBand, you would need to fully rotate your lower arm outwards or turn your entire body around. In contrast, with the ring design proposed in this thesis, you would only need to extend your index finger or wrist to obtain distance measurements in directions to your right.



Figure 14: Illustration of ring configuration number one: By extending the lower arm and moving the wrist and index finger, the user can easily scan the environment with the sensor.

As a second configuration, the sensor can be rotated to the radial side of the index finger (Fig. 15), a feature made only possible by the ring's rotating ability. This allows the user to swing their arm naturally while walking and automatically scan the environment straight ahead for potential obstacles. With this setup, users can maintain their usual walking motion without needing to

modify it, while still detecting obstacles outside the reach of their white canes. This unique feature would be impossible with the OptiBand and is one of the main advantages of the proposed ring design. Fig. 15 illustrates how a user walking normally — that means without making any special adjustments to their arm swing — can still avoid objects thanks to the sudden vibration in the wristband upon the objects' detection.

It should be noted that, in addition to these two configurations, there may be other innovative ways to use the ring design that I have not yet considered. Also, the ring solution has some drawbacks: first, it requires individual size adjustment to fit each user, and second, the system consists of two components (the ring and wristband) rather than a single, fully integrated unit. Nonetheless, in my opinion, the advantages of this approach far outweigh these limitations.

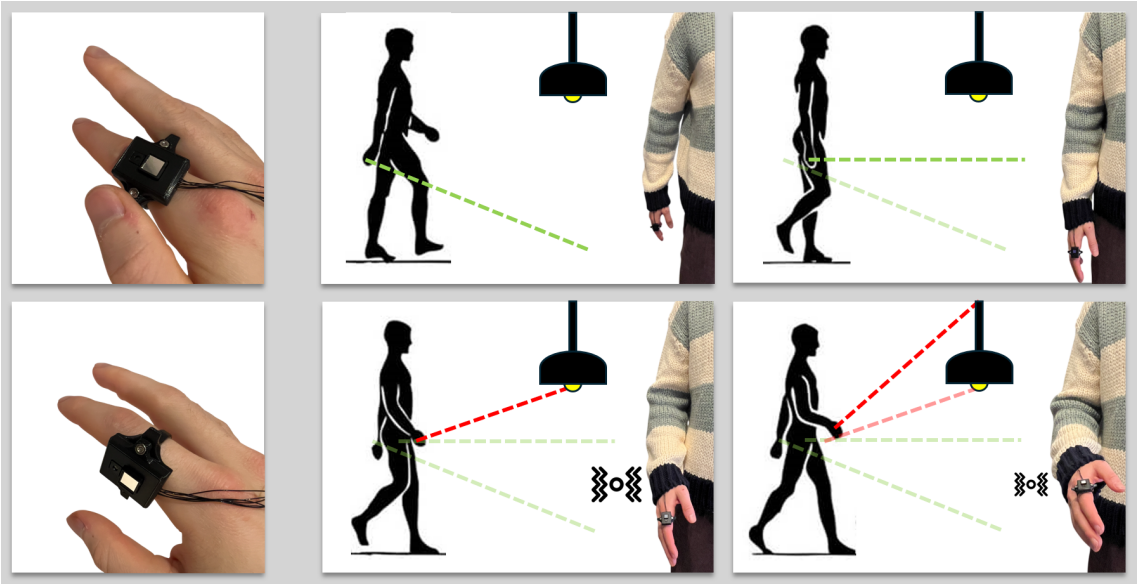
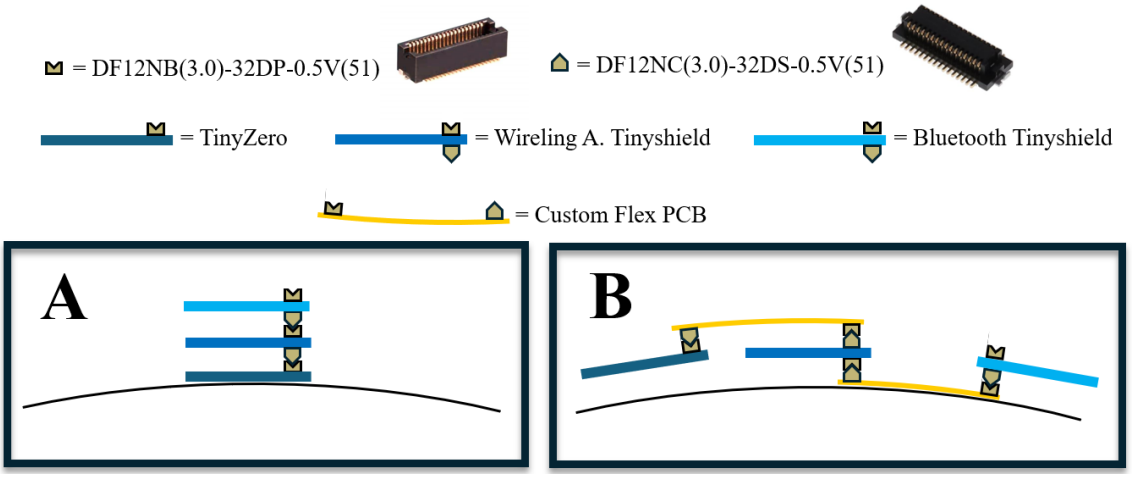


Figure 15: Illustration of ring configuration number two: The ring can be rotated using the thumb and middle finger to face the radial direction of the hand. This way, obstacles straight ahead can be detected during normal walking motion. A large portion of the space in the plane in front of the human avatar[8] is scanned during each step. The more the arms swing during walking, the more space is scanned for potential obstacles. Depending on the proximity of a detected obstacle, the intensity of the vibration changes. This allows users to navigate around obstacles, without needing to change their behavior extensively.

## 3.2 Use of Flexible PCBs




[8]Source Material: human avatar

Figure 16: Depiction of vertically stacked TinyCircuits boards (A) and horizontally connected boards using custom-designed Flex PCBs (B). The Flex PCBs enable a flatter wristband design while providing potential space for integrating the battery and Haptic Driver Wirelings beneath the boards. The two connectors shown in the legend are used on each TinyCircuits board for stacking. This same mechanism is used in (B) to link each board to a Flex PCB, and in turn, to another board. This configuration could allow for the addition of more Wireling A. Tinyshields without increasing wristband height, as the boards can be evenly spread around the wrist.

One improvement I wanted to implement was flattening the wristband to make it more ergonomic and discreet. To achieve this, I came up with the idea of connecting the three TinyCircuits boards (TinyZero, Wireling Adapter Tinyshield, and Bluetooth Tinyshield) "horizontally" rather than stacking them vertically. To do this, I planned to connect the 20-pin ports (*DF12NB(3.0)-32DP-0.5V(51)* and *DF12NC(3.0)-32DS-0.5V(51)*) of each board using custom-designed Flex PCBs (Fig. 16). I chose Flex PCBs because they are made from a flexible, non-rigid material, which allows them to bend. This is important because the boards needed to be arranged to follow the curved shape of the wrist.This was my first time designing a PCB, and it turned out to be a valuable learning experience. The schematic and final design of the Flex PCB are shown in Fig. 17. As seen in Fig. 17A, it is a simple PCB that essentially connects the 20 pins of the two connectors found on the TinyCircuits boards.

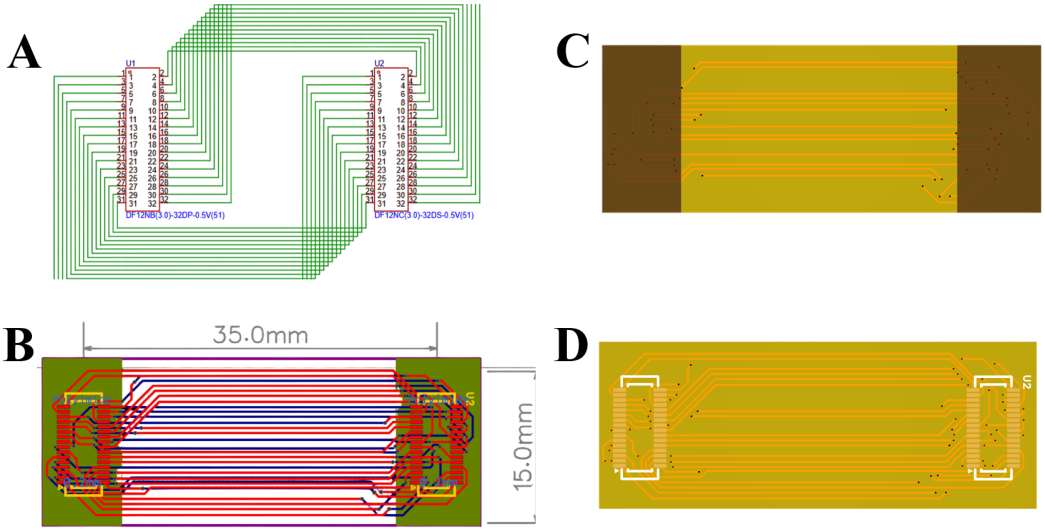



Figure 17: (A) Schematic of the custom-designed Flex PCB. (B) PCB Layout & Routing: The red connection lines represent traces on the top layer, while the blue connection lines correspond to the bottom layer. (C) Bottom Plate: Stiffeners (brown) are added in areas where connectors are placed to prevent breakage during use. (D) Top Plate: Both connectors are assembled on this side to reduce manufacturing costs.

However, when I attempted to order these custom-designed Flex PCBs, I discovered that having SMT components, such as these connectors, pre-assembled onto the Flex PCBs is quite costly. Table 1 breaks down the total price of 153.14€ for an order of 5 PCBs. The Flex PCBs themselves were not the major cost, as they would have only amounted to 26.15€. The high cost primarily stems from the connector assembly onto the Flex PCBs, which is a process that requires specialized PCB assembly fixtures (which for my case cost 91.31€). According to JLCPCB's website[9], the high cost is due to the fact that "*Flex boards are prone to wear and tear as they are thin and lightweight. To successfully assemble SMT components, rigid carriers are used. The positioning and consistency of the carrier are crucial to the assembly process. Many auxiliary fixtures are required for flex assembly, including board carrying trays, baking, electrical testing, functional testing, and cutting fixtures.*"

But ordering just the Flex PCBs and assembling them myself was not possible. The connector pins, with a width of just 0.25 mm and a pitch of 0.5 mm, were too small to solder onto the

[9] JLCPCB article on PCB Assembly Fixtures, 2025.

Flex PCBs using the tools available in the Soft Robotics Lab. Therefore, given that I wanted to keep the device low-cost, I had to abandon the idea of using Flex PCBs to flatten the wristband. However, if production prices for Flex PCBs with preassembled SMT parts decrease in the future, this approach could be reconsidered.

| **PCB Price** | **26.15 €** |
|---|---|
| Engineering Fee | 7.75 € |
| Via Covering | 0.00 € |
| Surface Finish | 15.69 € |
| Board | 2.71 € |
| **PCB Assembly Price** | **127.00 €** |
| Setup Fee | 24.21 € |
| Stencil | 7.61 € |
| Panel | 0.00 € |
| Large Size | 0.00 € |
| Components (2 items) | 0.00 € |
| Feeders Loading Fee | 2.91 € |
| SMT Assembly | 0.50 € |
| PCB Assembly Fixture | 91.31 € |
| Packaging Fee | 0.46 € |

Table 1: Table listing the charge details of the Flex PCB order. PCB assembly costs account for 82.93% of the total price. This high cost is due to the need for PCB assembly fixtures, which are required to solder SMT components onto the Flex PCBs. In comparison, the PCB assembly cost for the same PCB, but rigid, is only around 16.11€.

## 3.3 Protection of Electronics

As mentioned in Section 2.4, the electronics inside the wristband prototype were not protected against potential collisions. Since people with and without sight use their hands constantly in daily life to perform various tasks, the risk of accidentally bumping the wristband into an object and damaging the electronics cannot be ignored. I determined that a soft protective layer was needed above the electronics to: 1) protect the internal components of the wristband, and 2) create a smoother, softer outer surface that would improve user-friendliness and discretion.

For my first prototype, I molded silicone into a custom shape that would fit nicely around the electronics. However, silicone is not very effective at conducting heat (its thermal conductivity typically ranges from 0.2 to 0.4 $W/m \cdot K$[10]), which is detrimental to heat dissipation within the wristband. Additionally, most people who tried it during user testing said that the material felt a bit too soft. Moreover, it is difficult to create holes in the silicone mold, which is necessary for accessing the micro-USB port on the TinyZero for charging, for example.

Given these issues, I decided to switch to printing the protective layer with soft TPU. TPU is a better conductor of heat[11], making it more suitable for preventing overheating. Additionally, TPU has a texture that provides a higher-quality feel in my opinion, and it is much easier to print holes in the design using support material. After learning how to print with TPU and completing several design iterations, I finalized the protective TPU layer[12], which is shown in Fig. 18.

[10] https://www.intertronics.co.uk/wp-content/uploads/2016/11/TB2007-12-Thermally-Conductive-Silicones.pdf, 2025.
[11] https://expafol.com/en/thermoplastic-polyurethan/#:~:text=TPU%20and%20silicone%20are%20comparable,cases%20to%20stop%20devices%20overheating., 2025.
[12] https://drive.google.com/file/d/1mUrbuqx_S5opnoBCO50tqZI3zzIQzrNP/view?usp=sharing, 2025.

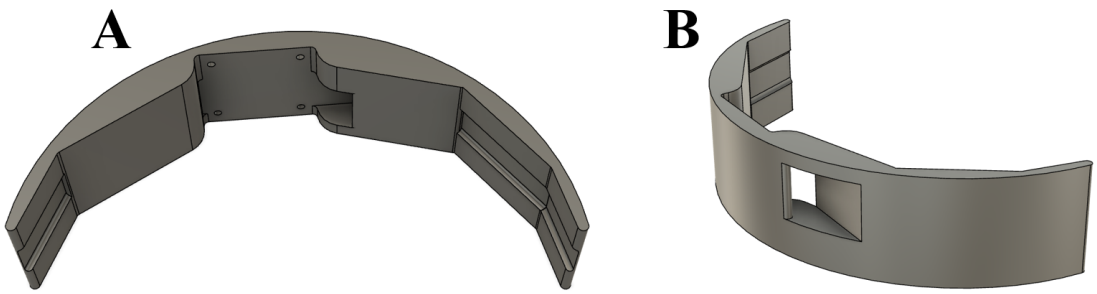



Figure 18: CAD design of the TPU protective layer, shaped in an oval form to better conform to the wrist. The varying heights inside the layer are designed to accommodate and separate the different electronic components of the wristband (Fig. 19). In (A), four holes are visible through which pins are inserted to secure the TPU layer laterally. Additionally, rails along the sides help adjust the orientation of the TPU layer when pressed onto the wristband. In (B), a hole is provided for a micro USB cable to charge the wristband. The TPU layer is printed with 20% infill.

## 3.4 The Final Hardware Design

Before the addition of the TPU layer, the wristband consisted solely of electronic components glued onto Velcro strips, which in turn were attached to the base Velcro band. This assembly method is exactly the same as that depicted in Fig. 11D. Fig. 19 provides an illustration of the internal layout of the wristband, showing how the electronic components are distributed along the base Velcro in the final design.

(A) Haptic driver Wireling for generating vibration signals on the left side of the wrist. (B) TinyCircuits stack, consisting of TinyZero, Wireling Adapter TinyShield, and Bluetooth TinyShield. Note the four pins mounted onto the stack, which limit the movement of the TPU layer attached above. (C) 3.7V 290mAh battery, enabling untethered use of the device. (D) Haptic driver Wireling for generating vibration signals on the right side of the wrist. Two distinct vibration signals are needed to convey more global navigation information, as discussed in section 4.3 on the final software design. All components are mounted onto one-sided Velcro, which is attached to the base Velcro, as shown in Fig. 11D.

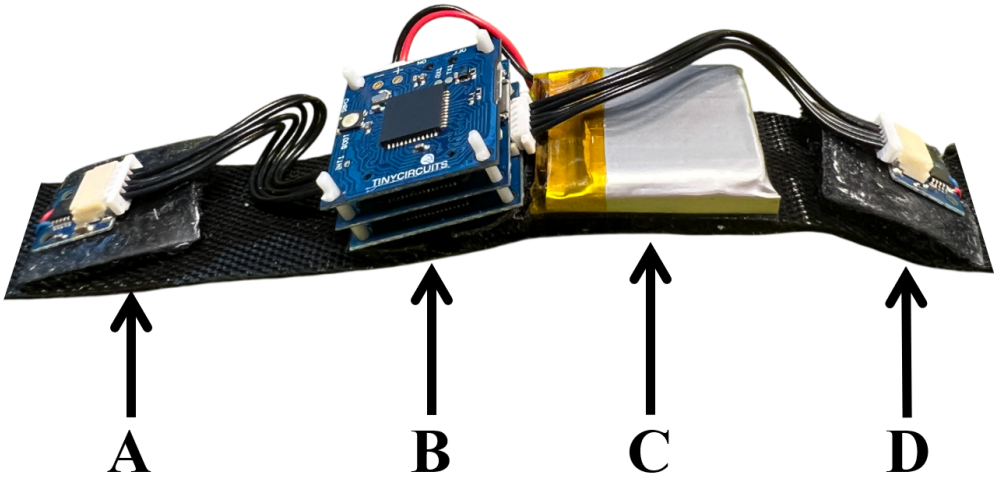



Figure 19: (A) Haptic Driver Wireling to give Vibration signal on the left side of the wrist. (B) TinyCircuits stack consisting of TinyZero, Wireling Adapter Tinyshield and Bluetooth Tinyshield. Note the 4 pins mounted onto the stack, that allow for limiting the movement of the TPU layer. (C) 3.7 Volt 290 mAh Battery allows for untethered use of the device. (D) Haptic Diver Wireling to give Vibration signal on the right side of the wrist. Two distinct vibration signals are used to indicate more complex information, which is more discussed in section 4 on Software. All components are glued onto one sided velcro, which in turn is attached to the base Velcro, same as explained in Fig. 11D.

When incorporating the TPU protective layer (Fig. 20), I noticed that it was not adequately secured to the electronics. It could shift laterally when subjected to external impacts, although vertical movement was later mitigated by wrapping a soft fabric around the wristband. To address this lateral movement issue, I designed 3D-printable pins that fit into the four holes of the stacked TinyCircuits boards (see Fig. 19B) and the TPU layer (see Fig. 18A). These pins ensured that the TPU layer was securely attached to the wristband and limited its lateral movement and wobbling in general.




Figure 20: The TPU protective layer is attached to the wristband, with four pins holding it vertically in place. Each electronic component is protected from significant impacts. No TPU is present on the sides to allow for better heat dissipation. (A) Full setup showing how the components are nested inside the TPU protective layer. (B) Detailed view of the charger hole.

The next step in the hardware design finalization was to develop a mechanism for tightening the wristband. The design needed to be intuitive and easy to use, particularly for visually impaired individuals. I chose a 3D-printed ring attached to one side of the Velcro, through which the other side of the Velcro could be threaded (Fig. 21). This mechanism allows users to tighten the wristband using only one hand. It is also easily adjustable, making it suitable for people with various wrist sizes.

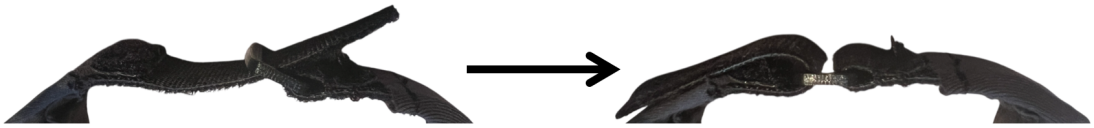


Figure 21: Simple tightening mechanism for the wristband, designed to be adjusted with just one hand. It accommodates wrist sizes ranging from 14 to 18 cm. A small Velcro strip is sewn onto the wristband (on the left side in both pictures), allowing for a robust fixation of the wristband's size

As part of the final design refinement, I also wanted to add a discreet, soft covering to the entire wristband. This covering would serve multiple purposes: securing the components, visually concealing them, improving the device's appearance to resemble a professional product rather than a prototype, and increasing the comfort of the wristband on the skin securing the components. The material needed to be soft, lightweight, heat-conductive, and visually appealing. After evaluating several options, I selected an elastic, soft fabric that I sewed around the wristband (Fig. 22A). The fabric allowed for easy insertion of holes, including one for the charger, and contributed to a polished, professional finish in my opinion (Fig. 22B).




Figure 22: Depiction of the final wristband design. (A) Soft elastic fabric is sewn around the wristband to secure and conceal the inner components. (B) Additional fabric is glued into the charger hole, giving the device a more professional appearance. The entire wristband measures no more than 2.3 cm in height and maintains a discreet look due to its neutral colors and smooth shape.

As a final hardware improvement, I decided to also add a protective layer around the cables connecting the TS20-L sensor to the wristband. To achieve this, I first wrapped the cables with insulation tape and then covered them with the same soft, elastic fabric used for the wristband (Fig. 23). While I considered using a flexible cable duct, tests showed that it was too stiff for this application. Maintaining cable flexibility was crucial to allow users to move their index finger freely and naturally. At the same time, the cable coating had to be robust enough to ensure users could easily put on the ring and wristband without the risk of damaging or tearing the cables. This is why I opted for the "insulation tape plus soft fabric" approach, as it ensured both the flexibility needed for natural finger movement and the durability required to prevent damage or tearing when putting on the ring and wristband.

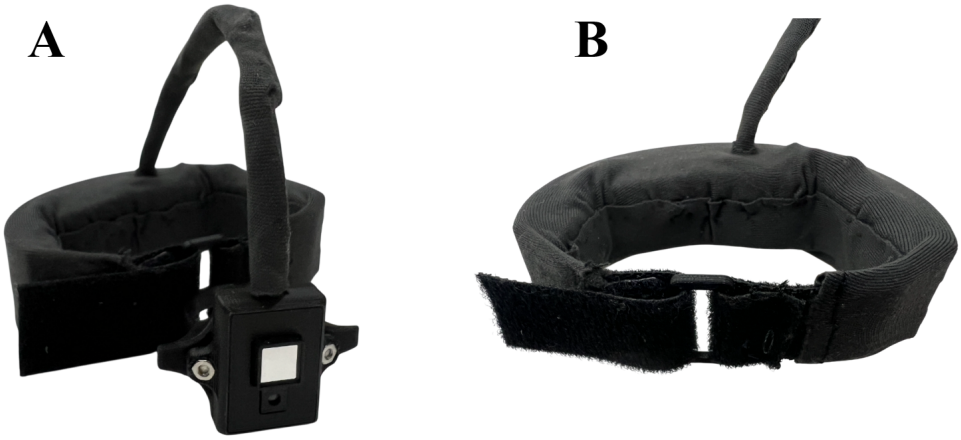



Figure 23: Depiction of the sensor ring attached to the wristband. The thin connection cables are reinforced by wrapping them with insulation tape. The same soft, elastic fabric is glued over the insulation tape to give the connection piece a more professional appearance. (A) The length of the connection cables is chosen so that they are nearly stretched when the wrist and index finger are fully flexed. (B) The cables pass through a tiny hole in the fabric on the side of the wristband and connect to the Wireling Adapter TinyShield inside.

The complete final hardware setup, shown on the hand in various configurations, is depicted in Fig. 24. Table 2 provides an overview of the main components and price-driving factors, which contribute to our estimated total cost of 127.19 CHF for the wristband as of the time of writing this thesis. This is significantly cheaper than the devices introduced by Quick et al. (2023), Zahn and Khan (2022), Katzschmann et al. (2018), and the WayBand (249 USD[13]) which was one of

[13] https://www.amazon.com/WAYBAND-Navigation-Accessibility-International-Skateboard/dp/B0BKK2XK42, 2025.

the key objectives of this design.

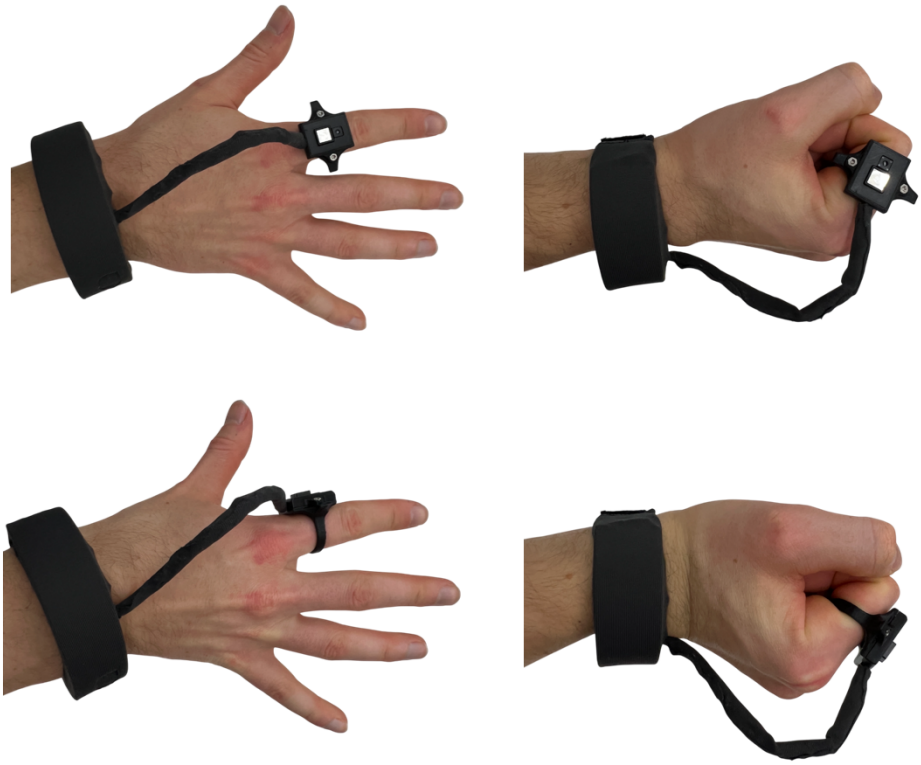


Figure 24: Examples of four different configurations for using the device. Note that the cable connection is never fully stretched, allowing for natural and unrestricted movement of the index finger.

| **Component** | **Quantity** | **Price (CHF)** |
|---|---|---|
| TS20-L Sensor | 1 | 39.19 |
| TinyZero Processor Board | 1 | 18.00 |
| Bluetooth Tinyshield | 1 | 27.00 |
| Wireling Adapter Tinyshield | 1 | 9.00 |
| Haptic Driver Wireling | 2 | 28.00 |
| 3.7V 290mAh Battery | 1 | 6.00 |
| **Total** | | **127.19** |

Table 2: Main components and price-driving factors for the hardware. Notably, the device is very low-cost in terms of raw components compared to similar works.

# 4 Software Implementation & Integration

## 4.1 Why build a Phone Application?

It may be argued that a mobile application for steering the device would not significantly benefit the visually impaired community, as navigating apps can be challenging for them. However, this thesis posits that a phone app offers substantial advantages when designed with accessibility and ease-of-use in mind. By integrating a Bluetooth Tinyshield module into the hardware (costing 27 CHF), the device can connect to a phone, unlocking additional capabilities and enhancing user experience drastically. While the Bluetooth module increases power consumption and slightly adds to the device's bulk, its benefits far outweigh these drawbacks in my opinion.

First, vibrational feedback can be extended from the wristband to the phone. For instance, if an object is detected at extremely close proximity, both the wristband and the phone can vibrate to alert the user. Second, the Bluetooth connection enables offloading computations and utilizing geographical data directly on the phone, capabilities that would be unfeasible on the TinyZero board alone.

This thesis demonstrates that the functionalities of the HapticNav app (Fig. 4) — a geolocation-based iOS application that guides users along predefined navigation paths using vibrational feedback — can be qualitatively replicated and, in some aspects, even enhanced. (However, a quantitative comparison between both apps is beyond the scope of this thesis.) The proposed app is designed for Android, features a user interface (UI) tailored specifically to the needs of visually impaired individuals and expands the application capabilities of the wristband by enabling functionalities that would otherwise be impossible to implement.

## 4.2 How to build a Phone Application?

Programming mobile applications can be a tedious task. Since the two dominant operating systems are iOS and Android, many enterprises must develop two entirely separate apps—one using Xcode (iOS's programming IDE) and the other using Android Studio (Android's programming IDE). However, in recent years, a third approach has emerged that allows developers to efficiently create a single codebase for an app that can be deployed simultaneously on both iOS and Android devices. This solution is known as Expo.

Expo is an open-source platform for building universal native apps for Android, iOS, and the web using JavaScript and React. Its core idea is to enable developers to write an app once while ensuring it runs seamlessly across multiple platforms. By providing an abstraction over native code, Expo allows developers to focus on app functionality without needing to manage platform-specific code, making it a powerful tool for building cross-platform apps.

Expo offers two primary methods for testing code on simulated or real devices:

1) **Expo Go**: Expo Go is a mobile app (available on the App Store and Google Play) that enables developers to run and preview their Expo app during development. However, Expo Go supports only the modules included in Expo's standard libraries. If an app requires custom native code or third-party modules that necessitate native installation, Expo Go cannot be used for testing.

2) **Expo**: Expo itself is a framework and toolset for building and deploying React Native apps. Built on top of React Native, Expo includes a collection of prebuilt native components and APIs for common functionalities, such as camera access, push notifications, Bluetooth, and GPS location.

In this project, I required native modules, such as GPS localization and Bluetooth Low Energy (BLE), which cannot be used with Expo Go. Although Expo Go is the simpler option for testing during development, I had to use the full Expo framework to meet the native API requirements of my app.

Another important consideration was determining the target platform(s) for development. Upon researching, I discovered that deploying an app to iOS devices—even for personal testing—requires membership in the Apple Developer Program (99 USD annually) and access to a macOS device,

neither of which I had. Consequently, I decided to develop the app solely for Android while ensuring the code remained adaptable for future iOS deployment. I hope that this allows for easy expansion to iOS if necessary, with minimal additional implementation efforts in any potential future updates of the project.

To install the latest version of this app on a physical phone, newcomers to Expo should first follow this guide[14] to set up Expo and all related libraries. This process also involves downloading either Android Studio or Xcode and configuring them to work with Expo[15]. Extensive documentation is available to guide users through setting up an Expo development environment, and the entire process can typically be completed in less than a day, even for those with little coding experience.

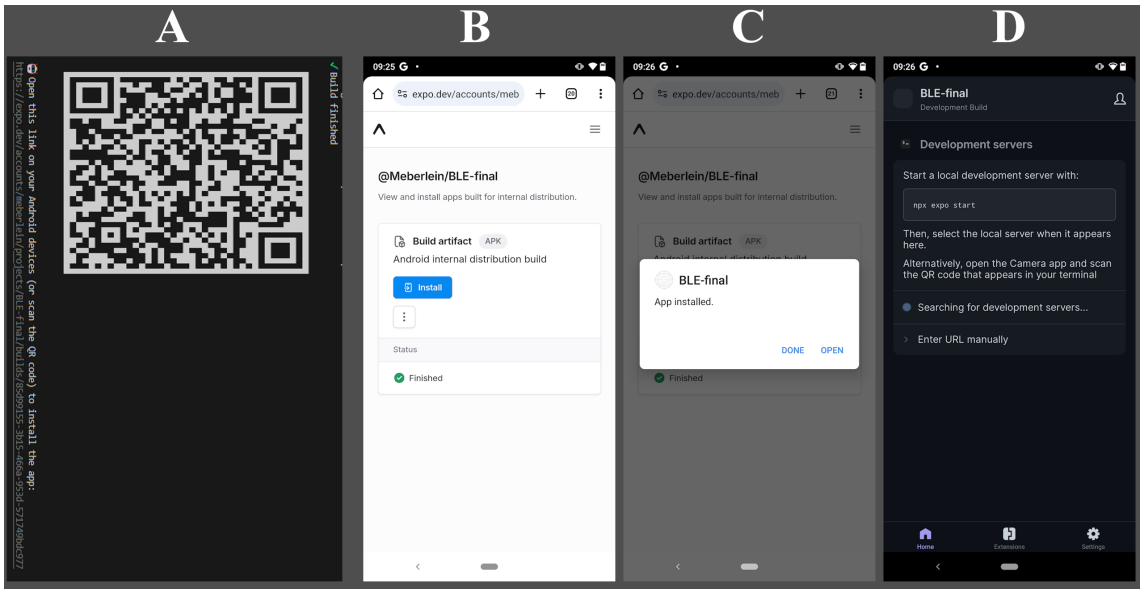



Figure 25: (A) A QR code and URL link are generated in the CLI on the computer, after building the project (image rotated 90 degrees). (B) Scanning the QR code or entering the link on an Android phone opens a website. (C) From the website, the app artifact can be downloaded and opened on the phone. (D) The server connecting the app on the phone to the code on the computer can now be started. Note that this process differs slightly for iOS devices.

```
# Navigate to a desired project folder
cd some_folder

# Clone the GitHub repository of the app
git clone https://github.com/maximilianeberlein/ble-final.git

# Navigate into the cloned repository folder
cd BLE-final

# Open the project in the code editor (e.g., VS Code) to make modifications
code .  # Edit code as needed

# Prebuild the project for Expo (e.g., generating native code for dependencies)
npx expo prebuild

# Build the app using Expo Application Services (EAS) for Android with the
# development profile. This can take a while.
eas build --profile development --platform android
```

[14] https://docs.expo.dev/workflow/overview/, 2025.
[15] https://docs.expo.dev/workflow/android-studio-emulator/, 2025.

After a few minutes, a QR code will be generated and displayed in the CLI (Fig. 25A). This QR code can be scanned with an Android phone, which will open a link (Fig. 25B) from which the app artifact (Fig. 25C) can be downloaded. Once the artifact is opened, the development build of the app will launch directly on the phone (Fig. 25D). Next, a local development server must be started, which enables us to load the app with the most recent code. (Note: the computer running the CLI commands and the phone with the app artifact must be connected to the same network for these steps to work.)

The local development server can be started by typing the following into the CLI:

```
# Start the Expo development server to run and preview the app
npx expo start
```

This generates another QR code (Fig. 26A), which can be scanned to open the fully functional native app in a development build (Fig. 26B). The development build enables us to debug specific features directly within the UI. It also allows for instant updates on the real phone device whenever changes are made to the code. For example, if I change the background color of the app in VSCode, I can immediately see the updated color on my physical phone after a fast refresh. This is just one of many features that Expo provides, which facilitate rapid app prototyping and a fast development workflow in general.

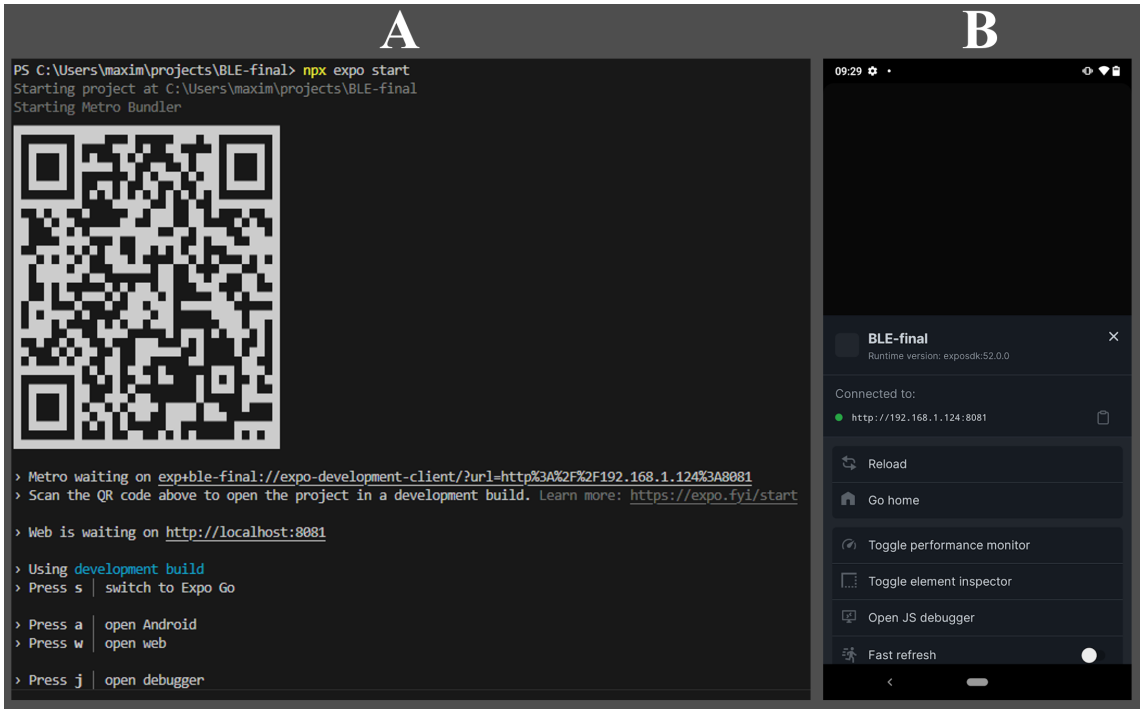



Figure 26: (A) A QR code and URL link are generated after starting the Expo development server. (B) Scanning the QR code or entering the link on an Android phone loads the app. The app has access to all native functionalities of the phone, such as Bluetooth, camera, and GPS localization. Note that both the phone and the computer must be connected to the same network for this process to work.

With these steps, the app should be up and running. There are many tutorials available online for getting started with Expo. I recommend first creating a custom app from scratch before importing the app proposed in this thesis and building upon it. While the app iteself could potentially be distributed via Google Play with minimal effort, this was beyond the scope of this research-oriented thesis.With that said, we are now ready to finally delve into the core functionalities, architecture, and UI design of the app.

## 4.3 The Final App

The final app consists of two main UI screens: the Home Screen (Fig. 27A) and the Main Screen (Fig. 27B) . This section provides an in-depth explanation of both screens, their functionalities, and the design choices made to support the needs of visually impaired users.

**Home Screen**: The home screen serves as the app's entry point, opening immediately when the user launches the app. The interface features a large, dark-gray button that spans the entire screen so that visually impaired users can easily locate and interact with it. When pressed, this button initiates the process of connecting to the Bluetooth Tinyshield using Bluetooth Low Energy (BLE).

When using the app for the first time, the user will be asked to grant permissions for Bluetooth usage, as well as other features like GPS localization. The Bluetooth Tinyshield's device name is "BlueNRG." Once the "Connect to HAPPI" button is pressed, the app searches for a device with this name using Bluetooth. If the connection is successful, the user is taken from the home screen to the main screen, and both the phone and wristband vibrate in a "success" pattern to confirm the connection. However, if no device is found or the connection attempt fails after 10 seconds, the phone vibrates for 400 milliseconds (ms) to indicate "failure".

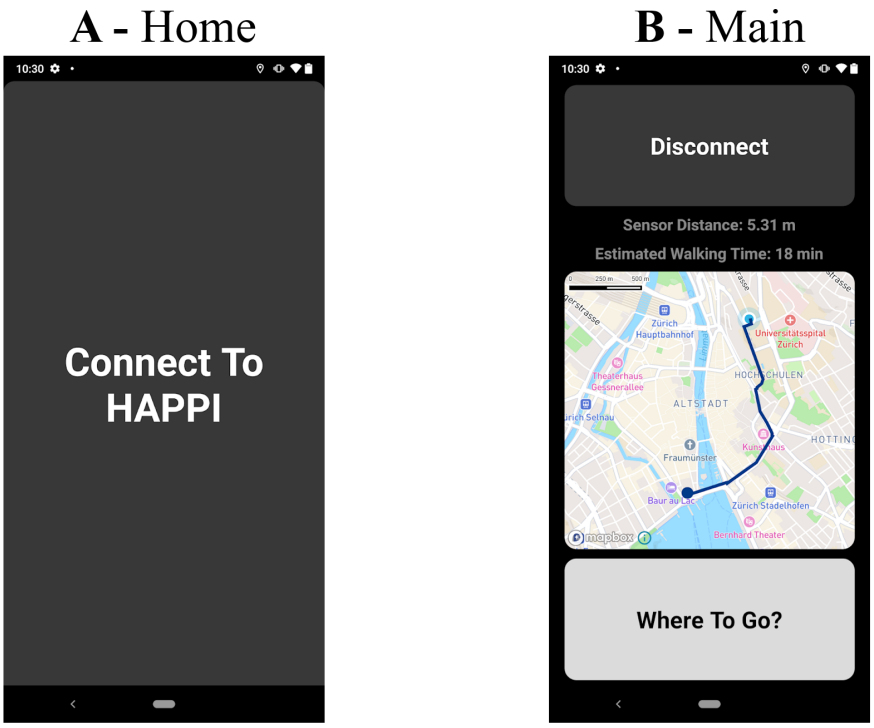



Figure 27: (A) Home screen: The home screen features a single, large button that spans the entire screen, allowing users to easily connect to the wristband (coined HAPPI device, which stands for Haptic Assistance for Pathwayfinding and Personal Independence). The oversized button helps visually impaired users quickly locate it. (B) Main screen: The main screen contains three primary components: the Disconnect button, an interactive map that displays the user's position and navigation route, and a button for setting a destination via voice command. Each interactive element in the app is maximized in size for accessibility and provides haptic feedback through the phone or even wristband to confirm whether the user's action was successful. Additionally, the destination-setting feature provides audio feedback to offer the user more detailed information on Mapbox's response to their address request.

Vibration feedback is a key feature of the app and is used throughout to provide visually impaired users with intuitive signals about the app's status. Most of the app's design choices have been made with visually impaired individuals in mind. The BLE connection process is quick and reliable, and in my testing, I rarely encountered issues with connecting to or disconnecting from the wristband. The logic behind this process is implemented in useBLE.ts in the GitHub repository for this app[16].

[16]GitHub Page of the App: https://github.com/maximilianeberlein/HAPPI-wristband.

The wristband itself is mostly agnostic to this process. However, once a connection between the wristband and phone is successfully established, the app sends a simple *"1"* over Bluetooth to the TinyZero microcontroller, which continuously checks for incoming Bluetooth data. When it detects the *"1"*, it "knows" that the connection was successful and triggers a "success" vibration pattern with the haptic motors, providing the user with immediate feedback regarding the app's status on the wristband as well.

**Main Screen**: The main screen is more complex than the home screen, as it contains the app's core functionality. However, it was crucial for me to ensure that it remained easy and intuitive to use for the visually impaired community. Therefore, the two large buttons on main screen each occupy roughly one-fourth of the screen. One button is light-gray and placed at the bottom, while the other is dark-gray and positioned at the top. These buttons are spaced apart both spatially and by color to help visually impaired users easily locate and interact with the correct button. When a button is pressed, the phone vibrates slightly to provide haptic feedback, confirming that the button was found and successfully pressed.

If the dark-gray button is pressed, the phone disconnects from the wristband and returns the user to the home screen. If the connection to the wristband is lost another way (for example, due to a drained battery), the app automatically returns to the home screen.

With this final auxiliary feature explained, we can now dive into the core functionality of the app: global navigation assistance. Similar to HapticNav, this app allows users to set a geographical destination, for which the most efficient pedestrian route is calculated. The main contribution of both apps is their ability, to provide haptic feedback - also in conjunction with a wristband - that guides the user along the calculated route. However, the question arises: how can users with limited eyesight easily set a navigation destination in the app, and how can they effectively follow the path to it?

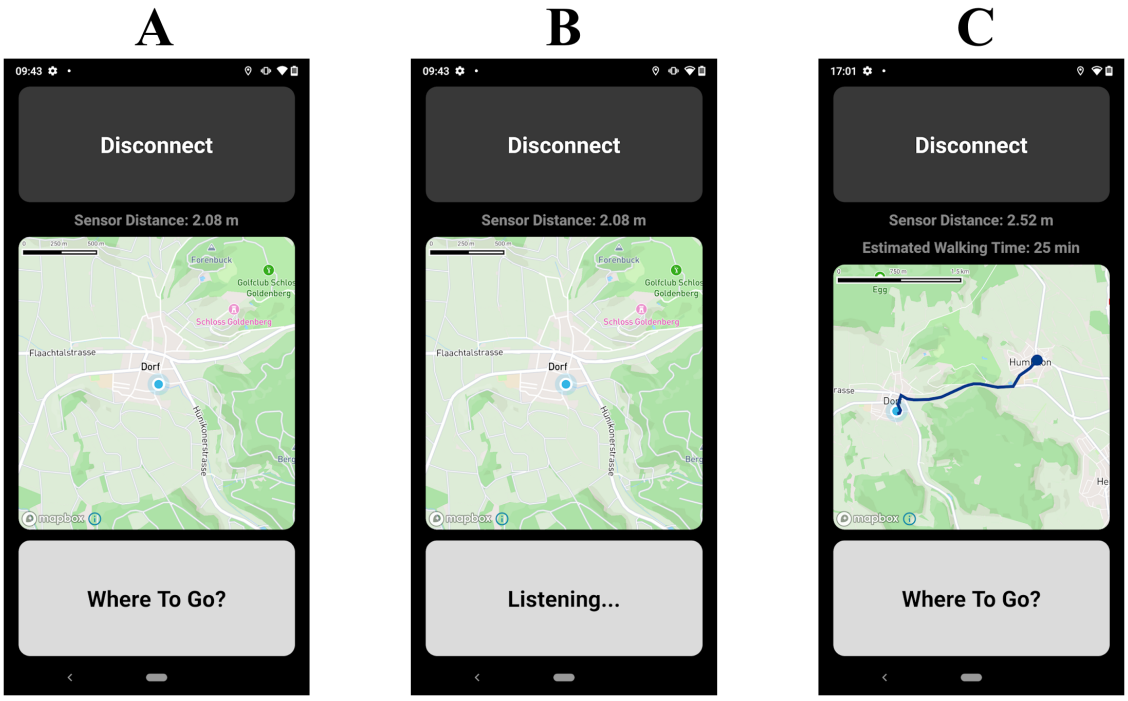



Figure 28: (A) The initial main screen after connecting to the wristband. (B) After pressing the 'Where to Go?' button, the app begins listening for audio inputs from the user to capture the desired address. Once the user has stopped speaking, the address is parsed and searched by Mapbox. (C) If the address is recognized and within 20 km, Mapbox calculates the most efficient pedestrian route and starts guiding the user to it with haptic cues on both the phone and wristband

I adopted a two-pronged approach. First, the user can press the large white "Where To Go?" button on the bottom of the screen (Fig. 28A) to activate the device's voice assistant, which

listens for audio input (Fig. 28B). The user can then say the desired destination, such as "Zürich, Bürkiplatz." The voice assistant converts the audio input into text, which is then passed to various Mapbox libraries and APIs for processing.

Mapbox is a powerful mapping and location platform that enables developers to integrate custom maps, geolocation, and navigation functionalities into their applications. It offers detailed map data, APIs, and SDKs for both mobile and web development, making it a popular choice for navigation-based projects. Initially, I considered using the Google Maps API for navigation, believing it to be the most accurate. However, I found that Google Maps charges based on the number of API requests made each month.[17] For example, 2,000 destination requests in a month would already cost around 50 USD.

In contrast, Mapbox only charges when monthly destination requests exceed 100,000.[18] As a result and also because it is compatible with Expo and well-documented, I opted to use the Mapbox SDK for this project. If someone wishes to use or extend this app, they will need to generate their own Mapbox Access Token and paste it into line 25 of *App.tsx* and line 38 of *app.json* in the GitHub repository, as I cannot share my token publicly.

To my surprise, the Mapbox Directions API is really accurate, offering pedestrian paths as fine-grained as those generated by Google Maps. In many cases, it even directs users to crosswalks instead of simply guiding them across streets. The Directions API has an extensive library of addresses, and using simple keywords like the town name and street name is usually enough to find the correct destination. For example, saying "Zürich, Bürkiplatz" allows the app to set the destination correctly (Fig. 27B).

But what happens if the voice assistant does not correctly identify the address, or if the address is located across the Atlantic Ocean? Based on the audio input, the app can respond in several ways:

1. If the audio input does not match any address in the Mapbox library, the phone will vibrate in a 'failure' pattern (on-off-on-off-on, each lasting 200 ms), and the voice assistant will announce: "No valid navigation path found for parsed address from audio input. Try again."
2. If the audio input matches an address in the Mapbox library, but that address is farther than 20 kilometers away, the phone will vibrate in the same 'failure' pattern, and the voice assistant will say: "parsed address from audio input is too far away. Try again."
3. If the audio input matches an address in the Mapbox library and that address is within 20 kilometers, the phone will vibrate in a 'success' pattern (a short, light vibration), and the voice assistant will reply: "Setting navigation goal to: parsed address from audio input. Estimated walking time: walkingTimeMinutes minutes."

This way, the user always receives non-visual feedback on whether a valid destination was successfully set. An the good thing is that even if app users have the sound turned off or are in a noisy environment, they will still know whether setting the destination was successful, thanks to the haptic feedback provided by the phone. The estimated walking time needed to reach the destination is calculated by the Mapbox API and is also visualized in the UI above the map for users with only slight visual impairments.

There is also a second way to define a navigation path in the app. By pressing and holding on a point on the map for a longer period, a route will be calculated to that location using the same Mapbox procedure as with voice input. As before, both vibrational and audio feedback are provided, depending on whether a route to the selected location can be set and whether it lies within 20 kilometers. The need to press and hold a point on the map reduces the likelihood of accidentally changing the destination by tapping the screen.

But why is this second method of setting a destination necessary? First, individuals with slight visual impairments may still be able to set an approximate destination directly on the map, which is helpful if the address of the destination is unknown. Additionally, visually impaired users could

[17] https://mapsplatform.google.com/pricing/, 2025.
[18] https://www.mapbox.com/pricing, 2025.

more easily get help from sighted individuals—for example, by asking them to find a destination on the map and set it for them in the app.

Unlike HapticNav, which only allows navigation to be set via text or audio input, this app offers the flexibility to set the navigation goal directly on the map. Additionally, as shown in Fig. 29A, the UI bar in the HapticNav app (located at the top of the screen) for setting a destination is very thin, making it easy to miss for visually impaired users.

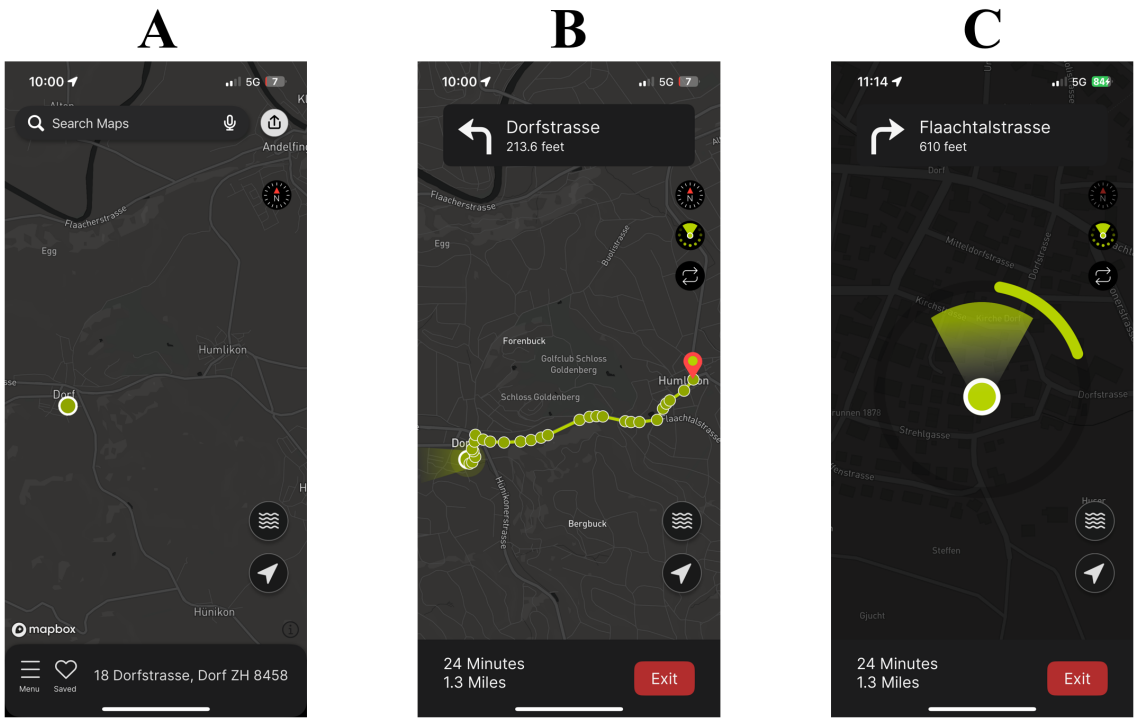



Figure 29: Screenshots from the HapticNav application. (A) The UI overview after launching the app. It is not optimized for visually impaired users, as the icons are small and lack haptic or audio feedback when interacted with. (B) After setting a route via text or speech, the route is calculated. (C) The user follows the route by orienting the phone so that the two green visual indicators align, requiring the user to hold the phone and match their own orientation with it.

There is another significant difference between the HapticNav app and the app proposed in this thesis regarding how haptic feedback is used for route following. HapticNav relies on the orientation of the phone to guide the user in the right direction (Fig. 29C). If the discrepancy between the desired orientation to follow the route and the user's actual orientation (as determined by the phone's orientation) is too large, the phone vibrates. The larger the discrepancy, the stronger the vibration. While this method works, as claimed in several reviews, and could help users once they adapt, it requires the user to always hold the phone facing forward. Otherwise, if the phone lay for example in a sidepocket, its orientation would not match the actual orientation of the direction the user is facing. However, since visually impaired users often need to hold a white cane for navigation, obstructing their free hand with a phone may need limit them in the execution of other tasks.

The larger the discrepancy, the stronger the vibration. While this method works, acoording to several reviews, and could be helpful once users adapt, it requires the phone to always be held facing forward. If, for example, the phone is placed in a side pocket, its orientation would no longer align with the direction the user is facing. Since visually impaired users often need to hold a white cane for navigation, using a phone with their other hand would obstruct both their hands, and thus limit their ability to perform other tasks.

To solve this issue, I chose a different approach. I initially implemented the same orientation-based feedback feature from HapticNav, but given the focus on the visually impaired community, I

decided to develop a more suitable method. Instead of relying on the phone's orientation, I decided to utilize the user's relative position to the desired path.

I implemented two ways of providing haptic feedback to correct the user's steps and guide them back to the desired path. First, if the user moves more than 25 meters away from the closest point on the route, the **phone** will vibrate for 300 milliseconds to alert the user that a new route will be recalculated. If the user is within 25 meters but more than 3 meters off the path, they will receive haptic feedback through the **wristband** to guide them back. Any position within 3 meters of the path is considered to be on track, as Mapbox directions or GPS localization are not always accurate to the meter.

However, if the user veers too far left of the path (i.e. more than 3 meters), the right-side haptic motor on the wristband will vibrate (Fig. 30B). The farther the user strays left, the stronger the vibration on the right motor will be (Fig. 30C). Similarly, if the user moves too far right, the left motor will vibrate, guiding them back or at least closer to the path (Fig. 30A). The system works by mapping the user's distance from the path's closest point to the vibrational intensity of the motors. If the distance is between 3 and 25 meters, it is mapped to a range from *"10"* to *"19"* for the left side of the path and from *"20"* to *"29"* for the right side. This value is then sent via Bluetooth to the TinyCircuits stack, which reads it to activate the correct motor and adjust its vibration intensity accordingly. The intensity is determined by the second digit of the mapped value, with *"9"* corresponding to maximum vibration and *"0"* to almost no vibration, while the first digit indicates which motor should be actuated (*"1"* for right, *"2"* for left).

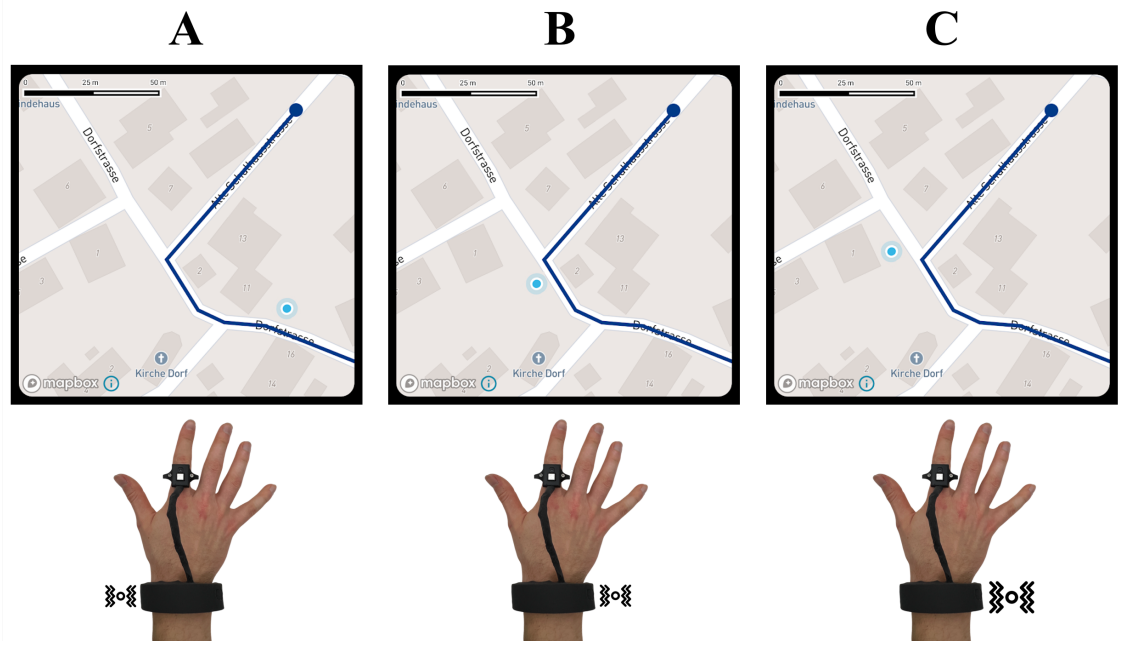



Figure 30: Illustration of the proposed path-following method. The final destination is represented by a dark-blue point, while the user's current position is depicted as a light-blue point. (A) The user is positioned too far to the right of the street, prompting a slight vibration in the left wristband motor to guide the user to adjust their steps to the left. (B) The opposite scenario of (A), where the user is too far left of the street. (C) If the user moves farther left from the path, the right wristband motor vibrates stronger, indicating an increasing deviation. This method works intuitively as long as the user is heading towards the final destination. However, if the user turns around and starts moving in the opposite direction, the left-right guidance may become misleading. The change in vibrational intensity could help mitigate this issue, as it would signal an incorrect orientation. In (C), for example, if the user turns around, they would receive a signal to move right to approach the path, but the vibration intensity would continue to increase instead of decreasing, signaling the user is oriented incorrectly.

The effect of this proposed method is similar to the orientation-based solution from HapticNav, with the key difference being that the user does not need to hold the phone but must in turn remain somewhat aligned with the correct direction. If the user moves in the opposite direction to

the designated route, for example, the directional cues from the wristband will be exactly inverted, leading to incorrect guidance. For instance, if the person in Fig. 30C were to turn around 180 degrees, the path would suddenly be on their left side, while the wristband would instruct them to move more to the right. One potential solution to this would be to track the user's geographical displacement over time and infer their orientation from that. However, despite the code using the most precise localization settings available, frequent position fluctuations make this approach unreliable.

This is where varying vibrational intensity comes into play. Even if a user is not facing the correct direction, walking around will still change the intensity of the vibration, which can guide them back toward the path. This is a hypothesis that requires testing with real visually impaired users, but in theory, combining information on whether the user is to the left or right of the path together with vibrational intensity changes could allow the user to intuitively adjust their steps to follow the designated path.

Some individuals with slight visual impairments or even sighted people might be interested in the exact distance measured by the sensor. However, this distance is calculated within the TinyZero code and not directly accessible to the user. Fortunately, it is not only possible to receive values via Bluetooth with the Bluetooth Tinyshield, but it is also possible to send them. Each time a measurement is taken with the TFS20-L sensor, its value is transmitted to the phone via Bluetooth and displayed in a Textbox below the Disconnect button on the main screen. While this feature can be fun to experiment with in different scenarios, it's more of an added bonus than a primary feature of the app.

In summary, when using the app, the wristband provides both local and global navigation assistance. If the phone is not connected, the wristband still works perfectly well, but it only offers support for local navigation, such as obstacle detection and avoidance. However, when the app is connected to the wristband, the haptic feedback from the wristband — which normally indicates the distance to objects in front — will be interrupted by an additional haptic signal that updates the user on their position relative to the global navigation path every second and after every meter walked. It is still unclear how quickly and effectively users can adapt to this type of bi-functional local and global navigational feedback system, and further user testing is needed, which leads us to the final section of this thesis: The conclusion and potential further work,

# 5 Conclusion & Further Work

After improving the hardware and developing an app to steer the device, my next step was to conduct a user study. As sighted individuals, my colleagues and I found it challenging to predict how well visually impaired people could utilize this device. None of us had experience using a white cane, and since this device is designed to serve as an additional assistive tool to the cane, we could not measure its impact on local and global navigation accurately. Although we were able to detect avoid obstacles, and to follow predesignated paths while blindfolded, it remained unclear how much faster or safer navigation would be with the device compared to without it.

As I believe that quantitative results are crucial for evaluating the device's impact, I sought permission from the ETH ethics commission to conduct a user study with visually impaired participants. However, the commission stipulated a minimum two-month waiting period for approval, which extended beyond the deadline for this semester thesis. As a result, this report does not include a user study. Nevertheless, I am keen to continue the project beyond this thesis, conduct a user study, and potentially publish any relevant findings.

While this thesis cannot quantitatively assess the device's performance, I can qualitatively affirm its utility and advantages. The wristband is low-cost, lightweight, comfortable, and relatively discreet. It is easy and intuitive to use, does not occupy the hand and gives assistance in local and global navigation tasks. However, there is still some room for improvement, such as incorporating flexible PCBs or a custom PCB to replace the TinyCircuits stack and developing a wireless sensor ring to eliminate the connection cable running above the hand for an even more discreet look. The wristband also has practical limitations: it is difficult to repair due to its sewn fabric design, it is not waterproof, and it has not been tested for prolonged use beyond one hour.

On the software side, developing an iOS-compatible app and distributing both iOS and Android versions through their respective app stores could significantly broaden accessibility. Additionally, the app has not yet been tested by visually impaired users, whose feedback could be invaluable in identifying opportunities for improvement and the implementation of additional features. While there is potential for commercializing this project, it is important to emphasize that thorough user feedback is essential before pursuing such efforts. Within the scope of this thesis, however, I believe the primary objectives outlined at the beginning of this report have been successfully achieved.

Another direction to explore is incorporating the TinyZero's IMU to provide orientation data, which could improve route-following by comparing the device's actual orientation changes to the orientation adjustments required for accurate global navigation. Additionally, integrating a second sensor with a wider FOV into the ring could enhance the device's obstacle detection capabilities. The user could more passively scan the environment with the wide FOV sensor (i.e. without swinging the ring) and, upon detecting an obstacle, switch to the TFS20-L sensor with its narrower FOV for a more detailed analysis of the obstacle's geometry.

In general, this project presented numerous engineering challenges that demanded creativity and critical problem-solving to address. Academic projects often place less emphasis on user-friendliness, but this thesis prioritized the end user, designing the device around the needs of the visually impaired community. This work involved building a soft wristband from scratch, integrating multiple electrical components, designing custom rings and protective layers, developing an app tailored for visually impaired users, and, most importantly, combining all these elements into a cohesive system.

Throughout this project, I gained a variety of valuable, new skills: programming apps with Expo, 3D-printing soft TPU, working with $I^2C$ devices, writing BLE connection code, and designing user-friendly, wearable robotic systems. However, the most significant challenge — and therefore the greatest learning experience — was conceptualizing and integrating these diverse components into one functional device that met all of the requirements and needs defined in the beginning.

This project has been a deeply rewarding experience, and I am excited about the prospect of improving this device further. Assisting visually impaired individuals through haptic feedback remains an area full of potential, and I look forward to iterating on this idea in the years to come.

# 6 Acknowledgements

I would like to express my profound gratitude to my supervisor, Dr. Ronan Hinchet, for his exceptional guidance and support throughout this thesis. His invaluable advice and openness to discussions on the best approaches to implementing various features were instrumental to the project's success. I am also grateful to my co-supervisor, Prof. Dr. Robert Katzschmann, for granting me the opportunity to pursue this thesis and for his support in its organization. A special thanks goes to my family, Cléli, and my friends for their encouragement throughout the semester. Their assistance in testing the device and providing constructive feedback was crucial to its development. Lastly, I want to emphasize how much I enjoyed working on this thesis. It has been a really rewarding experience, and I eagerly anticipate potential continuation and publication efforts for this project.

# References

Karki, B., Bhattarai, M., Bajracharya, M., Karki, S., & Devkota, A. (2014). Correlation of neck and wrist circumference with waist circumference. *Journal of Advances in Internal Medicine*, *3*(2), 47–51 (cit. on p. 3).

Katzschmann, R., Araki, B., & Rus, D. (2018). Safe local navigation for visually impaired users with a time-of-flight and haptic feedback device. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *PP*, 1–1. https://doi.org/10.1109/TNSRE.2018.2800665 (cit. on pp. 1, 2, 17).

Quick, R., Bontula, A., Puente, K., & Fitter, N. T. (2023). Using the optiband to increase the long-range spatial perception of people with vision disabilities. *2023 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 248–254. https://doi.org/10.1109/RO-MAN57019.2023.10309546 (cit. on pp. 2, 5, 10, 11, 17).

Zahn, M., & Khan, A. A. (2022). Obstacle avoidance for blind people using a 3d camera and a haptic feedback sleeve. https://arxiv.org/abs/2201.04453 (cit. on pp. 1, 17).