Semester project report on ”SafeStep”:

Submitted by: Team A – Group 1

Ayman Fatich, Tarek Saade, Alexis Balestra, Aya Mekkass, Romain Georgenthum, Sifeddine Louzar

Under the guidance of : Antonio Faonio

25/09/2023 - 29/01/2024

Table of contents:

1- Introduction :

This semester project is about developing a system capable of guiding a visually impaired person through an obstacle course. Our main goal is to make visually impaired people feel free and independent in their daily life and in the practice of some sports. Our system utilizes a Raspberry Pi, a LiDar sensor to detect presence of an object in real-time, and a haptic output. The device computes the distance between the sensor and the object, and warns its user with various vibrations through the haptic output. It is a multidisciplinary project as it requires many skills in both hardware and software. It has also helped us develop some soft skills (teamwork, working with deadlines, explaining our work in an efficient way…). In this paper, we will examine various components, from our python code and the logic behind it to the different mapping models tested. We will then explain the reason of our choice through testing results. Finally, we will outline our future work.

2- Method/Implementation/Design:

The hardware architecture consists of three components. First, the Raspberry Pi serves as the central processing unit to organize all computational tasks. Second, the Lidar sensor, whose role is to detect distances and communicate this data to the Raspberry Pi. Finally, the haptic motor responds to instructions from the Raspberry Pi to generate vibrations. The sensors are connected to the central unit by means of the input and output ports using cables.

The implementation strategy of the device involves its integration with gloves, optimizing user interaction and convenience. The Raspberry Pi is

strategically positioned atop the haptic feedback motor, ensuring direct contact with the user's hand. The Lidar sensor is thoughtfully situated at the device's forefront to enhance ease of use and bolster its ability to detect obstacles surrounding the user.
This meticulous attachment to gloves not only facilitates user comfort but also maximizes the efficiency of the device in navigating various environments.

The functional design embodies a practical and intuitive approach.Programmed vibrations occur at specific intervals, and the duration of these intervals dynamically adapts based on the proximity of detected obstacles. This design guarantees both user-friendly operation and operational efficiency, enabling users to promptly ascertain the relative distance of obstacles at a glance.
The intentional simplicity of the device offers users immediate and clear feedback concerning the proximity of objects, whether they are in close proximity, at an intermediate distance, or positioned at a considerable distance from the user.

<u>3- Source Code:</u>

The initial segment of our source code is designed to interpret distance measurements from the LiDAR sensor, and this portion was adapted from the LiDAR's official documentation.

Within our program, an endless loop serves as the primary structure. At the beginning of this loop, the system captures the distance data from the LiDAR. Subsequently, the duration since the last vibration event is determined using the time library. Following this, the corresponding delay for the measured distance is computed using the `calculate_delay` function. Should the elapsed time since the last vibration exceed this calculated delay, the system triggers a vibration alert.

An alternative approach could have involved imposing the delay by pausing the code execution with the `time.sleep()` function for the duration of the delay. Nevertheless, our chosen methodology enhances the device's responsiveness, as it allows for the delay to be updated in real-time.

For instance, if the LiDAR sensor initially detects an object at a considerable distance and then suddenly identifies a closer object, the delay is adjusted instantly. This immediate update would not be feasible with the `time.sleep()` function.

Throughout the development process, we refined the `calculate_delay` function multiple times. We established that distances below 10 centimeters or beyond 8 meters should be standardized as 10 centimeters and 8 meters, respectively. The rationale is that the LiDAR sensor cannot detect objects beyond 8 meters, and readings under 10 centimeters are deemed unreliable and excessively proximate for the user.

The accompanying graph elucidates how a specific function is applied based on the detected distance. We segmented the range of possible distances into three categories: from 0 to 0.6 meters, from 0.6 to 5 meters, and from 5 to 8 meters.

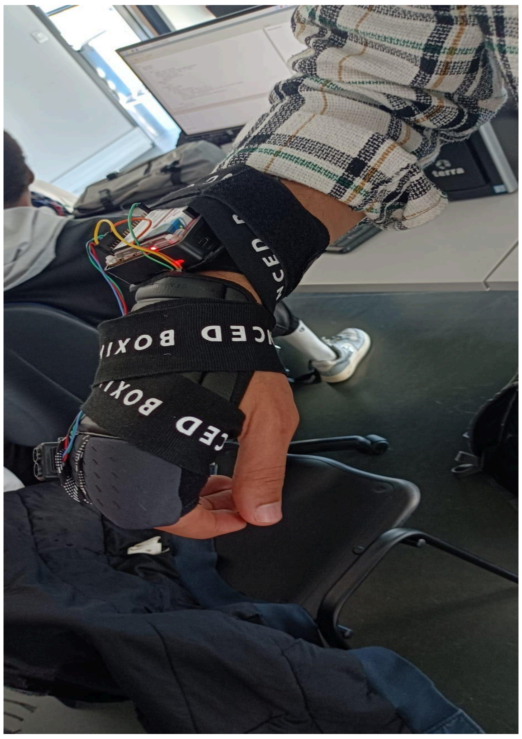
- Within the 5 to 8-meter range, the correlation between distance and delay is linear, ensuring the delay between vibrations is sufficiently long to alert the user to distant obstacles without being intrusive.
- From 0.6 to 5 meters, the object's proximity necessitates increased user awareness, hence our choice of a logarithmic function.
- Below 0.6 meters, the situation demands immediate action; therefore, a continuous vibration is emitted to prompt an urgent response.

In addition, the system is programmed to inform the user when the battery level is low.

<u>4- Experiments and testing:</u>

4-1- Setup:

As shown below, the user is equipped with the device scratched into its hand and put in front of many obstacles. We did run many tests over and over again. After some failed tests, we re-adjusted the code in order to detect more moving objects. On a track of over 6 meters, with multiple obstacles we used **SafeStep** to navigate successfully blindfolded



4-2- Results:

Our results were of over 96% of accuracy, meaning we only hit less than 4% of obstacles, but for obstacles over 0.4 meters of height, safe step detected 100% of them.

4-3- Analysis:

Our results indicate that vision impaired individuals may use our device in a safe way. It might be possible to have 100% accuracy if the user is trained and has experience with the device.

7- User manual:

Instructions for users:

Charge the battery before usage using the included charger, or any 15w charging port, including your phone. The device turns on immediately after being powered and keeps working as long as its battery is not empty. Charging takes about 3 hours, while usage on battery takes 10 hours. You can either use the device standalone, or connected to a laptop, smartphone or power bank. Do not put it in contact with water as it will damage the device.

Updating the firmware:

-Open up the device

-Connect it by ethernet

-Identify its IP on your router settings

-Connect to ssh using its ip and the username “SafeStep” and the password “admin”

-replace the main.py file in the $HOME path with the new script, do not change the name or else it won’t work

-reboot the device