



## **Smart stick for blind people**

**Graduation project is submitted to Computer and Systems Engineering  
Department in Partial fulfilment of the requirements for the Degree of  
Bachelor of Computer and Systems Engineering**

Made by:

***Eslam Mohamed Ashour Mohamed***

***Asmaa Khaled Mohamed Motawie***

***Rehab Fakhry Gibreel Mohamed***

***Sara Salah Ahmed Mohamed***

Supervised by: **Assoc. Prof. Amr E. Mohamed**

## **Acknowledgment**

We are deeply grateful for the support we have received throughout our Graduate project. We would like to extend our heartfelt appreciation to our supervisor, Assoc. Prof. Amr E. Mohamed, for his insightful comments, helpful information, practical advice, and continuous flow of ideas have been instrumental in the success of our project. We recognize that without his support and mentorship, this project would not have come to fruition. And we are truly privileged to have had the opportunity to work under his guidance, and we express our gratitude for his continued encouragement throughout this journey.

## **Keywords**

- Visually impaired persons
- Blind people
- AI
- Socket.io
- Object detection
- YOLOv8
- Speech Recognition
- Android mobile application
- Firebase
- Google Maps

## **Abstract**

The Smart Stick is an innovative device designed to empower individuals with visual impairments in their daily lives and navigation tasks. Unlike traditional canes, the Smart Stick incorporates advanced features that make it highly efficient for the visually impaired. It utilizes a camera-based object detection system and an ultrasonic sensor to accurately identify objects and measure distances. Additionally, the Smart Stick can provide detailed information about identified objects, such as their names. Furthermore, our project takes advantage of integration with a mobile application, reducing the need for extra hardware and minimizing costs. The mobile app offers a range of supportive features for visually impaired individuals and their relatives. With the tracking system, relatives can easily locate the Smart Stick holder, enhancing safety and peace of mind. The navigation system enables users to reach desired destinations effortlessly by issuing voice commands, while the integrated calling system facilitates seamless communication with desired contacts. By combining cutting-edge technology with user-friendly features, the Smart Stick provides an invaluable solution for individuals with visual impairments. It empowers them to navigate their surroundings independently, reducing their reliance on external assistance. This innovative device significantly enhances the overall quality of life for visually impaired individuals, promoting greater autonomy and convenience.

# Table of Contents

<b>Acknowledgment.....</b>	<b>2</b>
<b>Keywords .....</b>	<b>2</b>
<b>Abstract.....</b>	<b>3</b>
<b>Table of Contents .....</b>	<b>4</b>
<b>List of Tables .....</b>	<b>6</b>
<b>List of Figures.....</b>	<b>7</b>
<b>List of Abbreviations .....</b>	<b>9</b>
<b>Chapter 1: Introduction .....</b>	<b>11</b>
1.1 Introduction .....	11
1.2 Problem statement .....	13
1.3 Literature Survey .....	14
1.4 Objectives .....	17
<b>Chapter 2: Project Specifications .....</b>	<b>18</b>
2.1 System Features.....	18
2.2 Block diagram: Hardware .....	18
2.3 Block diagram: Software.....	19
2.4 System Diagrams.....	19
2.4.1 Use-case Diagram.....	19
2.4.2 Sequence Diagram .....	20
<b>Chapter 3: Software.....</b>	<b>23</b>
3.1 Background.....	23
3.1.1 Linux.....	26
Raspbian .....	27
3.1.2 Network .....	28

3.1.3	Node.js Server.....	32
	Socket.io .....	33
3.1.4	Object Detection .....	34
	YOLOv8 .....	42
3.1.5	Speech Recognition .....	45
3.2	Stick Implementation .....	47
3.2.1	Structure.....	47
3.2.2	Implementation .....	47
	Raspberry Pi 4.....	48
	Network .....	49
	Node.js server .....	52
	Socket.io server.....	53
	Socket.io client .....	54
	Buffers .....	55
	Configurations .....	55
	Sensor drivers .....	56
	Object detection .....	57
3.2.3	System logic.....	58
3.3	Mobile Application.....	60
3.3.1	Structure.....	60
3.3.2	Authentication System.....	60
3.3.3	Relative System .....	61
3.3.4	Holder System .....	62
<b>Chapter 4: Testing and Results .....</b>		<b>64</b>
4.1	Stick.....	64
4.1.1	Object Detection .....	64
4.1.2	Object Distance.....	67
4.1.3	Node.js Server.....	68
4.1.4	Socket.io responses (server/client) .....	68
4.2	Mobile Application.....	70
<b>Chapter 5: Hardware .....</b>		<b>76</b>

5.1	Components.....	76
<b>Chapter 6:</b>	<b>Conclusion.....</b>	<b>80</b>
6.1	Conclusion.....	80
6.2	Future Work.....	81
6.3	References .....	82

## List of Tables

Table 1:	Abbreviations table .....	10
Table 2:	Comparison between YOLOv8 models.....	44
Table 3:	Average detection time for Image 2 .....	67

# List of Figures

Figure 1: Number of people with blindness in 2020.....	12
Figure 2: Block diagram of one of literature survey .....	14
Figure 3: Using track from literature survey.....	16
Figure 4: Block diagram of one of literature survey .....	16
Figure 5: System features.....	18
Figure 6: Block diagram hardware .....	18
Figure 7: Block diagram software.....	19
Figure 8: Use-case diagram .....	19
Figure 9: Object detection sequence diagram .....	20
Figure 10: Signup Sequence Diagram .....	20
Figure 11: Login Sequence Diagram .....	21
Figure 12: Relative System Sequence Diagram.....	21
Figure 13: Tracking System Sequence Diagram .....	22
Figure 14: Object detection stages .....	34
Figure 15: One stage object detection.....	35
Figure 16: Two stages object detection .....	35
Figure 17: Anchor-based vs Anchor-free .....	37
Figure 18: YOLO object detection models timelines .....	40
Figure 19: Comparison between yolo versions.....	43
Figure 20: Comparison between YOLO versions.....	44
Figure 21: Speech recognition explanation.....	45
Figure 22: How to convert speech to text. ....	46

Figure 23: Stick system structure.....	47
Figure 24: Mobile System Structure .....	60
Figure 25: Image 1 object detection.....	65
Figure 26: Result image 1 object detection.....	65
Figure 27: Image 2 object detection.....	66
Figure 28: Result image 2 object detection.....	66
Figure 29: Test object distance .....	67
Figure 30: Launching Node.js server.....	68
Figure 31: Socket.io responses .....	68
Figure 32: Server response: mobile requesting to change the mode to explore .....	69
Figure 33: Raspberry pi responses to mode changing operation from the mobile .....	69
Figure 34: Raspberry pi responses to mode changing to detect objects from the mobile .....	69
Figure 35: Error message displaying .....	70
Figure 36: Text to speech.....	71
Figure 37: Voice command guiding .....	72
Figure 38: According to voice command user initiate a phone call .....	73
Figure 39: Relative send request to holder .....	74
Figure 40: Relative can track his holder in real-time.....	75
Figure 41: RPI Camera v2 .....	77
Figure 42: DC Vibration Motor .....	77
Figure 43: Ultrasonic sensor .....	78
Figure 44: Raspberry Pi4 .....	79



## List of Abbreviations

HCI	Human–Computer Interaction
VLEG	Vision Loss Expert Group
API	Application Programming Interface
OCR	Optical Character Recognition
YOLO	You Only Look Once
GPS	Graphics Positioning System
APP	Application
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
IDE	Integrated Development Environment
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
HTTP	(Hypertext Transfer Protocol
Mbps	Megabits Per Second
Gbps	Gigabits Per Second
Wi-Fi	Wireless Fidelity
OS	Operating System
SSD	Single Shot Detector
R-CNN	Regional Convolutional Neural Network
IOU	Intersection Over Union
SOTA	State-Of-The-Art

mAP	Mean Average Precision
AP	Average Precision
PASCAL VOC	Pascal Visual Object Classes
GNU	General Public License
ASR	Automatic Speech Recognition
STT	Speech-to-Text
AI	Artificial Intelligence
iOS	iPhone Operating System
HMMs	Hidden Markov Models
DNNs	Deep Neural Networks
TTS	Text-to-Speech
NLG	Natural Language Generation
GPT	Generative Pretrained Transformer
gTTS	Google Text-to-Speech
PDF	Portable Document Format
SDK	Software Development Kit
RPI	Raspberry Pi
USB	Universal Serial Bus

*Table 1: Abbreviations table*

# **Chapter 1: Introduction**

## **1.1 Introduction**

Visually impaired persons have difficulty interacting and feel their environment. They have little contact with their surroundings, and a difficult time walking down a busy street. Blind people always needed someone to guide them. They must be dependent on someone. People are very busy with their own life, and they are hardly bothered by anyone else. This makes the life of such blind people very difficult. Physical movement is a challenge for visually impaired people because it can be tricky to distinguish obstacles appearing in front of them which make them unable to move from one place to another. These issues have driven the researchers toward exploring new avenues of research across several disciplines such as assistive technologies, cognitive psychology, computer vision, sensory processing, rehabilitation, and accessibility-inclusive human–computer interaction (HCI). Statistics before 2020 stated that 285 million people are visually impaired, of which 39 million are blind, and 246 million have low vision. About 90% of the blind population is living in developing countries, out of which 82% are aged over 50. The expected population of blind people would be 75 million by 2020. This increase in the population of blind people will affect the quality of life and cause social imbalance in the future, if not addressed adequately. [1]

A new study released in *The Lancet Global Health*, a leading medical journal, estimates that global blindness will triple by the year 2050. Produced by the Vision Loss Expert Group (VLEG), it projects that: cases of blindness will increase from 36 million to 115 million by 2050. Other types of eye problems will impact another 200 million people. [2] Researchers have spent decades developing an intelligent smart stick to assist and alert visually impaired persons to obstacles and give

information about their location. Over the last decades, research has been conducted for new devices to design a good and reliable system for visually impaired persons to detect obstacles and warn them at danger situations. So, the proposed system depicts the idea of smart stick for the blinds which will help them find the way on their own without the help of anyone. Smart walking stick is specially designed to detect obstacles which may help the blind to navigate care-free. The audio messages will keep the user alert and considerably reduce accidents. A voice enabled automatic switching is also incorporated to help them in private space as well. This system presents a concept to provide a smart electronic aid for blind people, both in public and private space. The proposed system contains different sensors to keep track of different aspects. These components relate to several software systems to operate and make the life of the holder much easier. In 2020, around 55 percent of people living with blindness were women, this amounts to almost 24 million women. From the 19 million men living with blindness, 75 percent were 50 years and over. This statistic shows the number of people with blindness worldwide in 2020, by age group and gender. [3]

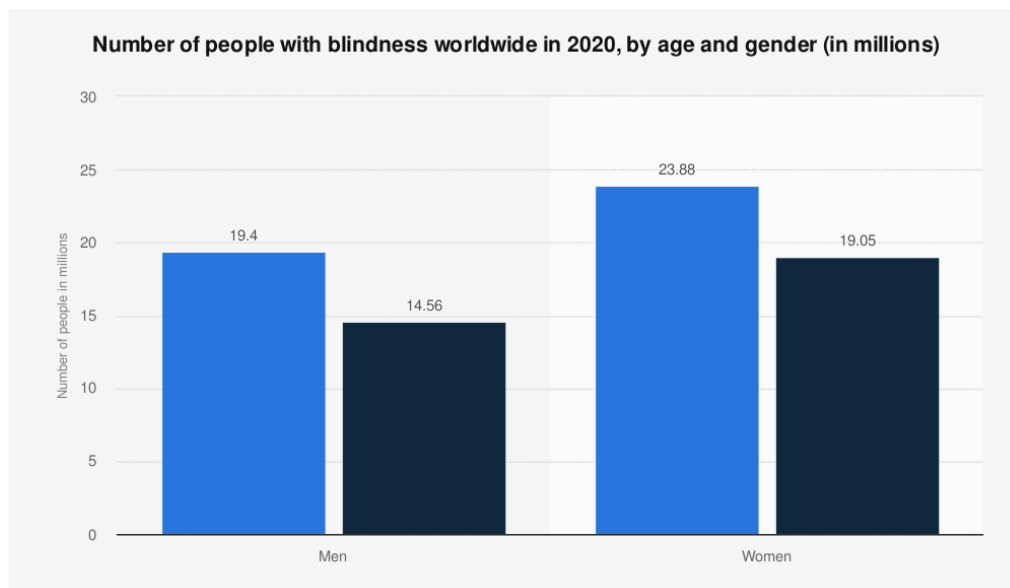


Figure 1: Number of people with blindness in 2020

## 1.2 Problem statement

One of the major problems Blind people have a hard time navigating outside their homes because they can't see what's in their way. They can use a cane or guide dog, but these methods are not always reliable. Canes can't detect small obstacles, and guide dogs can be distracted or led astray. [4]

Societal stigma: Blind people sometimes have accidents because the world is not designed for them. For example, they might trip over a chair that is not put away or knock over a glass that is left on the edge of a table. Sighted people might think that these accidents mean that blind people are not capable, but they just happen because the world is not accessible to blind people. [5]

Often living in isolation: it's not a surprise that living with a visual impairment might signify, often, living in isolation. Dealing with sight loss, already, is a challenge. The lack of emotional support at diagnosis centers, the limited accessibility to activities and information, the societal stigma, and the lack of unemployment, are all factors frequently leading blind or low vision individuals in isolation. [5]

The proposed system is designed to help disabled people gain independence, confidence, and flexibility. The system will do this by detecting obstacles in front of the user and identifying them, so the user can identify things themselves. The system will also allow the user to navigate anywhere they want to be using up-to-date Maps API. As the user gets used to things and memorizes them, the system will be handed to them in the regular form of their cane, so they can easily use the system and gain flexibility in dealing with it. The proposed system has the potential to make a significant difference in the lives of disabled people. It can help them to live more independently, confidently, and flexibly.

## 1.3 Literature Survey

### 1.3.1 Raspberry pi based smart walking stick for visually impaired person [4]

It offers a camera-based assisted reading system to assist blind people in reading text labels and product packaging from everyday things. The text that the user needs to read is collected as an image and sent to the image processing platform using a small camera. OCR tesseract is used to recognize the text on the acquired image. The e- speak algorithm converts the detected text into vocal output.

#### 1- Advantage

1. Receiving communications, we can keep track of the blind people's whereabouts.
2. Character recognition that is done automatically.
3. A Guide to Navigation for the Blind
4. A motion-based method for detecting the object for a few seconds to detect.

#### 2- Disadvantage

The smart stick doesn't support an Emergency System.

#### 3- Block Diagram

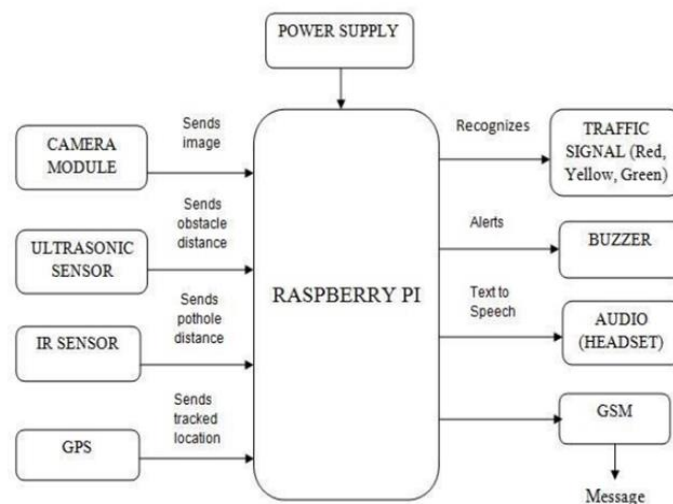


Figure 2: Block diagram of one of literature survey

### **1.3.2 Smart Stick for Blind People. [5]**

They propose to design an intelligent device which vibrates on occurrence of obstacles based on distance between the person and the obstacle. It will not only vibrate but also will tell what the obstacle is, and it will also tell the direction that the person should go. It gives them confidence and allows them to integrate into society as equals. It has a small earphone connected to the stick to help the person to hear the voice. It has a sensitive sensor to give an alarm when it finds fire.

#### **4- Feature**

1. Smart stick design to detect obstacles that may hinder their movement such as bumps in the street, stones.
2. It can detect water and fire with water and fire sensors.
3. It will enable them to know the things in front of them.

#### **5- Methodology**

1. Digital Image Processing.
2. Artificial Neural Network.
3. Automatic Speech Recognition.
4. Object detection using Yolo Algorithm.

#### **6- Future work**

1. Provide Arabic language support for both voice recognition.
2. Improve the performance and reduce response time of the device.
3. Add smart home commands by which user can control lights, air conditioner and other home appliances; so, the home will be more controllable and cozier.
4. Add a GPS feature to the device to guide the user outside the home.

### 1.3.3 Design and Implementation of a Walking Stick Aid for Visually Challenged People. [6]

This system was designed comprising a walking stick and an APP. As the walking stick is embedded with two ultrasonic sensors, it can detect obstacles at the ground level, as well as for upper body parts. The created APP is useful for parents tracking user location, and the emergency APP is very useful for the user in panic situations.

#### 1- Advantage

1. System can detect obstacles at the ground level.
2. It also can detect water.
3. APP is useful for parents tracking user location.
4. The emergency APP is very useful for the user in panic situations.

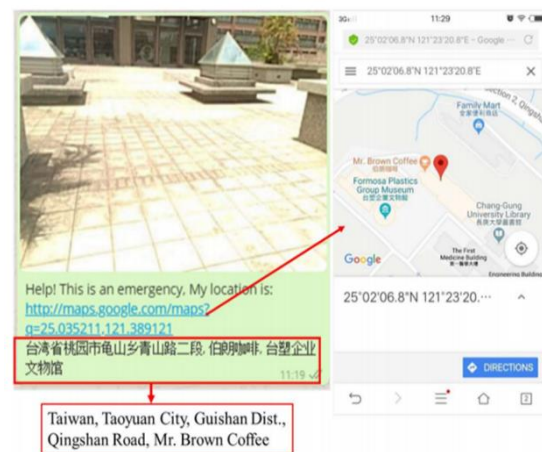


Figure 3: Using track from literature survey.

#### 2- Block Diagram

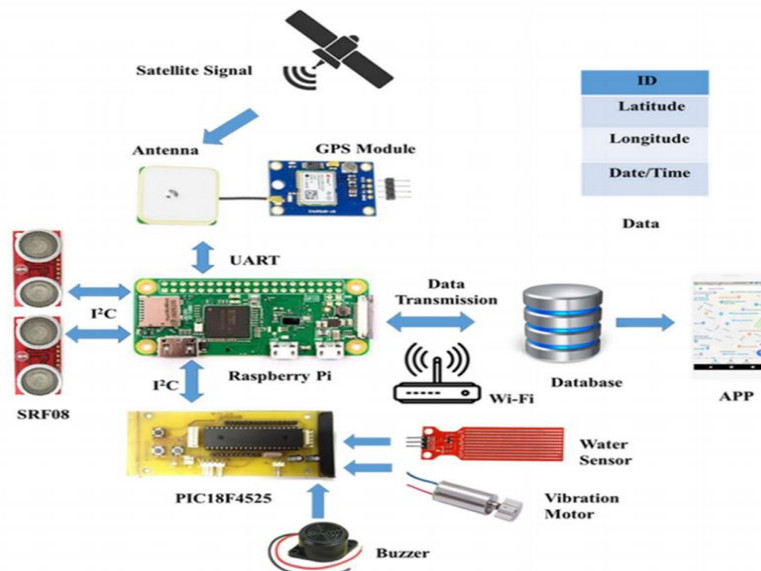


Figure 4: Block diagram of one of literature survey



## 1.4 Objectives

The main objective of our project is to empower blind individuals to navigate independently and comfortably. Here, we have developed an intelligent system that assists blind people in moving through their surroundings with confidence. Our project focuses on enhancing the holder's independence, self-assurance, and adaptability.

The system aims to provide blind individuals with a tool that boosts their confidence in moving through various environments, both indoors and outdoors. By incorporating advanced technologies such as computer vision, sensory feedback, and voice recognition, we strive to create a system that enables blind people to confidently explore different spaces. The objective is to design a user-friendly interface that is easy to understand and operate, ensuring that blind individuals can effortlessly interact with the system. Through this interface, they will gain the necessary information and guidance to make informed decisions and overcome obstacles in their path.

Additionally, the system aims to promote independence among blind individuals by reducing their reliance on others for assistance. By providing real-time feedback and alerts about potential hazards or obstacles, the system allows users to proactively navigate their surroundings and make necessary adjustments. Ultimately, our goal is to instill a sense of freedom and self-assurance in blind individuals, enabling them to explore their surroundings with greater autonomy and confidence. We believe that by equipping them with the necessary tools and support, we can empower blind individuals to lead fulfilling lives and actively participate in society.

## Chapter 2: Project Specifications

### 2.1 System Features

The following structure illustrates the overall functionalities of the project:

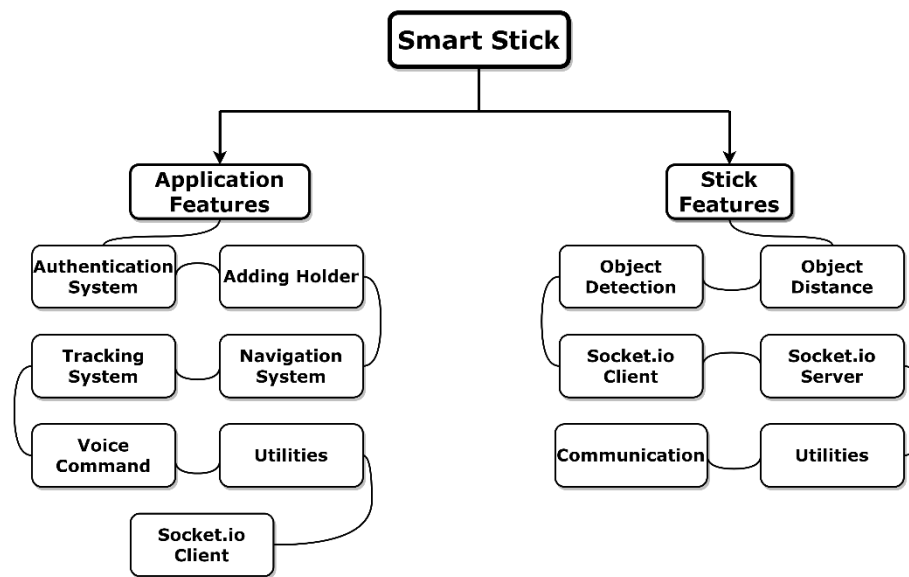


Figure 5: System features

### 2.2 Block diagram: Hardware

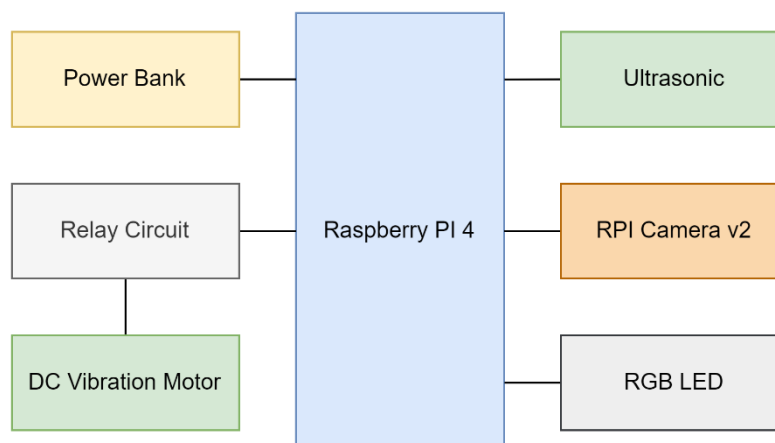


Figure 6: Block diagram hardware

## 2.3 Block diagram: Software

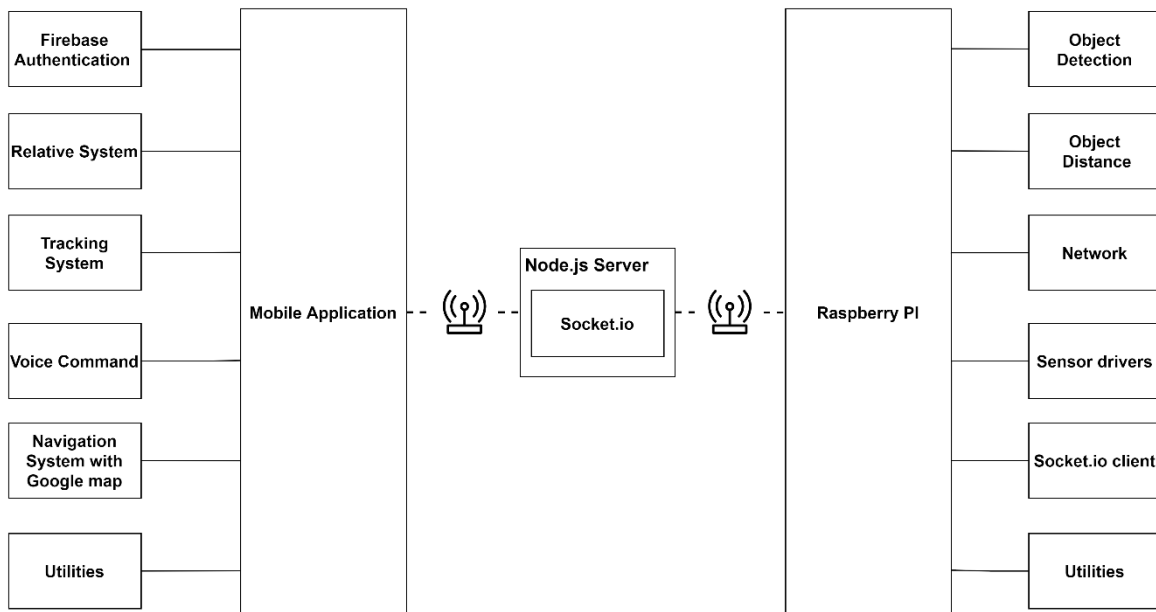


Figure 7: Block diagram software

## 2.4 System Diagrams

### 2.4.1 Use-case Diagram

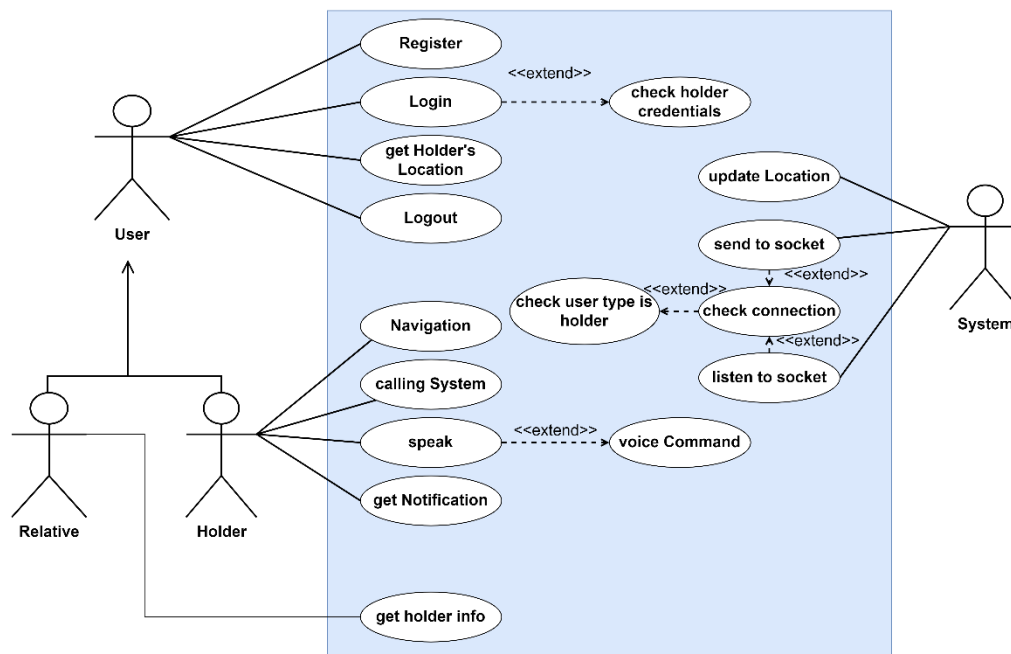


Figure 8: Use-case diagram

## 2.4.2 Sequence Diagram

### 1) Object Detection System

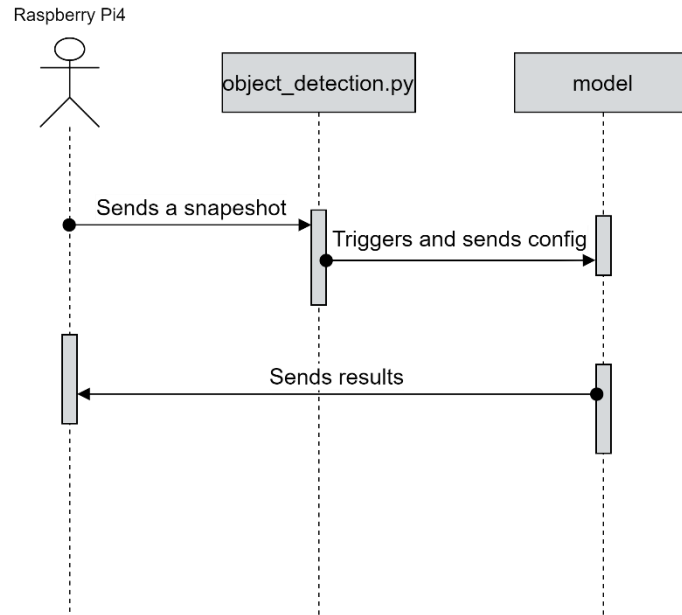


Figure 9: Object detection sequence diagram

### 2) Signup Sequence Diagram

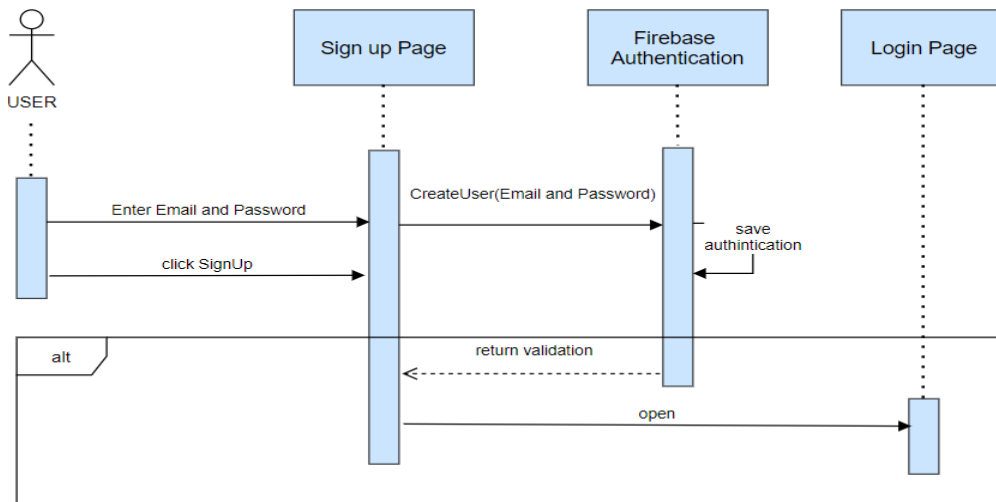


Figure 10: Signup Sequence Diagram

### 3) Login Sequence Diagram

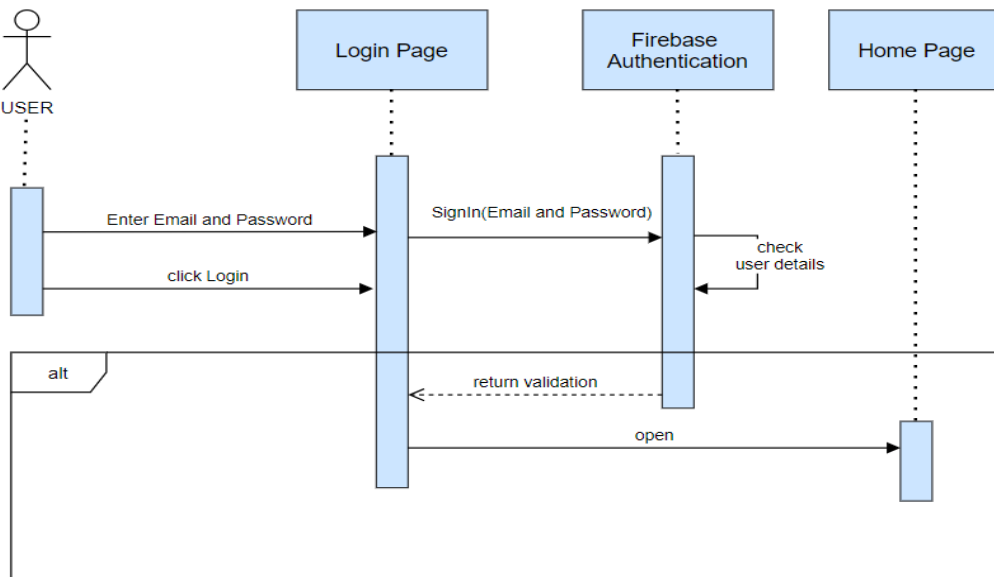


Figure 11: Login Sequence Diagram

### 4) Relative System Sequence Diagram

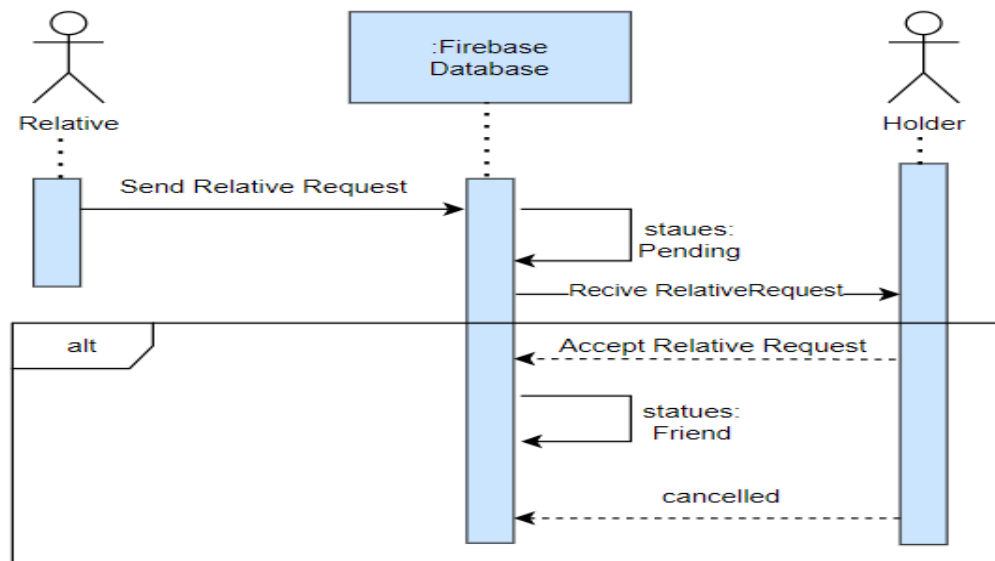


Figure 12: Relative System Sequence Diagram

## 5) Tracking System Sequence Diagram

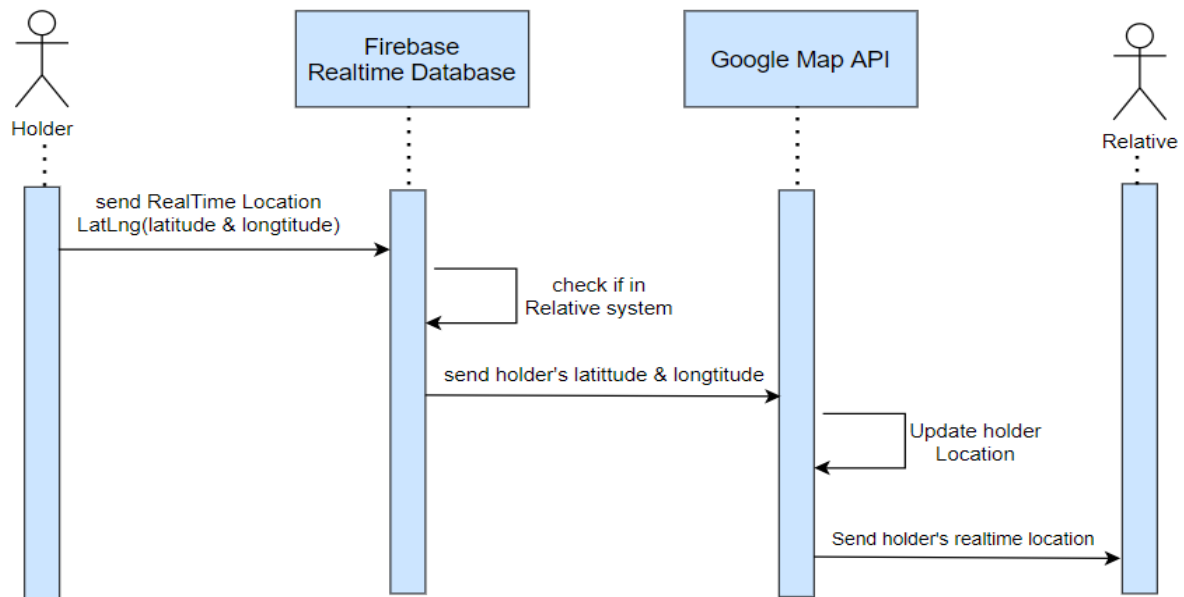


Figure 13: Tracking System Sequence Diagram

## **Chapter 3: Software**

### **3.1 Background**

In our project, we have developed a comprehensive solution that caters to the needs of visually impaired individuals. By combining various software components and cutting-edge technologies to enhance the functionality and accessibility of our system. Through the incorporation of advanced computer vision techniques and speech recognition capabilities, we aim to empower visually impaired individuals with greater independence and mobility. To ensure the reliability and efficiency of our platform, we have chosen Linux as the primary operating system, with a particular emphasis on Raspbian, which has been optimized specifically for Raspberry Pi devices. This strategic decision enables us to leverage the flexibility and capabilities of the Linux ecosystem, providing a solid foundation for our project. Additionally, we have integrated technologies and libraries such as OpenAI to further enhance our system's capabilities and performance.

Networking plays a pivotal role in our project, as it involves establishing seamless connections between devices to facilitate communication and data transfer. To design an efficient and reliable network infrastructure, we must have a comprehensive understanding of various networking concepts. Our exploration of topics such as network infrastructure, IP addressing, static IP addressing, DHCP, subnetting, DNS, network protocols, bandwidth and network speed, Wi-Fi and wireless networking, network troubleshooting and diagnostics, network scalability, network monitoring and management and firewall and security, enables us to optimize the network architecture of our smart stick system. By emphasizing the relevance of these networking concepts to our project, we underscore their contributions to the reliability, performance, and security of our system. We employ

specific technologies, such as Wi-Fi standards and network monitoring tools, to deepen our understanding and implement best practices. Additionally, we have implemented a Node.js server to facilitate real-time communication between the smart stick and other devices using the Socket.io library. This allows for seamless and instant updates, ensuring timely information exchange between components of our system. Our comprehensive approach empowers us to build a robust and effective smart stick system, ultimately enhancing the independence and mobility of visually impaired individuals.

Within our software solution, object detection utilizing the YOLOv8 algorithm plays a crucial role. This state-of-the-art technology enables the smart stick to accurately identify and recognize obstacles in the user's surroundings, allowing for confident and safe navigation. Furthermore, speech recognition capabilities enable users to interact with the smart stick through intuitive voice commands, enhancing the user experience and ease of use.

Recognizing the potential of mobile devices, we developed an Android application to complement the smart stick and expand its features. This approach allowed us to optimize functionality while reducing costs by leveraging the capabilities of smartphones. In the realm of mobile app development, we focused on native development for Android, utilizing the official Android Studio IDE and the Kotlin programming language. Kotlin's modern and expressive syntax, along with its interoperability with Java, allowed us to create high-performance and intuitive mobile applications that provided complete control over the development process. To enhance the functionality of our mobile application, we integrated Firebase, a versatile platform provided by Google. Firebase offered a range of services and tools that simplified the development process and enabled seamless integration of features such as authentication and real-time database functionality. Firebase Authentication



allowed us to incorporate secure user authentication easily, while Firebase Realtime Database provided real-time synchronization and offline capabilities, essential for tracking systems requiring location updates at regular intervals. Retrieving location data from Firebase involved subscribing to real-time events or querying for the latest coordinates periodically. With the obtained latitude and longitude data, we were able to update the map in the mobile application accordingly. By adding markers and animating their movement on the Google Map, we provided visually impaired individuals with a smooth and visually engaging experience, allowing them to navigate their surroundings with ease.

Overall, our project aimed to empower visually impaired individuals by developing a comprehensive system that addressed their unique challenges. Through the integration of software components and technologies, we aimed to enhance independence, foster inclusivity, and significantly improve the overall quality of life for visually impaired individuals.

### 3.1.1 Linux

Linux is one of the most important pieces of software and one of the core elements of computer operating systems. It is widely utilized in the operating systems of modern supercomputers, smart gadgets, and internet infrastructure equipment. With its versatility and compatibility, Linux can run a wide range of programs, ranging from basic calculators to sophisticated graphics editing tools. It finds applications in diverse contexts and supports various use cases, making it a widely adopted choice. [7]

Among the different Linux distributions available, Debian, Fedora, and Ubuntu are particularly well-known. Debian stands out for its portability and serves as a foundation for many other Linux distributions across different platforms. Raspbian, the most popular operating system for Raspberry Pi, is based on Debian. [8]

Linux Operating System has primarily three components

- **Kernel** – Kernel is the core part of Linux. It is responsible for all major activities of this operating system. It consists of various modules, and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.
- **System Library** – System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries implement most of the functionalities of the operating system and do not require kernel module's code access rights.
- **System Utility** – System Utility programs are responsible to do specialized, individual level tasks. [9]

## **Raspbian**

Raspbian is an operating system specifically designed for the Raspberry Pi hardware. It is based on Debian, a popular Linux distribution, and is optimized to provide the best performance and compatibility on Raspberry Pi devices. Raspbian goes beyond being a simple operating system by offering over 35,000 pre-compiled software packages in an easy-to-install format. The development of Raspbian started in 2012, with an initial build of over 35,000 packages tailored for the Raspberry Pi. Since then, it has been continuously developed and improved to enhance the stability and performance of the Debian packages that it includes. Raspbian is a community-funded and supported project, reflecting the collaborative efforts of developers and Raspberry Pi enthusiasts.

One of the key advantages of Raspbian is its lightweight nature, optimized to run smoothly on the Raspberry Pi's hardware while using minimal system resources. It is also open-source software, which means that it is freely available for anyone to use, modify, and distribute without restrictions. This open-source nature fosters innovation and collaboration within the Raspberry Pi community.

Raspbian features a user-friendly interface, making it accessible to beginners who are new to the Raspberry Pi platform. It comes with a set of pre-installed applications, including a web browser, a text editor, and a file manager, providing essential tools for various tasks. Additionally, Raspbian benefits from a large online community of Raspberry Pi enthusiasts who contribute tutorials, guides, and support, making it easier for users to get started and find assistance when needed. Overall, Raspbian offers a lightweight, optimized, and user-friendly operating system for the Raspberry Pi. Its extensive package repository, community support, and compatibility with the Raspberry Pi hardware. [10]

### **3.1.2 Network**

A network serves as a vital system that connects multiple computing devices, facilitating the transmission and sharing of information. This encompasses various devices like mobile phones, computers, and servers, which are interconnected using physical wires like fiber optics or wirelessly. [11]

#### **IP Addressing:**

IP (Internet Protocol) addressing is an essential system utilized to assign unique identifiers to devices on a network. An IP address consists of a network portion and a host portion. The prevalent IPv4 (Internet Protocol version 4) employs a four-part numerical representation (e.g., 192.168.0.1). IPv6 (Internet Protocol version 6), designed to overcome IPv4 limitations, offers a larger address space. In our project, we utilize IPv4, and comprehending IP addressing is crucial for configuring and facilitating communication within the network.

#### **Static IP Address:**

A static IP address is a fixed, permanent address assigned to a device on a network. Unlike dynamic IP addresses that periodically change, a static IP address remains constant, ensuring consistent and reliable communication. Implementing static IP addresses in our project guarantees predictable and stable network connections, enabling seamless communication and data transfer.

#### **DHCP (Dynamic Host Configuration Protocol):**

DHCP is a network protocol that automates the assignment of IP addresses and other network configuration parameters to devices on a network. By eliminating manual

IP address assignment, DHCP simplifies network management. In our project, DHCP can dynamically assign IP addresses to network-connected devices, facilitating the addition of new devices or reconfiguration of existing ones. [12]

### **Subnetting:**

Subnetting involves dividing a network into smaller subnetworks (subnets) to enhance performance and efficiently manage IP addresses. By implementing subnetting, we establish logical divisions within the network, aiding organization, and control. In our project, subnetting is instrumental in optimizing network traffic and improving the performance of the smart stick system.

### **DNS (Domain Name System):**

DNS is a distributed naming system that translates domain names (e.g., [www.example.com](http://www.example.com)) into IP addresses. It enables users to access web services using user-friendly domain names rather than complex IP addresses. DNS plays a vital role in resolving domain names, facilitating communication and accessibility over the internet. Incorporating DNS into our project enhances the user experience by enabling seamless and intuitive access to web-based services.

### **Network Protocols:**

Network protocols establish rules and conventions for communication between devices on a network. Commonly used protocols include TCP/IP (Transmission Control Protocol/Internet Protocol), UDP (User Datagram Protocol), HTTP (Hypertext Transfer Protocol), and DNS (Domain Name System). Understanding and utilizing appropriate network protocols in our project ensures efficient and reliable communication between the smart stick, mobile app, and other components.

### **Bandwidth and Network Speed:**

Bandwidth refers to a network's capacity to transmit data and determines the amount of data that can be transmitted over a network connection in each period. Network speed, typically measured in data transfer rates (e.g., Mbps or Gbps), indicates how quickly data can be transmitted between devices. Considering the required bandwidth and network speed ensures the smart stick system operates smoothly and without latency.

### **Wi-Fi and Wireless Networking:**

Wi-Fi technology enables wireless communication between devices, allowing them to connect to a local network or the internet without physical cables. Incorporating Wi-Fi capabilities into our project provides flexibility and mobility. It enables wireless communication between the smart stick and mobile app, allowing for seamless interaction and data exchange. By leveraging Wi-Fi technology, visually impaired individuals can access online services, receive real-time updates, and benefit from the power of the internet. Furthermore, ensuring a secure and properly configured Wi-Fi network is crucial in maintaining the privacy and integrity of user data, ensuring a safe and secure experience for users.

### **Network Troubleshooting and Diagnostics:**

Network troubleshooting and diagnostics involve identifying and resolving issues that may arise within a network. This encompasses diagnosing connectivity problems, troubleshooting hardware or software configurations, and analyzing network traffic. Acquainting ourselves with network troubleshooting techniques and tools equips us to effectively address any network-related issues that may occur during the development or deployment of our project.

**Network Scalability:**

Considering the scalability of our network infrastructure is crucial for accommodating potential future growth or expansion. As our project evolves, supporting a larger number of devices or users may become necessary. Planning for scalability involves designing the network infrastructure to easily accommodate new devices, increased bandwidth, and efficient management of network resources.

**Network Monitoring and Management:**

Network monitoring and management involve monitoring network resources, performance, and availability. Employing network monitoring tools and techniques enables administrators to track network traffic, diagnose performance issues, and ensure optimal network operation. Implementing network monitoring capabilities in our project detects and addresses network problems, ensuring a reliable and efficient smart stick system.

**Firewall and Security:**

Firewalls are network security devices that monitor and control incoming and outgoing network traffic, acting as a barrier between internal and external networks. They safeguard the network from unauthorized access and potential threats. Implementing a firewall in our project's network infrastructure enhances the security of the smart stick system, preserving the privacy and integrity of user data.

### 3.1.3 Node.js Server

Node.js is an open-source JavaScript runtime environment that allows developers to run JavaScript code on the server-side. It was initially developed by Ryan Dahl in 2009 and has since gained significant popularity in the web development community. Node.js's versatility extends beyond web development. It can also be used for building command-line tools, desktop applications, and even IoT (Internet of Things) devices, thanks to its lightweight and efficient nature. [13]

One of the key features of Node.js is its event-driven, non-blocking I/O model. Unlike traditional server-side technologies, which rely on a thread-based model, Node.js uses an event loop that allows it to handle concurrent connections efficiently. This event-driven architecture makes Node.js lightweight and well-suited for building scalable network applications. Node.js provides a rich set of built-in modules and libraries that simplify common tasks in web development, such as handling HTTP requests, working with file systems, and implementing real-time communication. It also has a vast ecosystem of third-party packages available through the npm (Node Package Manager) registry, which allows developers to easily integrate additional functionality into their applications.

One of the key advantages of Node.js is its ability to share code between the client and server. With Node.js, developers can write JavaScript code for both the front-end and back-end, eliminating the need to switch between different programming languages. This unified language approach improves developer productivity and allows for better code reusability. Node.js has become a popular choice for building web applications, especially those requiring real-time communication or handling many concurrent connections. It is commonly used in areas such as real-time chat applications, collaborative tools, streaming services, and APIs. [14]



## **Socket.io**

Socket.IO is a powerful and widely used library that extends the capabilities of Node.js, allowing for real-time, bidirectional communication between clients and servers. It simplifies the process of establishing WebSocket connections, enabling seamless data transfer and event-driven communication. [15]

By leveraging Socket.IO, developers can easily integrate real-time features into their Node.js applications. This includes functionalities such as chat systems, live updates, collaborative editing, multiplayer games, and more. Socket.IO abstracts away the complexities of working with WebSockets, providing a straightforward and unified API for handling events, sending, and receiving messages, and managing connections. The underlying technology behind Socket.IO is WebSockets, a communication protocol that enables full-duplex, real-time communication between clients and servers over a single, long-lived connection. Socket.IO takes care of establishing and managing WebSocket connections, ensuring reliable and efficient data transfer between the client and server.

One of the key advantages of Socket.IO is its ability to leverage the event-driven, non-blocking nature of Node.js. This means that it can efficiently handle multiple concurrent connections without blocking the execution of other tasks. This scalability makes Socket.IO suitable for applications with high levels of concurrent usage, ensuring optimal performance and responsiveness. Socket.IO provides a range of features and options to customize real-time communication based on the specific requirements of the application. It supports various transport mechanisms, including WebSockets, AJAX long polling, and more, ensuring compatibility across different browsers and network conditions. Overall, Socket.IO simplicity, scalability, and compatibility make it a popular choice for building real-time features in web applications. [16]

### 3.1.4 Object Detection

Object detection is one of the fundamental problems of computer vision. It forms the basis of many other downstream computer vision tasks, for example, instance segmentation, image captioning, object tracking, and more. In general, deep learning-based object detectors extract features from the input image or video frame. An object detector solves two subsequent tasks:

- Task 1: Find an arbitrary number of objects (possibly even zero).
- Task 2: Classify every object and estimate its size with a bounding box.

To simplify the process, you can separate those tasks into two stages. Other methods combine both tasks into one step (single-stage detectors) to achieve higher performance at the cost of accuracy. Object detection methods can be categorized into two main types: One-stage and two-stage object detectors. [17]

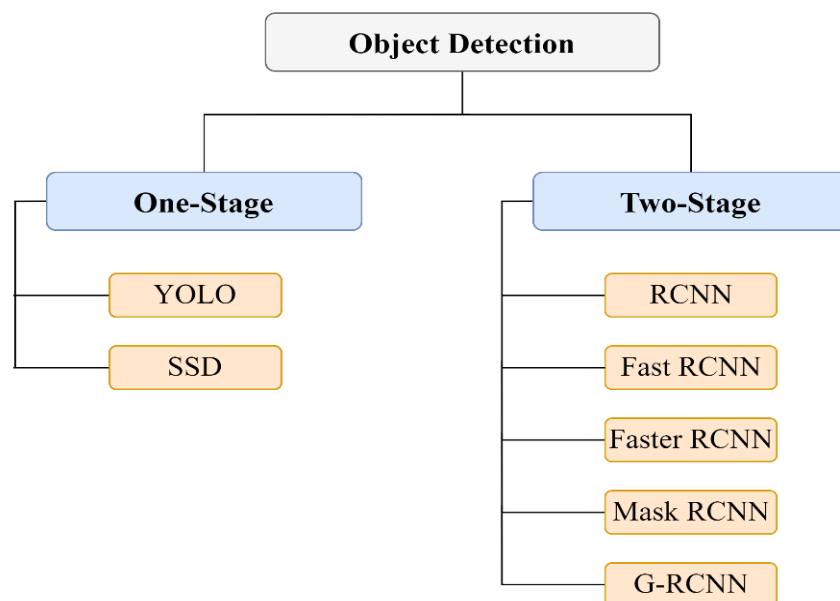


Figure 14: Object detection stages

One-stage detectors predict bounding boxes over the images without the region proposal step. This process consumes less time and can therefore be used in real-time applications. [18]

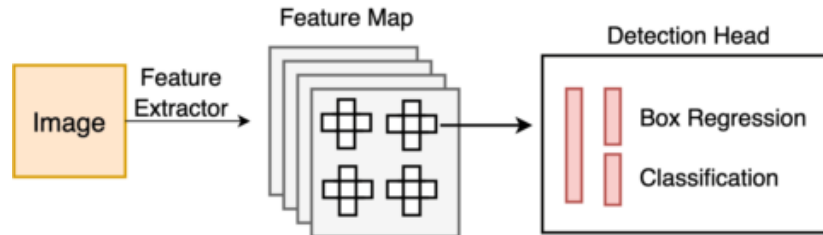


Figure 15: One stage object detection

- One-stage object detectors prioritize inference speed and are super-fast but not as good at recognizing irregularly shaped objects.
- The main advantage of single stage is that those algorithms are generally faster than multi-stage detectors and structurally simpler.

In two-stage object detectors, the approximate object regions are proposed using deep features before these features are used for the classification as well as bounding box regression for the object candidate. [18]

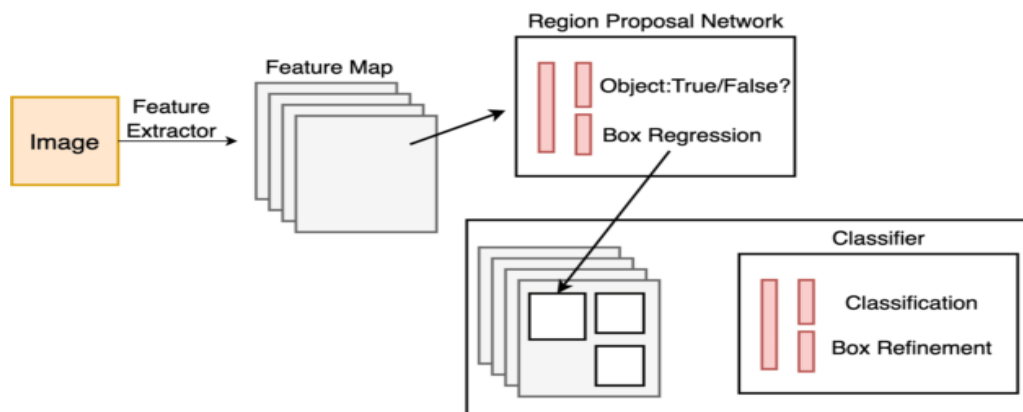


Figure 16: Two stages object detection

- Typically, methods achieve the highest detection accuracy but are typically slower. Because of the many inference steps per image, the performance (frames per second) is not as good as one-stage detectors.
- Two-stage object detectors first find a region of interest and use this cropped region for classification.

## **Anchor boxes and anchor-free detection**

Anchor boxes and anchor-free detection are two different approaches used in object detection algorithms.

- **Anchor boxes**

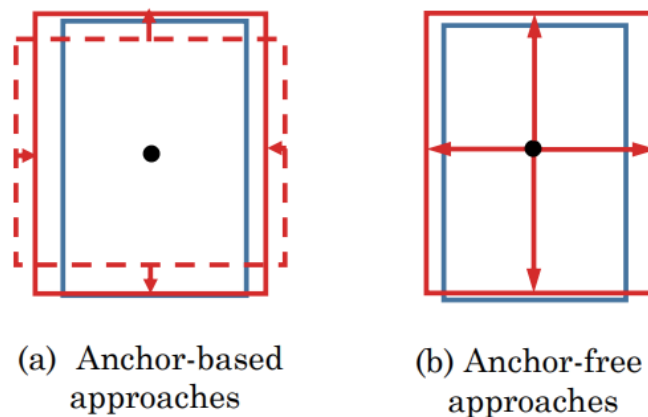
Anchor boxes, also known as prior boxes, are a crucial component of many traditional object detection methods. These methods divide an input image into a grid of cells and assign each cell multiple anchor boxes of different sizes and aspect ratios. The anchor boxes act as reference bounding boxes that are placed at predefined positions and scales across the image. During training, anchor boxes are matched with ground-truth objects based on their intersection over union (IoU). Positive matches are assigned to anchor boxes with high IoU with ground-truth objects, while negative matches are assigned to anchor boxes with low IoU. The network is then trained to predict the offset and classification scores for each anchor box. The use of anchor boxes helps to handle scale and aspect ratio variations of objects. However, it requires careful selection of anchor box sizes and aspect ratios, which can be challenging.

- **Anchor -free detection**

Anchor-free detection eliminates the need for predefined anchor boxes. Instead, they directly predict the object's center point and bounding box without explicitly using anchor boxes. In anchor-free methods, each spatial location in the feature map is responsible for predicting the position and size of an object. The network outputs the center coordinates of the object and offsets to predict the bounding box. This approach simplifies the detection process and reduces the design complexity by removing the dependency on anchor box priors. Anchor-free methods often achieve

competitive or even better performance compared to anchor-based methods while being computationally efficient. They can handle objects of various scales and aspect ratios more effectively since there are no anchor boxes restricting the predictions.

In summary, the main difference between anchor boxes and anchor-free detection lies in how the object detection algorithm handles bounding box prediction and matching. Anchor boxes use predefined boxes at different scales and aspect ratios, while anchor-free detection directly predicts the center points and bounding boxes without relying on anchor boxes.



*Figure 17: Anchor-based vs Anchor-free*

Different approaches for localization prediction. Blue boxes denote ground truth objects. Red boxes in solid line represent the box prediction. Red arrows are regression predictions. Predictions are based on current center locations (black dots). (a) Anchor based approaches: the dashed box in red denotes one anchor box with possibly high IoU with the target. It first uses classification to select an anchor box that has high IoU with the object, then uses regression to predict a more accurate location. (b) Anchor-free approaches: the four borders of a bounding box are directly regressed. [19]

# YOLO

## 1- Introduction

YOLO is a state-of-the-art (SOTA) real time object detection algorithm in which the problem of object detection has been reframed as a regression problem instead of classification problem by the creators. A convolutional neural network predicts the bounding boxes as well as class probabilities for all the objects depicted in an image. This algorithm identifies the objects and their positioning with the help of bounding boxes by looking at the image only once. Improvements were made to YOLO since the first release in 2015. It is classified as one of the one stage object detectors.

## 2- How it works

The algorithm takes an image and divides it into  $N$  grids which has dimensional region of  $S \times S$ . Each of these  $N$  grids is responsible for the detection and localization of the  $B$  bounding box for the object it contains. Formally, for a grid, it predicts the following parameters for single bounding box:

- The confidence probability ( $P_c$ ) is the probability of containing an object.
- The bounding box center ( $b_x, b_y$ )
- The bounding box dimensions ( $b_h, b_w$ )
- Classes/Categories ( $C_i$ )

These parameters are composed into a  $(5 \times B + C)$  vector that represents the grid where  $B$  is the number of bounding boxes inside the grid, and  $C$  is the number of classes. Correspondingly, the output tensor will be in the shape  $(S \times S (5 \times B + C))$ .

The next step is to compute the confidence score ( $C_s$ ) for each bounding box per grid by multiplying  $P_c$  with Intersection over Union (IoU) between the ground-truth

and predicted-bounding-box. If an object does not exist in the grid cell, the confidence score would be zero. Then class specific score ( $C_{ss}$ ) is computed for each bounding box of all the grid cells. It encodes both the probability of the class appearing in that box and how well the predicted box fits the object. If the cell contains multiple centers, then anchor boxes of different sizes and shapes are applied to separate different centers of objects. Anchor boxes are just a set of several standard bounding boxes, selected after analyzing the dataset and underlying objects in the dataset. Adjacent grid cells may also predict the same object, so there will be a lot of bounding boxes. YOLO uses Non-maximum Suppression algorithm to suppress bounding boxes with  $P_c$  less than pre-defined threshold. After that, the algorithm picks the one with the highest  $P_c$ , then suppresses the remaining that has IoU greater than another pre-defined threshold. [20]

### **3- Evaluation**

Evaluation of object detection algorithms is made using some metrics such mAP and IoU. These metrics are important for selecting an algorithm over one another. Intersection over union (IoU) is a metric that helps us measure the correctness of a prediction by evaluating the overlap of ground truth and predicted boxes. IoU values range from 0 to, where 0 means no overlap and 1 means perfect overlap. It is computed by dividing the area of overlapping of predict box and ground truth by the area of the union. Mean average precision (mAP) is the average of AP over all detected classes. Average Precision is calculated for each class separately as the area under a precision vs recall curve for a set of predictions. The precision vs recall curve is interpolated using PASCAL VOC Interpolation Method. Recall is calculated as the ratio of the total predictions made by the model under a class with a total of existing labels for the class. On the other hand, Precision refers to the ratio of true positives with respect to the total predictions made by the model.

#### 4- YOLO Object Detection Models Timelines [21]

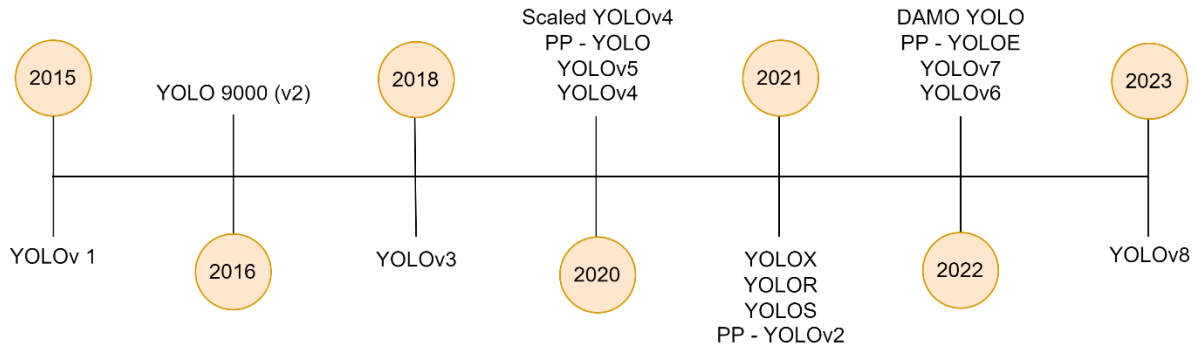


Figure 18: YOLO object detection models timelines

- **YOLOv1**

The very first version of YOLO object detection, that is YOLOv1 was published by Joseph Redmon et al. in 2015. It was the first single stage object detection (SSD) model which gave rise to SSDs and all the subsequent YOLO models.

- **YOLO 9000 (v2)**

YOLOv2, also known as YOLO 9000 was published by the original YOLOv1 author, Joseph Redmon. It improved upon YOLOv1 by introducing the concept of anchor boxes and a better backbone, that is Darknet-19.

- **YOLOv3**

In 2018, Joseph Redmon and Ali Farhadi published YOLOv3. It was less of an architectural leap and more of a technical report, but a big improvement in the YOLO family, nonetheless. YOLOv3 uses the Darknet-53 backbone, residual connections, better pretraining, and image augmentation techniques to bring in improvements.



- **Ultralytics YOLO Object Detection Models**

All the YOLO object detection models till YOLOv3 were written using the C programming language and used the Darknet framework. Newcomers find it difficult to traverse the codebase and fine-tune the models. Around the same time as YOLOv3, Ultralytics released the first ever YOLO (YOLOv3) implemented using the PyTorch framework. It was much more accessible and easier to use for transfer learning as well. Shortly after publishing YOLOv3, Joseph Redmon stepped away from the Computer Vision research community. YOLOv4 (by Alexey et al.) was the last YOLO model to be written in Darknet. After that, there have been many YOLO object detections. Scaled YOLOv4, YOLOX, PP-YOLO, YOLOv6, and YOLOv7 are some of the prominent among them. After YOLOv3, Ultralytics also released YOLOv5 which was even better, faster, and easier to use than all other YOLO models. As of now (January 2023), Ultralytics published YOLOv8 under the Ultralytics repository which is perhaps the best YOLO model till date. [21]

## **YOLOv8**

YOLOv8 is a new state-of-the-art computer vision model built by Ultralytics, the creators of YOLOv5. It is distributed under the GNU General Public License, which authorizes the user to freely share, modify and distribute the software. The YOLOv8 model contains out-of-the-box support for object detection, classification, and segmentation tasks, accessible through a Python package as well as a command line interface. [22]

### **1- Compare the changes in YOLOv8 with YOLOv5**

There are two main changes:

1. Anchor-Free Detection
2. Mosaic Augmentation

#### **Anchor-Free Detection**

Anchor Boxes generally improved training by increasing mAP. They were incorporated in previous YOLO models. In YOLOv8, the architecture moved away from Anchor Boxes for a few reasons:

1. Lack of Generalization: Training with prebuilt Anchors makes the model rigid and hard to fit on new data.
2. Lack of Proper Anchor Boxes in Irregularity: Irregularities cannot be mapped clearly with polygon anchor boxes. [23]

#### **Mosaic Data Augmentation**

During training, YOLOv8 does many augmentations to training images. One such augmentation is mosaic data augmentation. Mosaic data augmentation is a simple augmentation technique in which four different images are stitched together and fed into the model as input, forcing the model to learn objects in new locations, in partial occlusion, and against different surrounding pixels. [23]

## 2- Comparison between Versions and Models

As there are no official results from the paper, we are going to go through the official YOLO comparison plot from the repository.

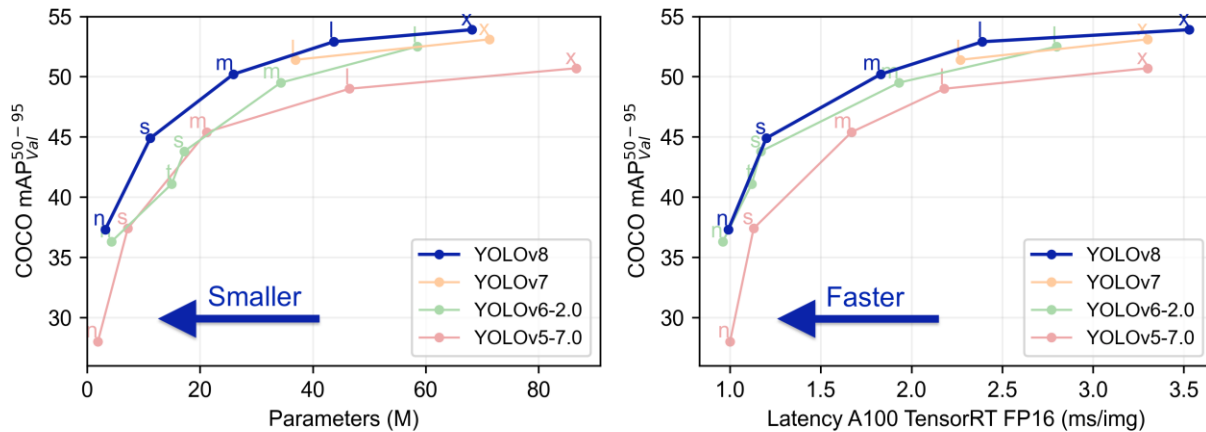


Figure 19: Comparison between yolo versions

As we can observe from the plot, YOLOv8 has more parameters than its predecessors, such as YOLOv5, but fewer parameters than YOLOv6. It offers about 33% more mAP for n-size models and generally a greater mAP across the board. From the second graph, we can observe faster inference time amongst all the other YOLO models. This is understandable and elegant. [23]

Within YOLOv8, we have different model sizes such as yolov8- n - nano, s - small, m - medium, l - large, and x - extra-large.

The model size is linearly proportional to mAP and inversely proportional to inference time. Bigger models take more inference time to accurately detect objects with higher mAP. Smaller models have faster inference time but have comparatively lesser mAP. Bigger models are better if we have less data. Smaller models are more efficient if we have less space (edge scenarios). [23]

Model	size (pixels)	mAP <sup>val</sup> 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOV8n	640	37.3	80.4	0.99	3.2	8.7
YOLOV8s	640	44.9	128.4	1.20	11.2	28.6
YOLOV8m	640	50.2	234.7	1.83	25.9	78.9
YOLOV8nl	640	52.9	375.2	2.39	43.7	165.2
YOLOV8x	640	53.9	479.1	3.53	68.2	257.8

Table 2: Comparison between YOLOv8 models

At Roboflow, we have drawn 100 sample datasets from Roboflow Universe, a repository of over 100,000 datasets, to evaluate how well models generalize to new domains. Our benchmark, developed with support from Intel, is a benchmark for computer vision practitioners. [24]

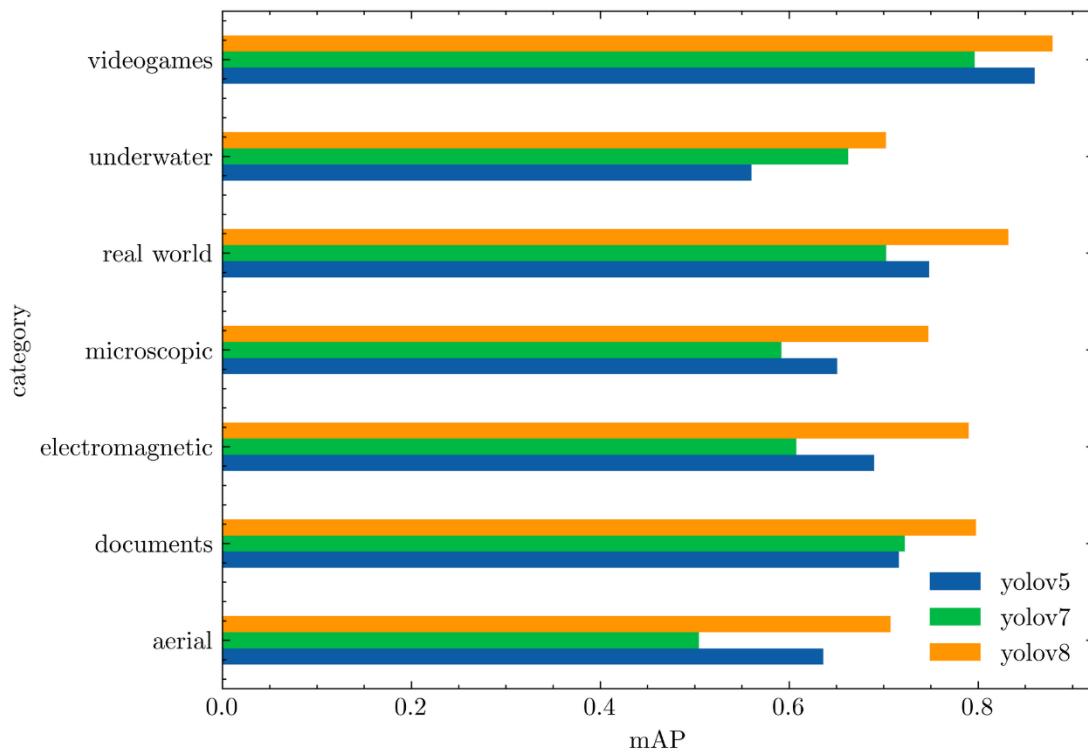


Figure 20: Comparison between YOLO versions

### 3.1.5 Speech Recognition

Speech recognition, also known as automatic speech recognition (ASR), Speech-to-text (STT), and computer speech recognition, is the ability of a machine or program to identify words spoken aloud and convert them into readable text. Rudimentary speech recognition software has a limited vocabulary and may only identify words and phrases when spoken clearly. More sophisticated software can handle natural speech, different accents, and various languages. [25]

Speech recognition systems use various algorithms and models to analyze and interpret spoken language.



*Figure 21: Speech recognition explanation*

Speech recognition and voice recognition are two different technologies and should not be confused:

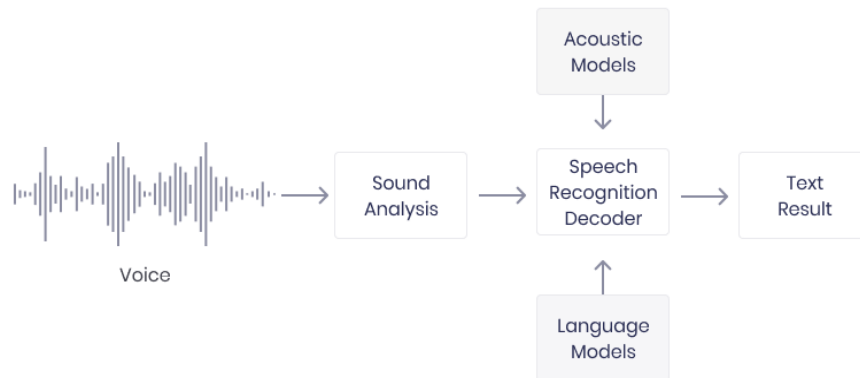
- Speech recognition is used to identify words in spoken language.
- Voice recognition is a biometric technology for identifying an individual's voice.

#### 1- Speech Recognition Works

1. Capture the human voice with physical devices like receivers or microphones.
2. Artificial intelligence (AI) and machine learning technologies filter background noise, accents, slang, and cross talk to increase precision and performance.
3. The hardware digitizes recorded sound vibrations into electrical signals.
4. The software attempts to identify sounds and phonemes from the signals and match these sounds to corresponding text. [26]

To meet these requirements, speech recognition systems use two types of models:

- **Acoustic models:** These represent the relationship between linguistic units of speech and audio signals.
- **Language models:** Here, sounds are matched with word sequences to distinguish between words that sound similar. [25]



*Figure 22: How to convert speech to text.*

## 2- Open-source Speech Recognition

There are various open-source speech recognition libraries and toolkits, each with its own strengths and focus, such as Kaldi, Mozilla DeepSpeech, Vosk, Julius, and CMU Sphinx.

Kaldi, known for its robustness and flexibility, offers advanced features and extensive support for acoustic and language models. Mozilla DeepSpeech, backed by a large community, focuses on providing accurate and state-of-the-art speech recognition using deep learning techniques. Julius, a popular open-source speech recognition toolkit, emphasizes real-time recognition with support for various languages. CMU Sphinx, a long-standing project, offers a suite of tools and libraries for speech recognition research and development.

## 3.2 Stick Implementation

### 3.2.1 Structure

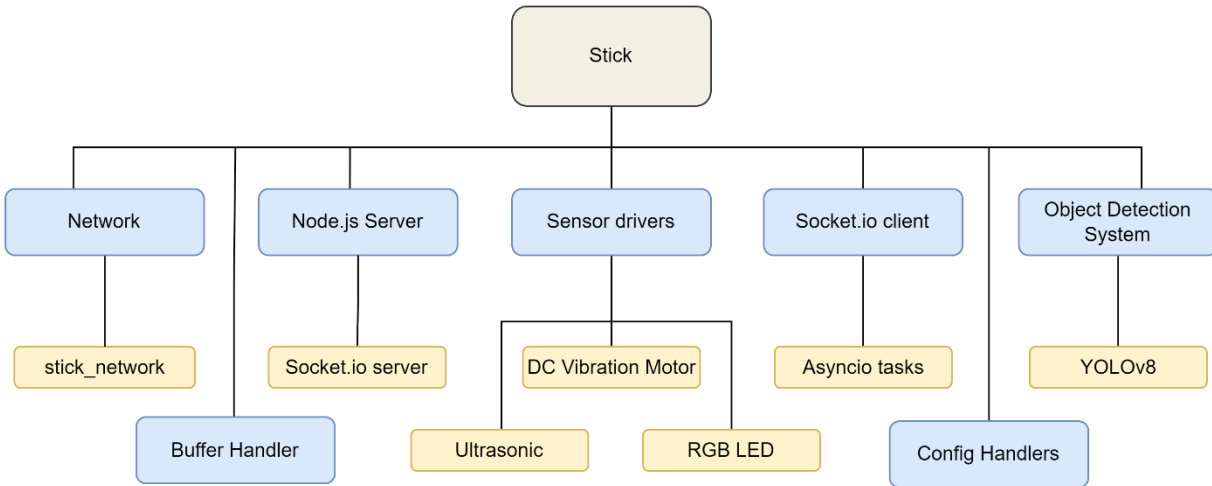


Figure 23: Stick system structure

### 3.2.2 Implementation

The implementation involves leveraging the power of the Raspberry Pi 4, along with the Raspbian operating system. We install Raspbian, designed specifically for Raspberry Pi hardware. Following the installation, we proceed with updates and post-configurations to ensure the Raspberry Pi 4 is ready for our project. Next, we focus on creating a network hotspot on the Raspberry Pi 4 to enable wireless communication between the mobile application and the Raspberry Pi. This eliminates the need for physical connections and allows for real-time interaction during prototype development. To facilitate seamless and instant updates between the smart stick and other devices, we incorporate a Node.js server using the Socket.IO library. Node.js, a runtime environment for executing JavaScript code on the server-side, enables us to create and host various services and applications.

## **Raspberry Pi 4**

### **Installation**

- 1- Download the latest version of Raspbian from the official Raspberry Pi website.
- 2- Use the Raspberry Pi Imager tool to write the Raspbian image onto an SD card.
- 3- Insert the SD card into the Raspberry Pi 4 and power it on to start the installation process.
- 4- Insert the SD card into the Raspberry Pi 4 and power it on to start the installation process.

### **Updates & post-configurations**

To get the raspberry pi 4 ready, there will be some additional steps that will be performed on the terminal:

- 1- Update Package Lists: **sudo apt-get update**
- 2- Upgrade Installed Packages: **sudo apt-get upgrade**
- 3- Firmware Update: **sudo rpi-update**
- 4- Raspberry Pi Configuration Tool: **sudo raspi-config**
  - Expand Filesystem: Go to "Advanced Options" and select "Expand Filesystem" to utilize the entire SD card.
  - Change User Password: Choose "Change User Password" to set a new password for the default 'pi' user.
  - Set Locale, Timezone, and Keyboard Layout: Navigate to "Localization Options" to configure these settings according to your location.
  - Enable SSH: Under "Interfacing Options," select "SSH" and enable it for remote access.
  - Enable VNC: to use VNC (Virtual Network Computing), select "Interfacing Options" and enable VNC.
- 5- Update pip: **python -m pip install --upgrade pip**



## Network

Creating a network hotspot on a Raspberry Pi 4 allows wireless communication between the mobile application and the Raspberry Pi. It provides mobility, a controlled prototyping environment, seamless data exchange, security, and scalability. The hotspot eliminates the need for physical connections and enables real-time interaction for prototype development.

### Creation procedures

To create a hotspot that runs when booting up on a Raspberry Pi 4, you can use the "dnsmasq" and "hostapd" tools. Here's a step-by-step guide to setting it up:

- 1- Start by installing the necessary packages. Open a terminal on your Raspberry Pi or connect to it remotely via SSH, then run the following commands:

```
sudo apt update
```

```
sudo apt install dnsmasq hostapd
```

- 2- Once the installation is complete, you'll need to configure dnsmasq. Create a new configuration file by running the following command:

```
sudo nano /etc/dnsmasq.conf
```

Add the following lines to the file:

```
interface=wlan0
```

```
dhcp-range=192.168.4.2,192.168.4.20,255.255.255.0,24h
```

```
domain=wlan
```

```
address=/gw.wlan/192.168.4.1
```

Save the file by pressing Ctrl+O, then exit nano by pressing Ctrl+X.

- 3- Next, you need to configure hostapd. Create a new configuration file by running the following command:

```
sudo nano /etc/hostapd/hostapd.conf
```

Add the following lines to the file:

```
interface=wlan0
driver=nl80211
ssid=YourHotspotName
hw_mode=g
channel=7
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=YourPassword
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Replace "YourHotspotName" with the desired name for your hotspot, and "YourPassword" with the password you want to use for connecting to the hotspot. Save the file by pressing Ctrl+O, then exit nano by pressing Ctrl+X.

4- Now, you need to tell the system where to find the hostapd.conf file. Open the following file:

```
sudo nano /etc/default/hostapd
```

Find the line #DAEMON\_CONF="" and replace it with:

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

Save the file and exit.

- 5- Next, you need to configure the network interface by modifying the `dhcpcd` configuration file. Open it by running the following command:

```
sudo nano /etc/dhcpcd.conf
```

Scroll to the bottom of the file and add the following lines:

```
interface wlan0
```

```
static ip_address=192.168.4.1/24
```

```
nohook wpa_supplicant
```

- 6- Finally, enable and start the services by running the following commands:

```
sudo systemctl enable dnsmasq
```

```
sudo systemctl enable hostapd
```

```
sudo systemctl start dnsmasq
```

```
sudo systemctl start hostapd
```

These commands ensure that the hotspot services start automatically on boot and are started immediately.

That's it! The Raspberry Pi 4 should now boot up with a hotspot running. Other devices can connect to it using the SSID you specified, and they will be assigned IP addresses in the range defined in the `dnsmasq.conf` file.

## **Node.js server**

Node.js is a popular runtime environment that allows you to run JavaScript code on the server-side. When installed on a Raspberry Pi, Node.js enables us to create and host various services and applications.

### **Installation**

The Raspberry Pi 4 must be powered on and connected to the internet. It's recommended to have an up-to-date operating system as well as the packages. The following steps are used to install Node.js on RPI 4:

- 1- Download the Node.js binary package for Linux (x64) ARMv8 from the official Node.js website. Since you mentioned version 18.16.1, you can use the following command to download it:

```
wget https://nodejs.org/dist/v20.4.0/node-v20.4.0-linux-arm64.tar.xz
```

- 2- Extract the downloaded archive using the following command:

```
tar -xf node-v20.4.0-linux-arm64.tar.xz
```

- 3- Move the extracted files to the desired location. For example, you can move them to the /usr/local directory:

```
sudo mv node-v18.16.1-linux-armv7l /usr/local/nodejs
```

- 4- Add the Node.js executable to the system's PATH environment variable. Edit the .profile file in your home directory by running:

```
nano ~/.profile
```

Add the following line at the end of the file:

```
export PATH=/usr/local/nodejs/bin:$PATH
```

Save the file by pressing Ctrl + X, followed by Y, and then press Enter to confirm the filename.

5- Load the updated profile by running:

```
source ~/.profile
```

6- To verify that Node.js is installed correctly, you can check the version by running:

```
node -v
```

It should display the installed Node.js version, which in this case should be v18.16.1. That's it! The Node.js has been successfully installed Node.js on the Raspberry Pi 4.

## **Server creation**

To create a server, get into a separate folder for the server. Then, run the following command: **npm init -y** to initialize the node.js project. Now it is the time to install Express.js using: **npm install express** which is a package built on Node.js to facilitate server's implementations. After installing express and writing server code in server.js, the server can be started by executing **node server.js**

## **Socket.io server**

Socket.io is a popular library that provides real-time bidirectional communication between the server and clients. To establish a connection between the RPI 4 running Node.js and the mobile application, we used Socket.io. Installation procedures:

7- Inside the Node.js server open the terminal and perform:

```
Npm install socket.io
```

8- In server.js, import the Socket.IO module and create a server instance.

## **Socket.io client**

Socket.io is a popular library for real-time web applications that enables bidirectional communication between the client (RPI 4 & the mobile application) and the server. Socket.io provides both server-side and client-side components to facilitate real-time data transfer.

## **Installation**

Before getting started, the python package installer (pip) needs to be updated to make sure that everything is ready to install socket.io client. To install socket.io client, perform `pip install "python-socketio[client]" asyncio`

Now we have the socket.io client and asyncio libraries installed on the RPI4.

## **Scenario**

We have used asyncio to implement non-blocking (asynchronous) socket.io client. This implementation allows the RPI4 to process incoming data without blocking the main program and to have everything working in a parallel manner. The socket.io client have different events that triggers different actions such as “connection” which is fired every time the client connects to the server. Another important event is “message”, it is responsible for receiving and sending messages from/to the server. This one takes a separate asyncio task to keep listening for incoming messages for processing.

## **Buffers**

We have designed a specific buffer for communication between the RPI 4 and the mobile application. They are made of “ operation separation data “ for example: “Stream: Hello from RPI4”. The operation Stream is responsible for streaming incoming data from RPI4 on the speaker of the mobile app. We have different buffers to control the mode of operation of the RPI and changing the language of the system.

## **Buffer handler**

After receiving a message from socket.io client, the message is delivered to the buffer handler. The handler tries to tokenize and scan the message to see if it follows the structure we made for buffers. If the message passes, it will be separated into two parts operation and data. Based on the operation, there will be different actions on the data.

## **Configurations**

The system is controlled to produce output. The output can vary from time to time based on user` needs. That`s why we were motivated to have parts of the system modified based on some parameters such as the system language. This implementation has added flexibility to the system in which its outcome can vary based on what is needed. These parameters are saved inside a file called “config.json”. There are handlers made specifically for adding, modifying, and reading configurations.

## **Sensor drivers**

Based on the implementation of our project in which the different parts of it are loosely coupled. This implementation allows us to modify, integrate, add, and test each part of the system. To keep this flexibility in the system, we needed to create fully editable drivers for the sensors in which they are used based on the task needed to be performed not the behavior of the system.

### **Ultrasonic**

The ultrasonic sensor emits high-frequency sound waves. These sound waves travel through the air and bounce off objects. The sensor then detects the echoes and calculates the distance based on the time it takes for the sound waves to return. Following the implementation, the driver of the ultrasonic is responsible for returning the distance between the sensor and the object that the emitted wave hit.

```
def get_distance(TRIG, ECHO): returns the distance.
```

### **DC vibration motor**

The driver we made for the motor has a high level of abstraction, as we have made functions to turn it on and off without the need for handling logic behavior.

```
def turn_on_vibration_motor(self):
```

```
def turn_off_vibration_motor(self):
```

These functions hide the implementation of logic behavior and hardware things.

### **RGB LED**

We have made three functions to control which color should be on.

```
def turn_on_red_led(self): def turn_on_green_led(self):
```

```
def turn_on_blue_led(self):
```



## Object detection

Object detection is a computer vision task that involves identifying and locating objects within an image or a video. YOLO (You Only Look Once) is a popular object detection algorithm known for its real-time performance and accuracy. YOLOv8 is the latest SOTA algorithm.

### Ultralytics YOLOv8

It is the latest version of the acclaimed real-time object detection and image segmentation model. It is built on cutting-edge advancements in deep learning and computer vision, offering unparalleled performance in terms of speed and accuracy. It can be installed via the python package installer (pip):

```
pip install cv2
```

```
pip install ultralytics
```

It has different models, but we have used the yolov8n. This model is suitable for our requirements. It can detect 5 frames/second which is good.

The implementation of object detection part is written to follow the loosely implementation of the system. So, we can use it in different modes and configurations. Example:

```
detector.get_detected_objects(detector.detect_object(detector.get_picture_from_camera(), object_detection_model))
```

This function takes output python object from the yolo engine then out list detected objects and returns them.

### 3.2.3 System logic

Before getting started to show how the system functions to get the output. It is important to mention how things works. When the RPI 4 boots up, it automatically boots up the Node.js server then it approaches to start the project from the file “stick.py”.

The system is well structured in a human readable manner, first there is a main entrance file which is called “stick.py”, and two other folders “utils” and “systems”. The systems folder contains the object detection system functionality and other systems as well. The utils contain the raspberry pi class, socket.io client, buffers, and configuration.

It starts by creating an event-loop for asyncio for later parallel manner. Then, it starts with initializing the project via `project_init()` function in which an object from RaspberryPi class is instantiated. The raspberry pi object contains everything about GPIO related to the RPI and their callbacks and mutable system parameters. The next step is to load the models such as object detection model to get it ready to save time while operating. The synchronous initializing takes seconds at the booting up the system, but it saves time later.

After the initialization, the system stays in a while loop, looping over the current mode of operation. There are two modes `DetectObjects`, the default one, and `ExploreEnviorment`.

The `DetectObjects` mode is the main logic of the project in which the stick is waiting for ultrasonic to get triggered if an obstacle crosses a pre-defined distance which identifies the danger zone if something goes below it. When the ultrasonic is triggered, the system triggers the vibration motor to notify the user that there is something in front of him/her. At the same moment, the LED color is changed to

show that the system is processing output. Then, the camera takes a snapshot then it is sent to the object detection model for processing. After that, the result is held into Stream buffer to be sent over the Node.js server and socket.io to be delivered to the mobile application. The mobile application is responsible for receiving and streaming data over the current audio device whether it is the speaker or the headset.

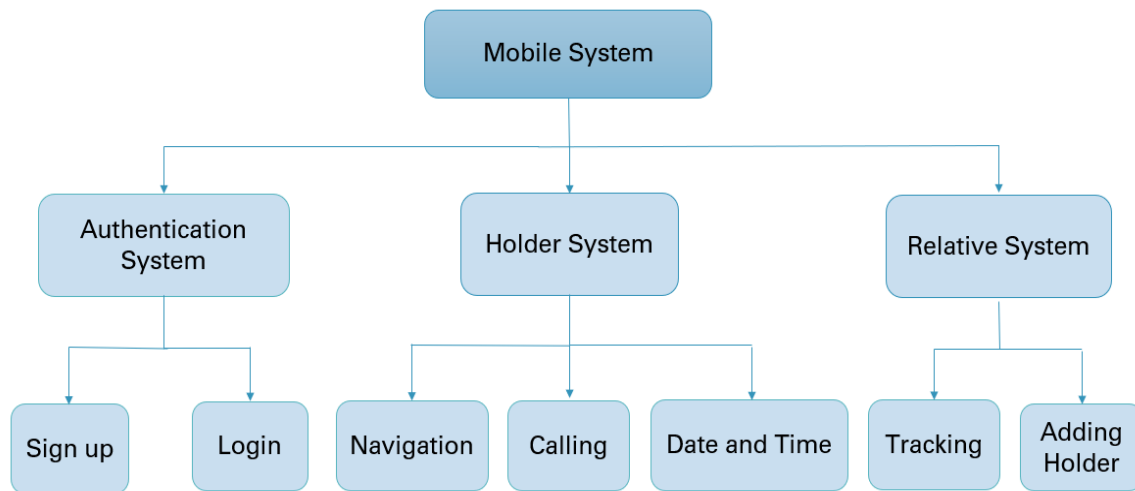
The ExploreEnvironment is like the first one, but it does not use the ultrasonic sensor nor the vibration motor. It keeps detecting what is in the frame of the camera and send the results to the mobile for identifying what is ahead of the user.

Finally, the stick is the heart of the project and the controller that manages what should be processed on the hardware and the mobile application. The reason why we have created the mobile application is to reduce the hardware by adding some additional features without requiring more hardware parts. This helped us to reduce the cost of the project and focus on enhancing the experience for the user.

## 3.3 Mobile Application

### 3.3.1 Structure

Our application serves two distinct user groups. The first group consists of "holders" who possess a smart stick. These holders are the primary users of the application and have exclusive access to certain features and functionalities. The second group comprises the relatives of the holders.



*Figure 24: Mobile System Structure*

### 3.3.2 Authentication System

To use the application, users need to go through a registration process and have their information stored in the database. Once registered, they can log in to access the application's features and functionalities, depending on whether they are a relative or a holder. Upon logging in, users will be prompted to specify their role as either a relative or a holder. Based on their selection, they will be directed to the corresponding section of the application. If the user identifies themselves as a relative. These functionalities enable the relative to stay connected with and monitor the movements of their respective holders. On the other hand, if the user identifies themselves as a holder, they will have access to features specific to their role.

### **3.3.3 Relative System**

The relative system implemented in the application acts as a communication bridge between the holders and their relatives. It operates similarly to a friend request system, where relatives can send friend requests to their respective holders. When a holder receives a friend request from a relative, they have the option to accept or decline it. Once the holder accepts the request, a connection is established between the holder and the relative within the application. This connection allows for seamless communication and interaction between the two parties. By implementing this relative system, the application fosters closer relationships and facilitates convenient communication channels for both the holders and their relatives.

### **Tracking System**

Once the connection is established between the holder and the relative within the application, the relative gains the ability to track the holder in real-time. This is made possible by accessing the latitude and longitude coordinates of the holder's current location. The application utilizes Google Maps integration to provide a map view that displays the real-time movements of the holder. Whenever the holder moves, the relative can access their updated location on the map, allowing for continuous tracking. In addition to real-time tracking, the relative also has access to the "last seen" information, which indicates the holder's recent location before they turned off our application. Even if the holder turns off their smart stick, the application retains the last known location of the holder. This feature ensures that the relative can still refer to the last seen location for reference or assistance.

### **3.3.4 Holder System**

Holder speaks into the mobile application, the application employs speech recognition technology to convert their speech into text.

This functionality allows holders to interact with the application using voice commands or by dictating messages instead of relying solely on manual input. By converting spoken words into text, the application can process and understand the holder's intent. The application leverages speech recognition technology to accurately convert spoken words into text, ensuring effective communication and interaction for holders. This feature adds an additional layer of convenience and usability to the application, making it more accessible and user-friendly.

#### **Navigation System:**

By utilizing voice commands, holders can navigate to various destinations within the application. For instance, they can use voice commands to specify their desired location or navigate to their saved home address, which is stored using Google Maps instructions.

When a holder wants to navigate to a specific place, they can simply speak the destination or provide relevant details using their voice. The application's voice recognition feature will convert the spoken words into text, allowing the application to interpret the request accurately.

For example, a holder can say, "Navigate to the grocery store" or "Take me to my restaurant." The application will process the voice command and initiate the navigation process, providing directions to the desired location.

Additionally, if a holder has previously saved their home address within the application, they can use a voice command like "Navigate home" to receive

directions to their saved address. The application will utilize the stored information and Google Maps integration to generate the appropriate route.

By incorporating voice navigation functionality, the application enables holders to interact with the navigation features in a hands-free manner. This improves user convenience and accessibility, allowing holders to effortlessly navigate to desired locations using voice commands.

### **Calling System :**

Through the voice command feature in the application, holders have the capability to make phone calls to their relatives or any other desired phone number. When a holder wants to make a call, they can simply use their voice to initiate the command. The application's speech recognition technology will convert the spoken command into text, enabling the application to identify the intended recipient or phone number. If the relative's number is saved within the application's database, the application will access the contact information and initiate the call to the specified relative. This allows for quick and convenient communication between the holder and their relatives.

### **Date and time**

Certainly! In addition to the other features, holders can also retrieve the current date and time through the application. By using a voice command, holders can request the application to provide them with the current date and time information with default language for the device.

## **Chapter 4: Testing and Results**

### **4.1 Stick**

#### **4.1.1 Object Detection**

To evaluate the performance of the object detection feature in our project, we conducted a series of tests. The goal was to assess the accuracy and speed of the YOLOv8 algorithm implemented on the Raspberry Pi.

#### **Test Setup:**

- **Test Images:** We selected two representative images that contained various objects commonly encountered in the surroundings of visually impaired individuals.
- **Detection Procedure:** We ran the object detection algorithm on the Raspberry Pi, utilizing the YOLOv8 model. The algorithm analyzed each image and identified the objects present along with their bounding boxes and labels.
- **Time Measurement:** To assess the speed of the object detection process, we recorded the time taken for each detection operation.

#### **Test Results:**

##### **1- Image one**

- **Detected Objects:** The YOLOv8 algorithm successfully identified objects.
- **Bounding Boxes and Labels:** The algorithm accurately outlined the detected objects with bounding boxes and provided corresponding labels for each object.
- **Detection Time:** The average time taken for object detection in Image 1 was measured to be 226ms.



## Detection Output

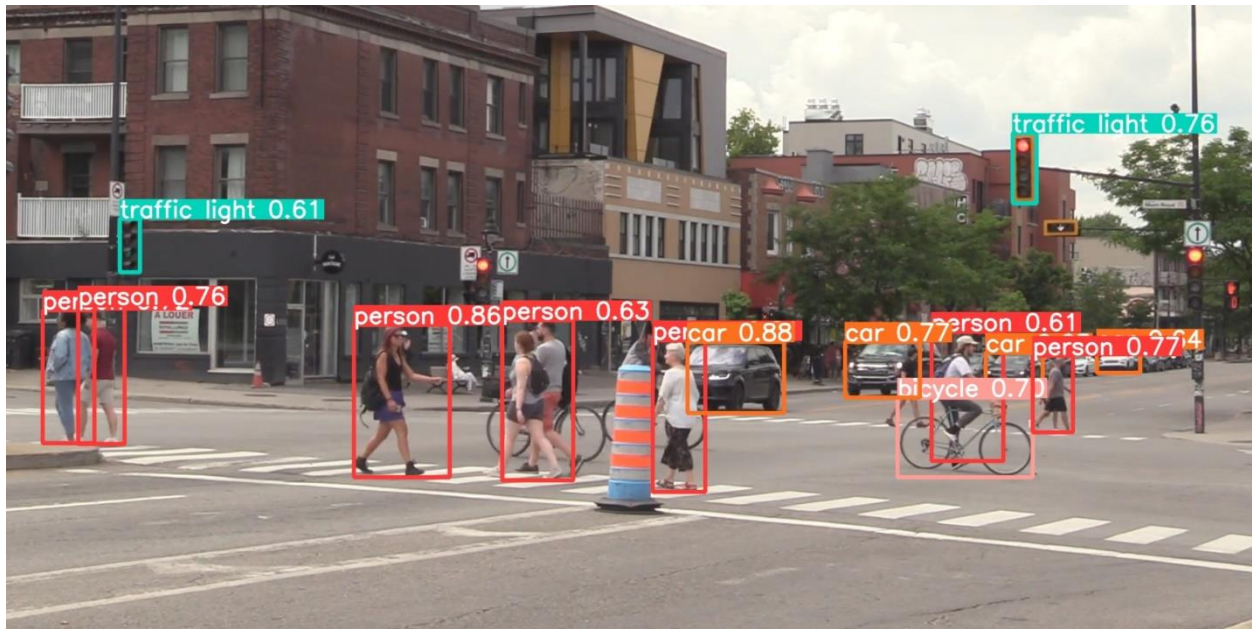


Figure 25: Image 1 object detection

## Raspberry pi detection log

```
0: 192x224 2 traffic light, 7 person, 4 car, 1 bicycle, 221.0ms  
Speed: 2.6ms preprocess, 221.0ms inference, 2.4ms postprocess per image at shape (1, 3, 224, 224)
```

Figure 26: Result image 1 object detection

## 2- Image two:

- **Detected Objects:** The YOLOv8 algorithm accurately identified objects like bicycles, benches, and trees in Image 2.
- **Bounding Boxes and Labels:** The algorithm precisely marked the detected objects with bounding boxes and provided corresponding labels for easy identification.
- **Detection Time:** The average time taken for object detection in Image 2 was measured to be 0.72 seconds.

Detection Output

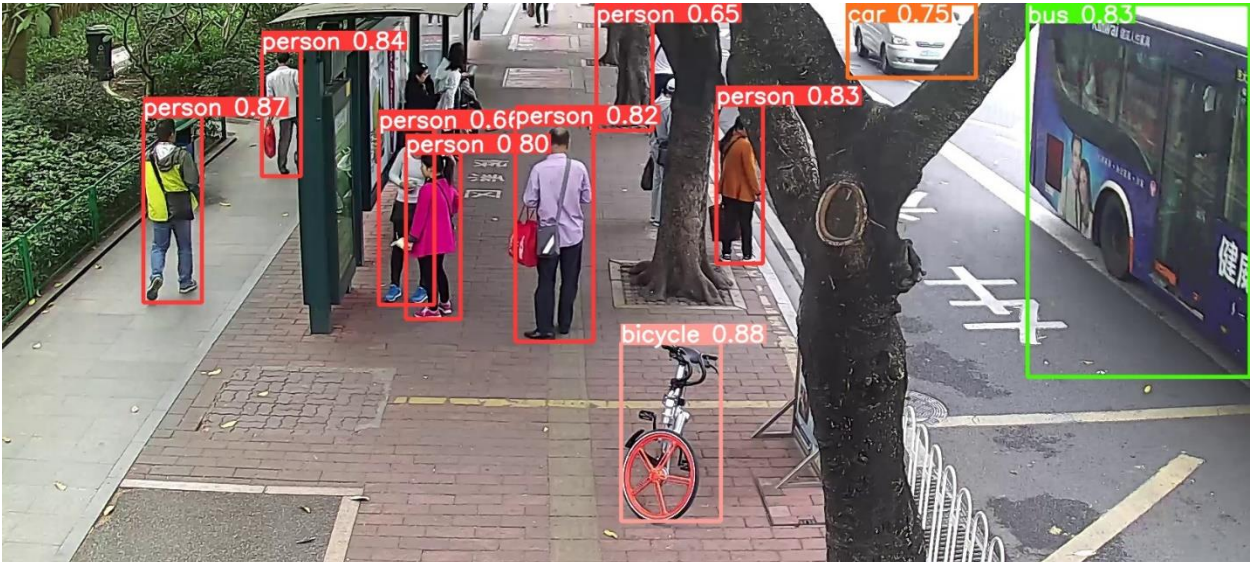


Figure 27: Image 2 object detection

Raspberry pi detection log

```
0: 192x224 7 person, 1 car, 1 bicycle, 1 bus, 245.5ms
Speed: 2.8ms preprocess, 245.5ms inference, 1.4ms postprocess per image at shape (1, 3, 224, 224)
```

Figure 28: Result image 2 object detection

These test results validate the effectiveness of the YOLOv8 algorithm in accurately detecting objects in real-world scenarios. The average detection times indicate efficient processing on the Raspberry Pi, ensuring timely feedback for visually impaired individuals during navigation. These outcomes contribute to the overall reliability and usability of our smart stick system, enhancing the independence and safety of visually impaired individuals.

The table represents the average detection time for each image over 10 repetitions. The values demonstrate consistent performance and provide insight into the speed of the object detection process.

Runtime no.	Pre-process	Inference	NMS
1	2.6ms	221.0ms	2.4ms
2	2.8ms	215.4ms	1.4ms
3	2.8ms	219.0ms	1.4ms
4	2.7ms	218.6ms	2.4ms
5	2.6ms	216.4ms	2.4ms
6	2.8ms	215.4ms	1.4ms
7	2.7ms	219.3ms	2.4ms
8	2.4ms	220.4ms	2.4ms
9	2.6ms	221.0ms	2.4ms
10	2.7ms	216.4ms	1.4ms
Sum	26.7ms	2182.9	20ms
Avg	222.96ms		

Table 3: Average detection time for Image 2

#### 4.1.2 Object Distance

To assess the effectiveness of the ultrasonic sensor in measuring object distances, we conducted a series of tests. The goal was to evaluate the accuracy and reliability of the sensor in detecting obstacles and providing distance measurements.

```

eslam@raspberrypi: ~/Smart-stick-for-blind-people-sourcecode/Stick
Distance between the stick and an obstacle: 54.86
Distance between the stick and an obstacle: 54.96
Distance between the stick and an obstacle: 55.9
Distance between the stick and an obstacle: 56.02
Distance between the stick and an obstacle: 56.52
Distance between the stick and an obstacle: 55.76
Distance between the stick and an obstacle: 56.25
Distance between the stick and an obstacle: 2342.56
Distance between the stick and an obstacle: 2346.7

```

Figure 29: Test object distance

### 4.1.3 Node.js Server

Launching the server in the development environment.

```
eslam@raspberrypi: ~/Smart-stick-for-blind-people-sourcecode/Server
eslam@raspberrypi:~/Smart-stick-for-blind-people-sourcecode/Server $ npm run start:dev
> smart-stick-server@1.0.0 start:dev
> nodemon server.js

[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
App running on port 5000...
```

Figure 30: Launching Node.js server

### 4.1.4 Socket.io responses (server/client)

Socket.io server responses when a client connects and sends a message and leaves.

```
eslam@raspberrypi: ~/Smart-stick-for-blind-people-sourcecode/Server
eslam@raspberrypi:~/Smart-stick-for-blind-people-sourcecode/Server $ npm run start:dev
> smart-stick-server@1.0.0 start:dev
> nodemon server.js

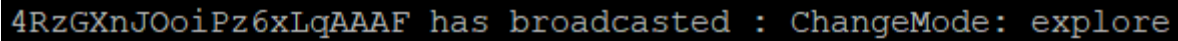
[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
App running on port 5000...
5DZCmlvkojNDdcpWAAAB has connected.
5DZCmlvkojNDdcpWAAAB has broadcasted : Hello from the mobile application
Acm6fT-rEPD9FQ87AAAD has connected.
Acm6fT-rEPD9FQ87AAAD has broadcasted : Hello from RPI4, I am ready to receive from the mobile.
5DZCmlvkojNDdcpWAAAB has left.
```

Figure 31: Socket.io responses

The following test cases are designed to validate the functionality and reliability of the Smart Stick system during mode changes. These test cases focus on the communication between the mobile application and the Raspberry Pi, ensuring that mode change requests are successfully transmitted.

## Test Case 1: Mobile has sent operation to change mode

**Objective:** To verify that the Raspberry Pi (RPI) successfully receives and changes the mode when instructed by the mobile application.

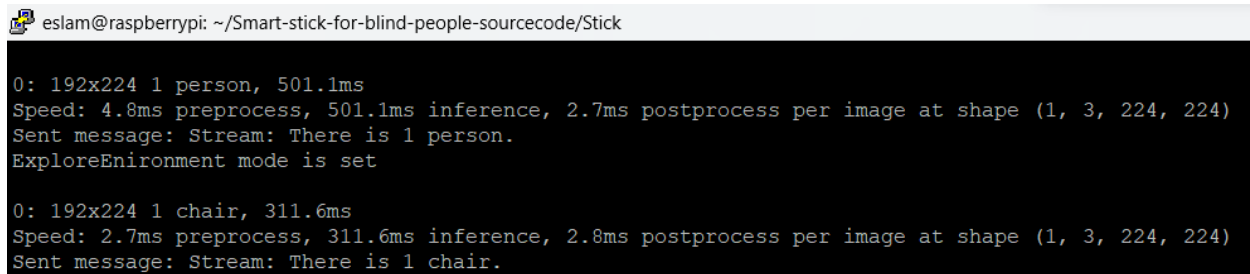


```
4RzGXnJOoiPz6xLqAAAF has broadcasted : ChangeMode: explore
```

*Figure 32: Server response: mobile requesting to change the mode to explore*

### Expected Result:

The Raspberry Pi receives the mode change request from the mobile application and updates its mode accordingly. The mobile application reflects the updated mode status, confirming the successful mode change operation.



```
eslam@raspberrypi: ~/Smart-stick-for-blind-people-sourcecode/Stick
0: 192x224 1 person, 501.1ms
Speed: 4.8ms preprocess, 501.1ms inference, 2.7ms postprocess per image at shape (1, 3, 224, 224)
Sent message: Stream: There is 1 person.
ExploreEnironment mode is set

0: 192x224 1 chair, 311.6ms
Speed: 2.7ms preprocess, 311.6ms inference, 2.8ms postprocess per image at shape (1, 3, 224, 224)
Sent message: Stream: There is 1 chair.
```

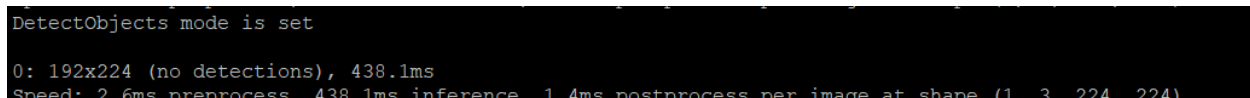
*Figure 33: Raspberry pi responses to mode changing operation from the mobile*

## Test Case 2: Changing Mode to DetectObjects

**Objective:** To verify that the Raspberry Pi accurately changes its mode to "Detect Objects" when instructed to do so.

### Expected Result:

The Raspberry Pi receives the mode change request to DetectObjects and updates its mode accordingly. The mobile application reflects the updated mode status as DetectObjects.



```
DetectObjects mode is set

0: 192x224 (no detections), 438.1ms
Speed: 2.6ms preprocess, 438.1ms inference, 1.4ms postprocess per image at shape (1, 3, 224, 224)
```

*Figure 34: Raspberry pi responses to mode changing to detect objects from the mobile*

## 4.2 Mobile Application

**Test Case:** Validation of Authentication with Incorrect Email or Password

**Objective:** To verify that the authentication process correctly handles the scenario when a user enters an incorrect email or password.

**Test Steps:**

1. Enter an incorrect email and correct password combination.
2. Observe the system response.

**Expected Result:** The system should display an appropriate error message indicating that the email or password is incorrect.

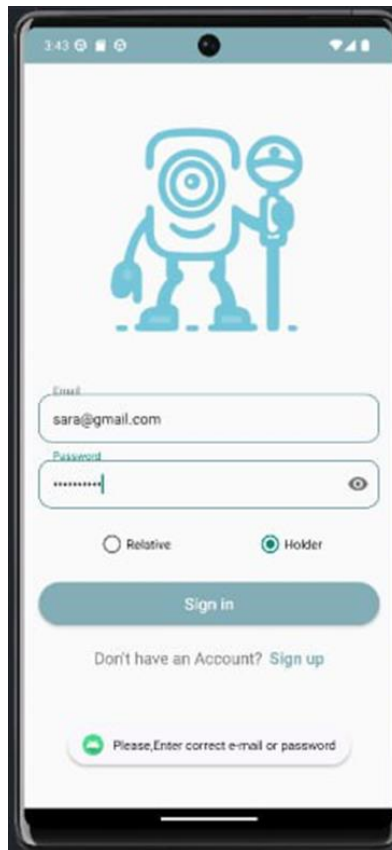


Figure 35: Error message displaying



## **Test Case:** Speech Recognition – Voice-to-Text

**Objective:** To verify that the speech recognition feature correctly converts the user's spoken words into text when the user presses the speech button.

### **Test Steps:**

1. Open the application.
2. Locate the speech button/icon.
3. Press the speech button to activate the speech recognition feature.
4. Clearly speak a predefined phrase or sentence.
5. Observe the system response.

**Expected Result:** The system should accurately convert the spoken words into text and display the converted text on the screen.



*Figure 36: Text to speech*

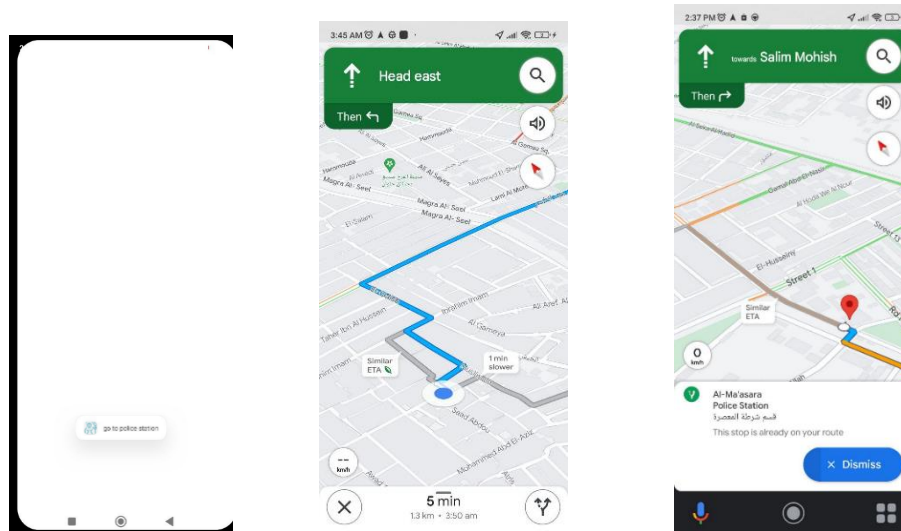
## **Test Case:** Navigation System - Voice Command for Police Station

**Objective:** To verify that the navigation system accurately interprets the user's voice command to navigate to a desired location, specifically a police station on the map.

### **Test Steps:**

1. Open the application and ensure the navigation feature is accessible.
2. Activate the voice command functionality by pressing the microphone or voice button.
3. Clearly speak the voice command, "Go to the police station " or a similar predefined phrase indicating the desired location.
4. Observe the system response and navigation instructions.

**Expected Result:** The navigation system should accurately interpret the voice command and provide appropriate directions to the police station location on the map. The system should generate a clear and concise route, guiding the user towards the desired destination.



*Figure 37: Voice command guiding*

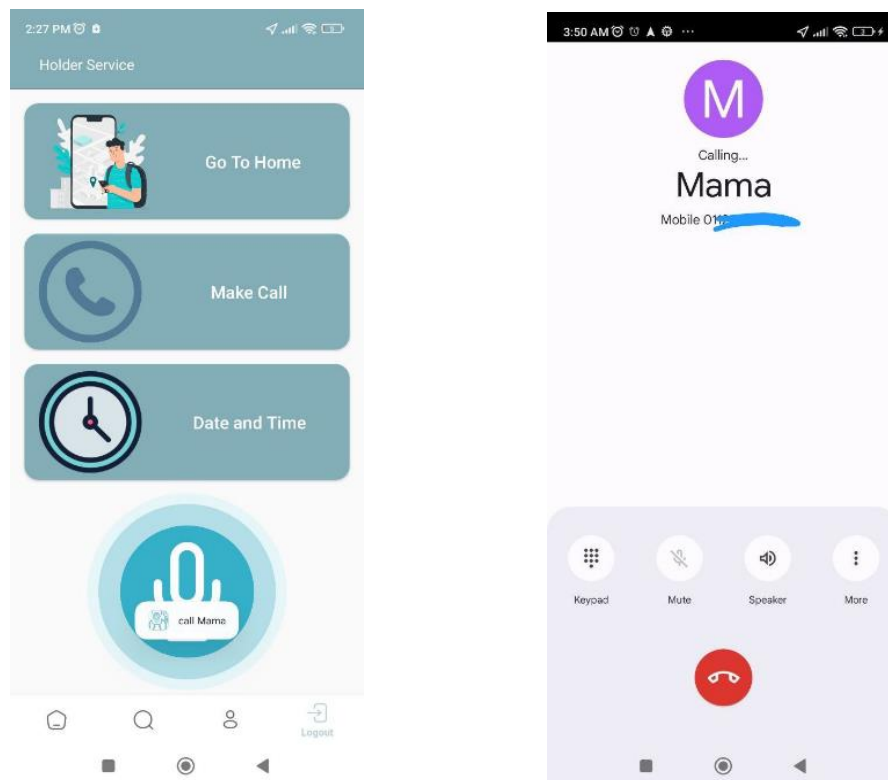


**Test Case:** To verify that the calling system enables the user to initiate a phone call to any desired number by using their voice.

**Test Steps:**

1. Open the application
2. Activate the voice command functionality by pressing the microphone or voice button.
3. Clearly speak the voice command, "Call [desired phone number]" or a similar predefined phrase indicating the desired number.
4. Observe the system response and the initiation of the phone call.

**Expected Result:** The calling system should accurately interpret the voice command and initiate a phone call to the desired number. The system should successfully establish the call, allowing the user to communicate with the desired contact.



*Figure 38: According to voice command user initiate a phone call*

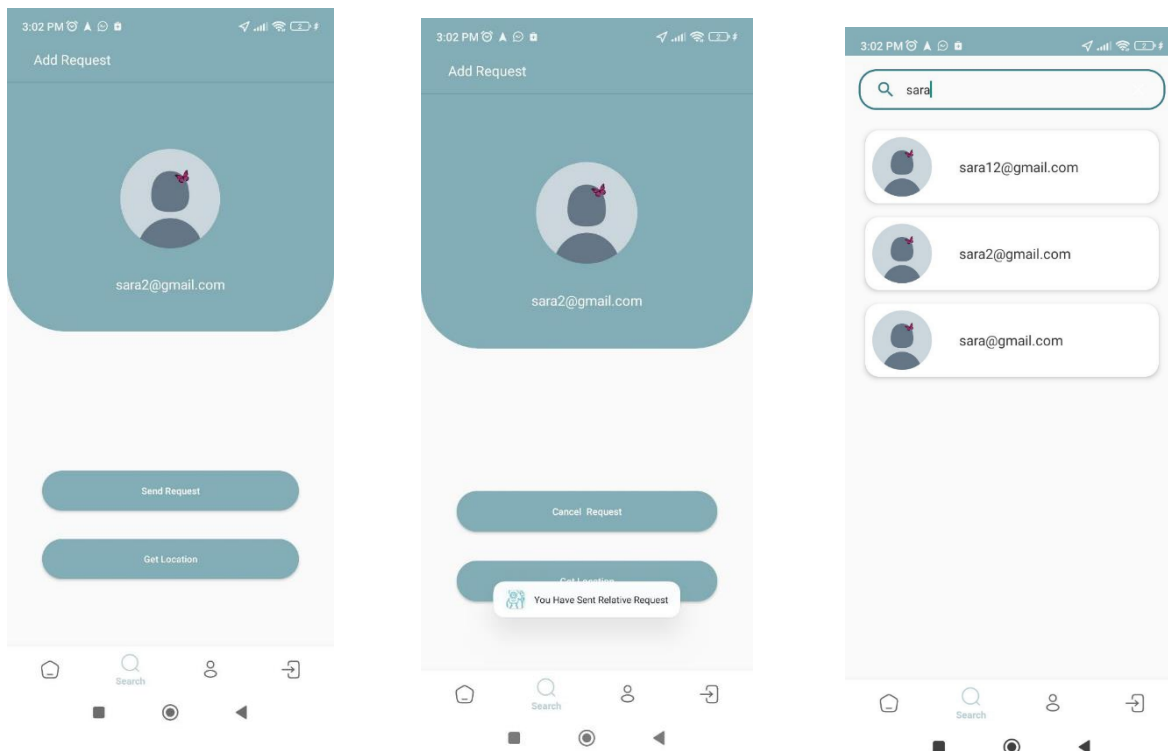
## **Test Case:** Relative System - Request for Communication via Tracking System

**Objective:** To verify that the relative system allows a relative to send a communication request to the holder (user) through to can tracking system.

### **Test Steps:**

1. Open the application and navigate to the search.
2. Select the desired holder (user) from the available list of contacts.
3. Initiate the communication request by pressing the send request button .
4. Observe the system response.

**Expected Result:** The relative system should successfully send a communication request to the selected holder



*Figure 39: Relative send request to holder*

## Test Case: Tracking System - Relative Tracking Holder

**Objective:** To verify that the tracking system allows a relative to track the location of their assigned holder by selecting the corresponding holder from the list.

### Test Steps:

1. Open the application.
2. Access the list of assigned holders available for tracking.
3. Select the desired holder from the list by tapping on their name or profile.
4. Observe the system response.

**Expected Result:** The tracking system should successfully establish the tracking connection between the relative and the selected holder. The system should display the real-time location of the holder on a map or provide location details, allowing the relative to track the holder's whereabouts.

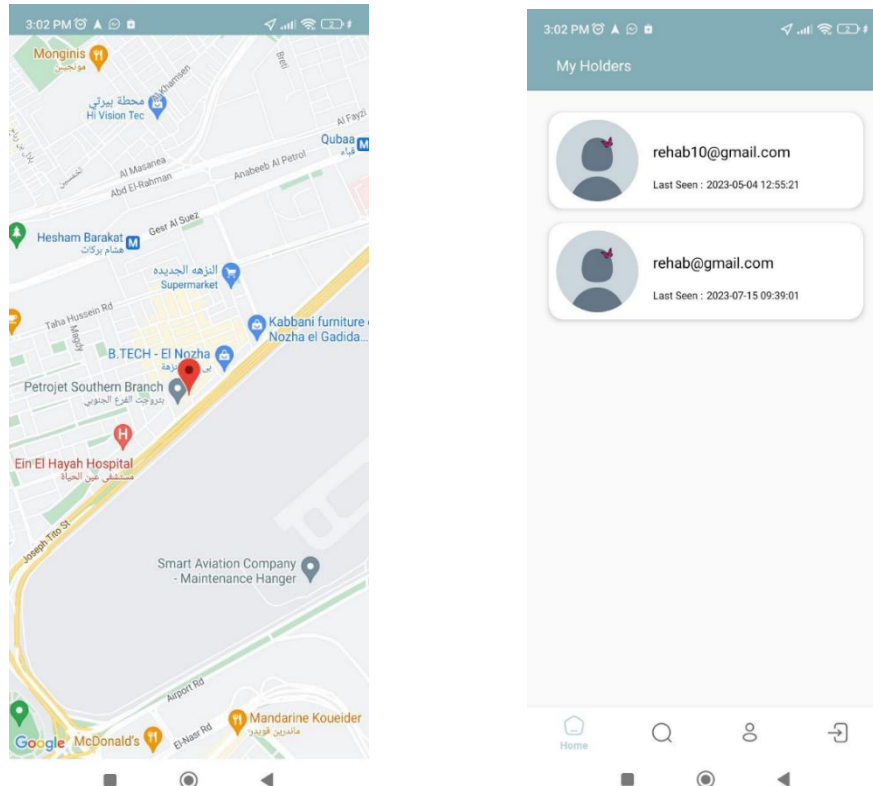


Figure 40: Relative can track his holder in real-time

# **Chapter 5: Hardware**

## **5.1 Components**

- Rpi Camera v2.
- Ultrasonic.
- Raspberry Pi4 model B.
- DC Vibration motor.
- RGB LED.
- Power bank.
- Cable Type-C.
- Relay.
- Resistors.

## RPI Camera V2

The Raspberry Pi Camera Module v2 is an 8-megapixel camera that can be used with the Raspberry Pi single-board computer. It has a fixed focus lens and can record video in resolutions up to 1080p30. The camera attaches to the Raspberry Pi using a 15cm ribbon cable. [27]



*Figure 41: RPI Camera v2*

Here are some of the features of the Raspberry Pi Camera Module v2:

- 8-megapixel Sony IMX219 image sensor, Fixed focus lens.
- Supports 1080p30, 720p60, and VGA90 video modes.
- Still capture up to 3280 x 2464 pixels.
- Attaches to Raspberry Pi using a 15cm ribbon cable.
- Works with all models of Raspberry Pi 1, 2, 3, and 4. [27]

## DC Vibration Motor

A DC vibrator motor is a type of electric motor commonly used for generating vibrations in various applications. It typically consists of a permanent magnet, an armature, and a coil. When a current is passed through the coil, it creates a magnetic field that interacts with the permanent magnet, causing the armature to move back and forth rapidly, generating vibrations. The smart stick incorporates a DC vibrator motor that activates when object detection is triggered, providing vibration feedback. [28]



*Figure 42: DC Vibration Motor*

## Ultrasonic

An ultrasonic sensor is a device that generates high-frequency sound waves. These sound waves are emitted towards an obstacle and then bounce back along their original path, where they are received by the sensor. By measuring the time, it takes for the signal to travel, the distance between the sensor and the obstacle can be calculated. [29]

These sensors are also capable of detecting objects. With a range of approximately 2m to 100m, they provide early detection of obstacles approaching from a distance. As the user walks, the sensor will emit sound waves and measure the distance to any objects in its path. If the sensor detects an obstacle, it will alert the user by activating a buzzer or vibration motor. [29]



*Figure 43: Ultrasonic sensor*

This allows the user to be aware of obstacles in their path and take steps to avoid them. This can help to improve the safety and independence of visually impaired people.

Here are some of the advantages of using an ultrasonic sensor in a smart stick:

- **Accuracy:** Ultrasonic sensors can accurately measure the distance to an object, even in low-light conditions. This makes them a reliable way to detect obstacles.
- **Range:** Ultrasonic sensors can detect objects over a wide range of distances, from a few centimeters to several meters. This makes them suitable for a variety of applications.
- **Durability:** Ultrasonic sensors are relatively durable and can withstand the bumps and knocks that are common when using a walking stick.
- **Low cost:** Ultrasonic sensors are relatively inexpensive, making them a cost-effective option for smart sticks. [30]

## Raspberry pi4

The Raspberry Pi 4 is a powerful and versatile single-board computer that can be used for a variety of projects, including smart sticks. Smart sticks are devices that can be used to help people with disabilities, such as blindness or low vision, navigate their surroundings. They typically have a camera and sensors that can be used to detect obstacles and objects in the user's environment. The Raspberry Pi 4 can be used to power these sensors and process the data they collect. It can also be used to display information to the user, such as through a built-in display or a connected smartphone or tablet. [31]



*Figure 44: Raspberry Pi4*

Benefits of using Raspberry Pi to create a smart stick for visually impaired people:

- **Cost-effectiveness:** Raspberry Pi is a very affordable computer, which means that the cost of creating a smart stick is relatively low.
- **Versatility:** Raspberry Pi can be used to connect to a variety of sensors, which gives the smart stick the ability to detect a wide range of obstacles and objects.
- **Open-source:** Raspberry Pi is an open-source platform, which means that there is a large community of developers who can provide support and help with development.
- **Ease of use:** Raspberry Pi is relatively easy to use, even for people with no prior programming experience. [31]

## **Chapter 6: Conclusion**

### **6.1 Conclusion**

The integration of the smart stick and mobile app represents a significant milestone in empowering individuals to navigate the world with confidence and independence. By combining advanced features such as object detection, description, and alarm systems, the smart stick offers invaluable support to visually impaired individuals, enabling them to navigate their surroundings more safely.

The seamless integration with the mobile app further enhances the user experience by providing comprehensive navigation assistance and real-time guidance. Users can effortlessly explore unfamiliar places with ease, knowing that they have access to reliable and up-to-date information at their fingertips. The app's ability to track relatives also enhances personal safety, fostering a greater sense of security and peace of mind. This innovative system serves as a catalyst for promoting self-assurance, exploration, and inclusivity. By empowering individuals with visual impairments to embrace new experiences and confidently engage with the world around them, it opens a world of possibilities and opportunities. Ultimately, this system strives to create a more inclusive society where everyone can navigate and participate actively, regardless of visual abilities.



## 6.2 Future Work

Our project, the Smart Stick for visually impaired individuals, has already incorporated several features to improve independence and mobility. However, there is always room for further enhancements and additional features that can make the Smart Stick an even more comprehensive and effective solution. Here are some potential feature ideas for future work:

- **Enhanced Outdoor Navigation:** Utilize computer vision algorithms and sensors to detect traffic lights and provide audio cues for safe road crossing. Enable the Smart Stick to follow designated pedestrian paths for easier sidewalk navigation.
- **Indoor Navigation:** Implement technologies like Bluetooth beacons or Wi-Fi positioning systems for seamless navigation in complex indoor spaces such as malls, airports, and office buildings.
- **Real-time Environmental Feedback:** Equip the Smart Stick with sensors to provide real-time information on temperature, humidity, and moisture detection. This enables users to adapt to changing environmental conditions and make informed decisions.
- **AI Assistant:** Integrate artificial intelligence and computer vision algorithms to analyze the environment and offer detailed descriptions of objects and obstacles in real-time. Utilize deep learning models and OCR technology for object recognition and text-to-speech capabilities.

By incorporating these additional features into the Smart Stick, we can further empower visually impaired individuals, enhance their mobility and independence. The iterative development of these features will require continuous user feedback, testing to ensure the best possible outcome for our target audience.

## 6.3 References

- [1] C. Banupriya, M. S. Sumathi and D. P. F. Antony Selvi, "Multipurpose Guide for Visually Impaired People using Raspberry pi," *International Research Journal of Engineering and Technology (IRJET)*, vol. 07, no. 07, p. 6, 2020.
- [2] K. Bryant, "See," May 2018. [Online]. Available: <https://www.seeintl.org/blog/global-blindness-2050>. [Accessed 7 10 2022].
- [3] J. Elflein, "Number of blind people worldwide in 2020, by age and gender," Statista, 2021.
- [4] R. S, K. M, K. S and B. N, "RASPBERRY PI BASED SMART WALKING STICK FOR VISUALLY," *International Research Journal of Engineering and Technology (IRJET)*, vol. 09, no. 05, p. 7, 2022.
- [5] G. P. i. FEHU, "SMART STICK FOR BLIND PEOPLE," cairo, July 2021.
- [6] N. Sahoo, H.-W. Lin and Y.-H. Chang, "Design and Implementation of a Walking Stick Aid for Visually Challenged People," *sensors*, p. 17, 2 January 2019.
- [7] "Linux Explained: Distributions, Differences, Benefits, Security," zenarmor, [Online]. Available: <https://www.zenarmor.com/docs/linux-tutorials/what-is-linux>.
- [8] A. Garnett, "How to Choose a Linux Distribution," *DigitalOcean*, 16 6 2022.
- [9] "Operating System - Linux," tutorialspoint, [Online]. Available: [figmaresources.com](https://www.tutorialspoint.com/linux/).
- [10] [Online]. Available: <http://www.raspbian.org/>.
- [11] C. Hope, "Computer Hope," 7 6 2021. [Online]. Available: <https://www.computerhope.com/jargon/n/network.htm>.
- [12] NordLayer, "NordLayer," 23 5 2023. [Online]. Available: <https://nordlayer.com/blog/what-is-static-ip/>.
- [13] "About Node.js," [Online]. Available: <https://nodejs.org/en/about>.
- [14] I. V. Abba, "Node.js Server-Side JavaScript – What is Node Used For?," freecodecamp, 27 9 2022. [Online]. Available: <https://www.freecodecamp.org/news/node-js-server-side-javascript-what-is-node-used-for/>. [Accessed Node.js Server-Side JavaScript – What is Node Used For?].
- [15] "Socket.IO," [Online]. Available: <https://socket.io/docs/v4/>.
- [16] feduser, 14 10 2021. [Online]. Available: <https://www.geeksforgeeks.org/introduction-to-sockets-io-in-node-js/>.

- [17] G. Boesch, "Object Detection in 2022: The Definitive Guide," viso.ai, 2022. [Online]. Available: <https://viso.ai/deep-learning/object-detection/>.
- [18] A. Sharma, "Understanding a Real-Time Object Detection Network: You Only Look Once (YOLOv1)," Pyimagesearch, 11 4 2022. [Online]. Available: <https://pyimagesearch.com/2022/04/11/understanding-a-real-time-object-detection-network-you-only-look-once-yolov1/>.
- [19] D. X. G. L. W. W. C. S. a. W. O. Geng Zhan, "Scope Head for Accurate Localization," p. 17, 12 5 2020.
- [20] T. A. G. & T. J. Diwan, "Object detection using YOLO: challenges, architectural successors, datasets and applications," *Multimed Tools Appl*, p. 33, 02 August 2022.
- [21] S. Rath, "LearnOpenCV," 10 1 2023. [Online]. Available: <https://learnopencv.com/ultralytics-yolov8/>.
- [22] ultralytics, "ultralytics," [Online]. Available: <https://github.com/ultralytics/ultralytics>.
- [23] M. Krishnakumar, "A Gentle Introduction to YOLOv8," 4 5 2023.
- [24] F. Jacob Solawetz, "What is YOLOv8? The Ultimate Guide," roboflow, 11 1 2023. [Online]. Available: <https://blog.roboflow.com/whats-new-in-yolov8/>.
- [25] B. Lutkevich, "speech recognition," TechTarget , 9 2021. [Online]. Available: <https://www.techtarget.com/searchcustomerexperience/definition/speech-recognition>.
- [26] Twilio, 17 4 2023. [Online]. Available: <https://www.twilio.com/blog/what-is-speech-recognition>.
- [27] "Raspberry Pi Documentation," Raspberry Pi, [Online]. Available: <https://www.raspberrypi.com/documentation/accessories/camera.html>.
- [28] "Vibrator Motor Working and Applications," ELPROCUS, [Online]. Available: <https://www.elprocus.com/vibrator-motor-working-and-applications/>.
- [29] P. H. Hemane, A. Singh, S. Shree and S. Pandey, "RASPBERRY PI BASED SMART NAVIGATION SYSTEM FOR BLIND PEOPLE," *International Research Journal of Engineering and Technology (IRJET)*, vol. 07, no. 07, p. 4, 2019.
- [30] "What is Ultrasonic Sensor," TechaTronic, 8 1 2022. [Online]. Available: <https://techatronic.com/what-is-ultrasonic-sensor-ultrasonic-sensor-working-hc-sr-04/>.
- [31] R. Pi, "Raspberry Pi 4 Computer," 6 2019. [Online]. Available: <https://www.raspberrypi.org/>.
- [32] R. S. Ben Lutkevich, "TechTarget," 3 2023. [Online]. Available: <https://www.techtarget.com/searchenterpriseai/definition/GPT-3>.