Taibah University
College of Computer Science Engineering
Computer Engineering Department

Mosaned — Navigation and Guidance System for Indoor Environments for Blind and Visually Impaired Individuals

Mosaned — Navigation and Guidance System for Indoor Environments for Blind and Visually Impaired Individuals

A Project Submitted in partial fulfilment of the requirements for the Bachelor Degree in Computer Engineering

Submitted By

Mohammed Alharbi	**4200174**
Firas Alsadeq	**4200625**
Mousa Alenzi	**4106574**

Project Advisor
<Title><Moteb Alghamdi>

<2024-2025>-<1st Semester>

ABSTRACT

Visually impaired individuals face significant challenges when navigating indoor environments, particularly in unfamiliar settings. Indoor spaces are often unpredictable, filled with obstacles, and designed with critical visual information intended for sighted individuals. These challenges create a reliance on external assistance, such as asking for help from strangers or using companions, which can diminish the independence of visually impaired individuals.

This project aims to provide a solution through a wearable, easy-to-use system that guides and assists users by determining their position and detecting obstacles in their surroundings.

The proposed system integrates four core components: Obstacle Detection, Localization, Customizable Guidance, and a Management Dashboard. These components work together to enhance the navigation experience for visually impaired individuals. The Obstacle Detection System uses ultrasonic sensors to identify obstacles in the user's path and provides tactile feedback through vibration motors, enhancing spatial awareness. The Localization System uses QR codes as artificial landmarks to determine the user's position relative to the environment, providing accurate and reliable indoor localization. The Customizable Guidance System delivers tailored, location-specific instructions via the smartphone, helping users navigate effectively through designated areas. Additionally, the Management Dashboard allows building managers to easily integrate the system into their facilities by enabling them to generate, manage, and deploy QR codes within the building.

By integrating these components, the system ensures a practical and cost-effective solution for improving mobility, independence, and accessibility for visually impaired individuals, while also simplifying adoption for building managers.

- the result obtained
- the significance of the result or finding

KEYWORDS: Localization; Obstacles Avoidance; Objects Detection, Customization; Guidance; Pose; Calibration; Assistive Technology; Smartphone Application; Accessibility Application

ACKNOWLEDGEMENTS

The content of this single page is left to the preference of the student. It is suggested however that the page makes reference to guidance received by the student from his or her supervisor and project Committee members. Reference should also be made to any financial assistance received to carry out the project. Any extraordinary assistance received by the student for example in word processing, data collection, data analysis, and so on, should be properly acknowledged. The acknowledgements should not exceed 250 words.

Contents

List of Tables v

List of Figures vi

Glossary and List of Abbreviations vii

1 Introduction 1

1.1 Overview 2
1.2 Motivation 2
1.3 Problem Statement 3
1.4 Project Objectives 3
1.5 Project Outline 3

2 Background 5

2.1 Introduction 6
2.2 Landmarks (Natural & Artificial) 6
2.3 QR code 6
2.3.1 QR Structure 7
2.3.2 QR Code Versions and Types 8
2.3.3 QR Code Encoding 9
2.3.4 QR Code Decoding 10
2.4 Indoor Localization 10
2.4.1 Indoor Localization with QR Codes 11
2.5 Android Platform 18
2.6 TalkBack: Accessibility on Android 18
2.7 ESP32 19
2.7.1 ESP Series Overview 19
2.7.2 Key Features of ESP Modules 20
2.8 Ultrasonic Ranging Module 20
2.9 Coin Vibration Motor 21
2.10 Related Work 22

3 Methodology 25

3.1 System Architecture Overview 26
3.1.1 High-Level View of Components 26
3.1.2 Use Case Diagram 26

3.1.3 System Architecture Diagram 27
3.2 Database System 27
3.3 Building Management Dashboard 29
3.4 Mobile Application 30
3.5 Localization System 31
3.5.1 Camera Calibration 31
3.5.2 Camera's Relative Pose 33
3.6 Customizable Guidance 34
3.6.1 Location-Based Instructions and Information 34
3.7 Obstacle Detection 35
3.7.1 System Overview 35
3.7.2 Ultrasonic Sensor Data Collection 36
3.7.3 Data Processing and Mobile Application 36
3.7.4 Vibration Feedback Mechanism 37
3.8 Experiment Setup 38
3.8.1 Mobile Phone and Stand Configuration 38
3.8.2 QR Code Placement 39
3.8.3 Wearable Obstacle Detection System 39
3.8.4 Back-end Infrastructure Hosting 40
3.8.5 Power Supply and Connectivity 41
3.9 Test & Evaluation 42
3.9.1 Evaluation of Overall Experience 42
3.9.2 Evaluation of Localization System 42

4 Results and Discussion 44
4.1 Results 45

5 Conclusion and Future Work 46
5.1 Conclusion and Future Work 47

References 48

Appendices 52

A 52
A.1 The divided environment method 52
A.2 Metadata and Features of the Database 52

List of Tables

4.1 Results of Various Scenarios 45

List of Figures

2.1 These are three QR Codes that store texts. The texts' lengths get larger going from the left to the right. Notice that the alignment pattern only appears at the bigger QR Codes, and there are multiple ones at the biggest QR Code. 7
2.2 QR codes Versions (adapted from [21]). 8
2.3 QR code types (adapted from [21]). 9
2.4 Projection of 3D points into a 2D image (Adopted from [6]) . . 14
2.5 Simple illustration for the camera's projection matrix made by Aqeel Anwar [17].1 15
2.6 This is a picture of a QR code that has the width/height of 16.8cm and its four corners points in the image coordinate system in pixels are known as it appears. 16
2.7 The left image represents the calibration pattern's real world inner points locations. While on the other hand, the right image shows the estimated inner points locations. 17
2.8 ESP32-WROOM Development board 19
2.9 HC-SR04 Module 21
2.10 C1026B Series Coin Vibration Motor 21

3.1 Use Case Diagram of Mosaned System 27
3.2 System Architecture Diagram of Mosaned System 28
3.3 Entity-Relationship Diagram (ERD) of the Mosaned Database System 28
3.4 Schematics of Ultrasonic Sensors and Mobile Application Integration 36
3.5 Schematics of Vibration Motors and PWM Control 38
3.6 Mobile Phone Mounted on Adjustable Stand 39
3.7 QR Code Placement on Wall 40
3.8 Wearable Obstacle Detection System with Integrated Belt . . . 41

Glossary and List of Abbreviations

Glossary

Artificial Landmark Human-made structures used for localization, such as QR codes, which help in navigating indoor environments.

Bluetooth Low Energy (BLE) A low-power wireless technology used for proximity detection and navigation.

Camera Calibration The process of determining a camera's internal parameters to enhance accuracy in object detection and localization.

Indoor Localization Technique for identifying a person or object's location inside a building.

Pose Estimation Technique for calculating precise position and orientation of an object or user in a given space.

Quick Response (QR) Code A matrix barcode that stores information accessible via scanning; useful in localization and navigation.

List of Abbreviations

BLE Bluetooth Low Energy

GPS Global Positioning System

MCU Microcontroller Unit

QR Quick Response

Chapter 1

Introduction

1.1 Overview

Navigating indoor environments poses significant challenges for individuals with visual impairments. These challenges stem not only from the absence of effective GPS signals indoors but also from the complexity of navigating unfamiliar spaces, which often lack accessible cues for orientation. Retail malls, hospitals, educational institutions, and other intricate spaces present obstacles that can impede mobility and independence for visually impaired individuals.

This project goes beyond solving the problem of indoor localization. It introduces a comprehensive navigation and guidance system that addresses multiple facets of the user experience. By leveraging QR codes as artificial landmarks, the system provides precise localization, enabling users to determine their position within an environment. Additionally, the system incorporates obstacle detection and real-time feedback through tactile and auditory alerts, enhancing spatial awareness and safety.

The proposed solution also offers a customizable guidance experience, allowing users to access location-specific instructions and tailored navigation paths. Furthermore, a user-friendly dashboard empowers building managers to integrate the system seamlessly into their facilities by managing and deploying QR codes efficiently. This end-to-end approach ensures that visually impaired individuals can navigate indoor spaces with confidence, independence, and ease.

By combining advanced localization, obstacle detection, and intuitive guidance, the system aims to deliver an accessible, and cost-effective solution that transforms the way visually impaired individuals navigate in unfamiliar building, as well as interacting with their surroundings.

1.2 Motivation

This project is motivated by the need to enhance the quality of life and autonomy for visually impaired individuals, especially in interior settings where conventional navigation aids such as GPS are inadequate. Visually impaired individuals encounter significant obstacles when navigating intricate indoor environments, which can restrict their mobility and independence. Existing options, such as guide dogs and human aid, although beneficial, may prove inadequate or unworkable in all situations. The motivation also is to develop a prototype solution utilising commonly accessible technology (such as QR codes and mobile devices) enhances the system's accessibility to a broader audience (Cost and Accessibility). This differs from expensive or complex systems that may require specialized hardware.

1.3 Problem Statement

Navigating indoor environments is challenging for visually impaired individuals, directly affecting their independence and daily activities. While some existing technologies have developed separate algorithms for indoor localization, they are often not integrated into a user-friendly device that individuals can easily use. Moreover, many solutions rely on expensive sensors, increasing the overall cost of the device. Additionally, the majority of current aids do not provide a cohesive approach that combines accurate localization with obstacle detection, and they often lack context-specific, non-visual instructions tailored to the unique needs of visually impaired users. This highlights the urgent need for an affordable, integrated indoor navigation system that enhances mobility and independence in everyday settings.

1.4 Project Objectives

The primary objectives of this project are as follows:

- To develop an intuitive and accessible mobile application that enables visually impaired individuals to navigate unfamiliar indoor environments with confidence and ease.
- To enhance the independence of visually impaired users by providing a reliable system for localization, obstacle detection, and customizable guidance.
- To increase spatial awareness for visually impaired users by offering real-time feedback on their surroundings using accessible technologies like vibration motors and voice guidance.
- To design a user-friendly dashboard for building managers, facilitating the integration of the system into buildings by allowing them to manage, configure, and export QR codes effortlessly.
- To ensure the system is cost-effective and leverages widely available technologies, making it accessible to a broader audience and promoting widespread adoption.

1.5 Project Outline

The remainder of this report is organised as follows: Chapter 2 provides the Background and the related works. Chapter 3 provides the project system design, and Chapter 4 covers the implementation. Chapter 5 provides the results obtained by the system, and Chapter 6 concludes the report.[1]

[1]Next Term. But we can mention it from now.

Chapter 2

Background

2.1 Introduction

This chapter is organized as follows: Section 2 discusses the role of landmarks in navigation, emphasizing artificial landmarks such as QR codes. Section 3 explores the structure, versions, and encoding of QR codes, while Section 4 focuses on QR code-based localization systems. Section 5 covers object detection, followed by Section 6, which addresses customizable guidance systems. Finally, Section 7 reviews related work in the field.

2.2 Landmarks (Natural & Artificial)

Landmarks can be categorized as either natural or artificial. Natural landmarks are formed by nature, such as mountains, rocks, trees, or any other natural formations. While on the other hand, artificial landmarks are human-made structures such as traffic signs, statues, and QR codes. both categorizes can be used in various important fields in computer science. For example, computer vision algorithms leverage landmarks by detecting, identifying, and tracking them to determine the precise location of an object, create a 3D map of the surrounding environment, or to monitor the state of an object.

While both categorizes can be very useful, natural landmarks are much harder to recognize, very diversified and do not have uniform shapes, difficult to customize, and they do not encode data. All of these characteristics can be opposite in the artificial landmarks, which make them better in terms of control, precision, reliability, and adaptability. Also there are a lot of libraries that support encoding, detecting, decoding, and tracking artificial landmarks. There are also a wide variety of artificial markers such as QR, ArUco, Topotag. More information on the other kinds of markers are present on reference [1].

2.3 QR code

Quick Response (QR) codes are a type of matrix barcode designed for efficient data encoding and rapid scanning. QR Codes first developed in 1994 by Denso Wave, to track automotive parts in manufacturing. but have since become widely used across various applications due to their versatility, simplicity, and capacity to store significant amounts of information compared to traditional barcodes.

2.3.1 QR Structure

QR Codes can be detected and decoded at various angles and distortion levels due to their unique structure. The code consists of black modules (squares) arranged on a white background in a grid pattern, allowing for rapid and error-resistant scanning. All QR Codes have standard structure as shown in Figure 2.1. This structure is made out of the following parts:

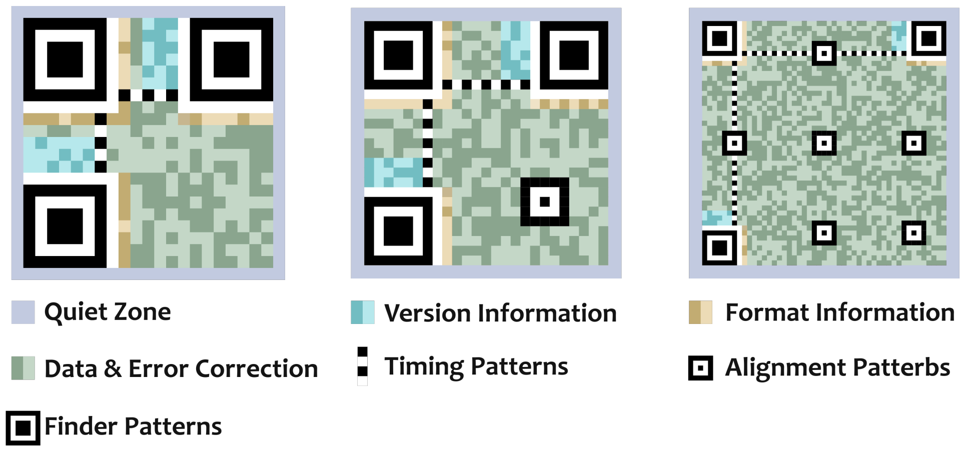

Figure 2.1: These are three QR Codes that store texts. The texts' lengths get larger going from the left to the right. Notice that the alignment pattern only appears at the bigger QR Codes, and there are multiple ones at the biggest QR Code.

- **Quiet Zone:** This is a white empty area surrounds the QR Code that helps distinguishing it from its surroundings.
- **Version Information:** QR Codes have different versions/sizes which specified by these two areas.
- **Format Information:** This part provides details about the error correction level and data mask pattern used.
- **Data & Error Correction:** This is where both encoded data & error correction are. data and error correction are stored together enabling the QR code to recover and reconstruct the stored data, even if up to 30% of the code is damaged or obscured. Data have different types, such as text, URLs, or other.
- **Timing Patterns:** This part is essential for defining the grid's structure and assists the scanner in establishing the size and coordinate system of the code.

- **Alignment Patterns:** The Alignment Patterns help correct distortion and skewing of the QR code when viewed from different angles. It is especially crucial for larger QR codes that may be prone to bending or misalignment.

- **Finder Patterns:** These patterns assist the scanning device in rapidly locating and orienting the QR code, regardless of its rotation or angle.

See [21], for more information.

2.3.2 QR Code Versions and Types

QR codes come in 40 versions, each representing a different size and data capacity. Version 1 contains 21×21 modules, while Version 40 has 177×177 modules. As the version number increases, so does the data capacity and the complexity of the code, making it capable of storing more information or supporting higher levels of error correction [21].

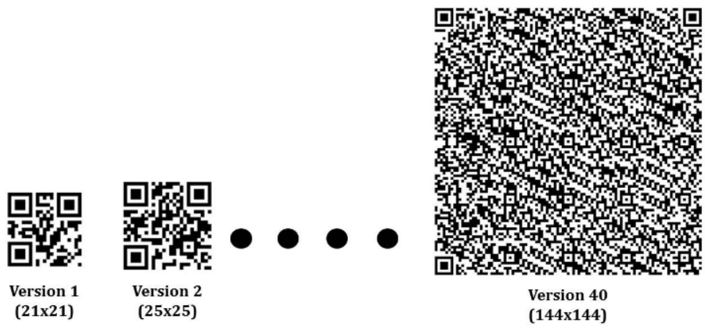

Figure 2.2: QR codes Versions (adapted from [21]).

There are also specialized QR code types designed for specific applications:

- **QR Code Model 1 and 2**: Model 1 is the original version of the QR code, developed in 1994 by Denso Wave. While model 2 is an improved version of model 1, and it is the one most commonly used today. These models are used for daily basis.

- **Micro QR Code**: Designed to be smaller and simpler, it uses only one Finder Pattern, making it more compact than traditional QR codes. It is often used when space is limited [21].

- **Logo QR Code**: Allows logos or images to be embedded within the QR code, enhancing the code's visual appeal for marketing or branding purposes [21].

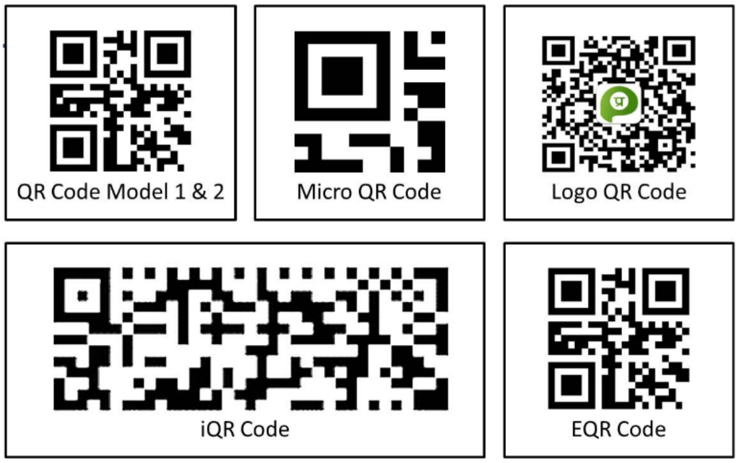

Figure 2.3: QR code types (adapted from [21]).

- **iQR Code**: A flexible matrix-type QR code capable of being printed in various sizes and configurations, from small, high-capacity codes to large codes. It can store more data than standard QR codes and can be inverted or turned into dot patterns for direct part marking [21].
- **Encrypted QR Code**: Uses encryption techniques to secure the information encoded within, making it suitable for applications requiring data confidentiality [21].

2.3.3 QR Code Encoding

The encoding process for a QR code involves converting data into a matrix of black and white modules. First, the input data is analyzed to determine the appropriate encoding mode, such as numeric, alphanumeric, byte, or Kanji. The data is then transformed into binary code, and error correction is added using Reed-Solomon algorithms, ensuring that the code can still be scanned if partially damaged [21].

Popular online tools for generating QR codes include:

- **Online Tools**: These free online tools offer simple and fast solutions for both generating and decoding QR codes. QRickit allows users to decode QR codes from uploaded images, while GOQR.me provides various output formats, such as PNG, SVG, and EPS, making it versatile for different use cases [23][24].

For generating QR codes programmatically, popular libraries include:

- **Python - PyQRCode**: A Python library that simplifies QR code creation, allowing output in SVG, PNG, and other formats [25].

- **Java - ZXing (Zebra Crossing)**: An open-source library widely used for encoding QR codes in Java and Android applications [26].
- **iOS - Core Image**: iOS provides native QR code generation capabilities via the 'CIQRCodeGenerator' filter in the Core Image framework [27].

2.3.4 QR Code Decoding

Decoding a QR code starts by scanning the Finder Patterns, which allow the scanner to properly align the code. Next, the Format Information is read to apply the correct error correction and mask pattern. Once decoded, the binary data is translated back into the original format (e.g., text, URL) [21].

Common online tools for decoding QR codes include:

- **Online Tools**: These tools offer versatile QR code decoding solutions across platforms. ZXing is an open-source decoder integrated into Android and web applications, QRickit provides online QR code decoding from uploaded images, and Google Lens allows users to scan and decode QR codes directly via smartphone cameras [26, 31].

For decoding programmatically, useful libraries include:

- **Python - Segno**: A versatile Python library for generating and reading both QR and Micro QR codes [28].
- **Java - ZXing**: Provides decoding functionality along with encoding, and is widely used for mobile and web apps [26].
- **iOS - AVFoundation**: iOS provides native support for scanning QR codes using the 'AVCaptureMetadataOutput' class in the AVFoundation framework [29].
- **Google ML Kit**: temporarily empty...

2.4 Indoor Localization

Indoor localization refers to the process of determining the precise position and orientation of an object or individual within indoor environments, where traditional Global Positioning System (GPS) signals are often unreliable. This technology is critical for a wide range of applications, including navigation within large buildings and asset tracking in warehouses.

Various technologies are employed for indoor localization, each with distinct advantages and limitations. Wi-Fi, Bluetooth, Radio-Frequency Identification (RFID), and Ultra-Wideband (UWB) are among the most widely used methods. Some systems rely on complex and expensive hardware, such as sensors and Inertial Measurement Units (IMUs), while others utilize more cost-effective solutions, including Bluetooth beacons and pose estimation using Quick Response (QR) codes. The selection of the appropriate technology depends on the specific requirements of the application, taking into account factors such as accuracy, cost, and ease of implementation [22].

As the field of indoor localization continues to evolve, new techniques and innovative approaches are emerging, such as the use of QR codes for precise positioning and tracking.

2.4.1 Indoor Localization with QR Codes

Indoor localization using QR codes enables position determination by detecting QR codes strategically placed throughout an indoor space. These codes can be affixed to floors, walls, ceilings, or suspended on hanging panels, each containing encoded positional information.

2.4.1.1 QR Code Placement

The placement of QR codes plays a critical role in ensuring effective indoor localization. Codes can be positioned in various locations:

- **Ceilings**: QR codes on ceilings are often out of the way and can provide an unobstructed view for overhead cameras, such as those mounted on hats or handheld devices. This placement is ideal for applications where the user's line of sight remains upward.

- **Walls**: QR codes can also be positioned on walls at different heights to accommodate various camera angles. This setup is particularly useful when the user or device is at eye level with the code.

- **Floors**: Placing QR codes on floors can be beneficial in environments where overhead cameras or downward-facing sensors (such as those on a robot or mobility aid) are used. However, codes on floors might be more prone to wear and tear and may require periodic maintenance.

- **Hanging Panels**: In environments where flexibility is needed, QR codes can be placed on hanging panels suspended from the ceiling. This allows for better visibility while keeping the codes elevated from foot traffic or other obstacles.

A standardized placement of QR codes across buildings is essential to ensure consistency and usability for visually impaired individuals. If each building adopts a different placement strategy, users may face difficulties adjusting their camera orientation for detection, leading to inefficiencies in navigation. To address this, it's recommended to standardizing QR code placement on walls and hanging panels at a uniform height, which offers the most practical balance between accessibility and ease of detection.

2.4.1.2 Data Encoding in QR Codes

Each QR code encodes essential information for localization purposes, including:

- **Coordinates**: The most important data that QR codes can encode are the precise coordinates of their position within the environment. These coordinates are predefined and allow the system to accurately calculate the user's or object's location when the QR code is detected. The coordinates might be in terms of x, y, z positions or based on a grid system specific to the environment.

- **Unique Identifier (ID)**: In addition to coordinates, each QR code will have a unique ID that differentiates it from others in the system. The ID can be referenced in the system's database to retrieve additional information, such as the room name or floor level.

- **Orientation Data**: QR codes can also encode information about their orientation, which helps in determining the user's orientation in relation to the environment (e.g., the angle of the code in reference to a global axis).

- **Additional Metadata**: If needed, QR codes can encode further metadata, such as room names, nearby landmarks, or points of interest. This is especially useful in environments where additional contextual information enhances the user's navigation experience.

2.4.1.3 Approaches to Localization with QR Codes

Several methods exist for implementing indoor localization using QR codes, each with its own benefits and trade-offs. Two common approaches include the divided environment method, which offers simplicity and low computational requirements, and the pose estimation method, which provides higher precision but at the cost of increased computational complexity. These approaches are discussed below.

The Divided Environment Method In the divided environment method, the space is partitioned into distinct pieces(e.g., squares, triangles, or hexagons), with each piece containing a QR code encoding its position. QR codes can be placed on the floor, ceiling, walls, or hanging panels depending on the system's configuration. A similar approach has been demonstrated by Zhang et al. [7].

To illustrate this method, consider a room measuring 4 meters in both width and height, divided into 16 equal squares, each with an area of 1m^2. If we customized a hat for example, that embeds a camera in its top, if we put the QR codes at the ceiling, then the user's position will get determined while navigating in the room wearing the hat.

Although this method is computationally efficient, it has limitations. The position values are discrete, which may not provide the continuous and precise localization required in certain applications. Nonetheless, this method can be highly effective for specific use cases, as discussed in Appendix A.

Pose Estimation Method Another important and interesting way to calculate the precise and continues values of the user's pose(position & orientation) is by detecting and decoding a QR Code using a camera to gain its global pose, then calculate its relative pose to the camera. After that, we will be able to use these two poses to estimate the user's precise pose as follows:

$$user\_global\_pose = QR\_global\_pose - QR\_relative\_pose$$

See [11], where they used this method(but different implementation) for their localization system.

But calculating the relative pose is not a straight forward process. First, let us explain how cameras work. The basic idea of cameras is to project the real world 3D points into a 2D image, see Figure 2.4 for illustration. The following equation describes the relation between the real world points and the image points:

$$x = PX \tag{2.1}$$

Where:

- $\mathbf{x} = [\mathbf{u},\ \mathbf{v},\ \mathbf{1}]^T$ is the homogeneous coordinate of the 2D point in the image plane.

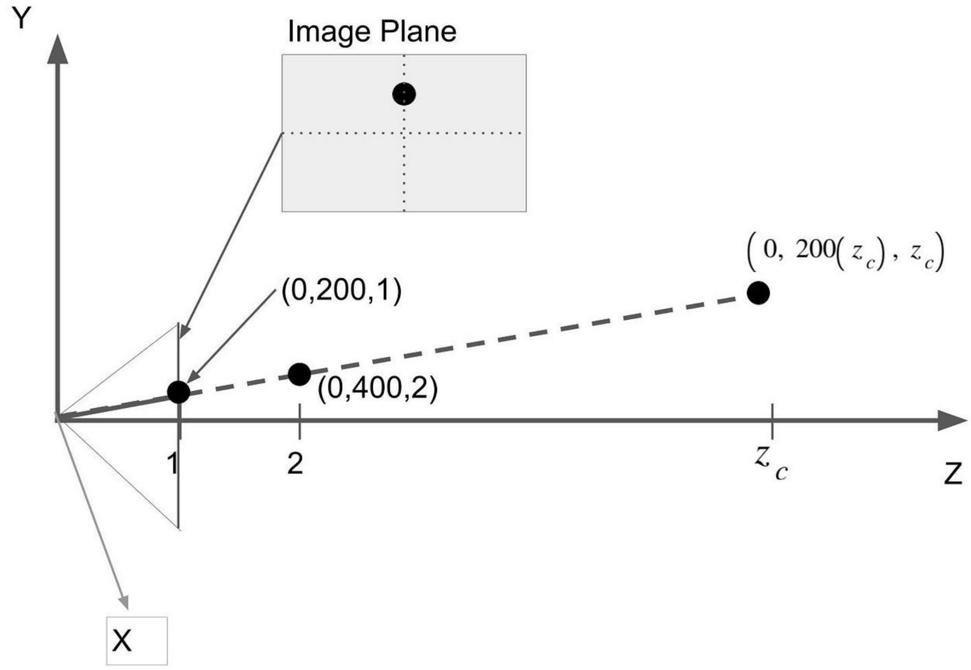

Figure 2.4: Projection of 3D points into a 2D image (Adopted from [6])

- $\mathbf{X} = [\mathbf{x}_w, \mathbf{y}_w, \mathbf{z}_w, \mathbf{1}]^T$ is the homogeneous coordinate of the 3D point in the world coordinate system.
- $\mathbf{P} = \mathbf{K}[R|t]$ is the camera projection matrix.

  With:

  – $[R|t]$ is the camera's extrinsic matrix, which "is a transformation matrix from the world coordinate system to the camera coordinate system" - Aqeel Anwar [17].

    R is a 3x3 matrix that represents the rotation of the object's with respect to the camera. t is 3x1 vector describes the object's position relative to the camera.

$$[R|t] = \begin{bmatrix} R_{3\times3} & t_{3\times1} \end{bmatrix}_{(3\times4)} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}$$

  – **K** is the camera's intrinsic matrix, which "is a transformation matrix that converts points from the camera coordinate system to the pixel coordinate system" - Aqeel Anwar [17]. It contains camera's focal length(f_x, f_y) and the principal point(c_x, c_y) as follows:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Notice how the camera's intrinsic matrix cares only about the camera's internal parameters, while the extrinsic matrix does not.

Now we can rewrite equation 2.1 as follows:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{2.2}$$

where s is a scale factor due to the homogeneous coordinates.

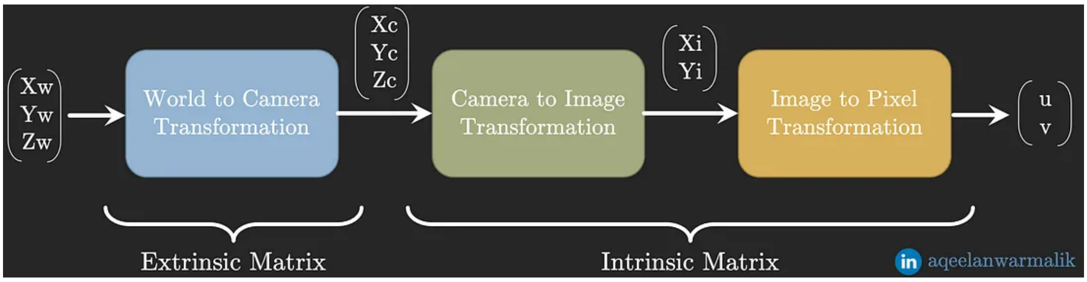

Figure 2.5: Simple illustration for the camera's projection matrix made by Aqeel Anwar [17].1

After we understood the basics of how cameras project 3D world points into 2D images, we can use this knowledge to calculate the pose of an object, such as a QR code. At least four 3D points at the object need to be known, such as the four corners of a QR code in the world coordinate system. Then, the corresponding 2D projections in the image plane of these points need to be known. See figure 2.6 for illustration.

For more illustration, let us take the QR code at figure 2.6 as an example. The QR code's width/height is equal to 16.8cm in the real world coordinate system. Using this value, we will be able to extract the four corners 3D points in the real world coordinate system as follows:

$$\begin{gathered} top-left-point = (-16.8/2, 16.8/2, 0) = (-8.4, 8.4, 0) \\ top-right-point = (8.4, 8.4, 0) \\ bottom-right-point = (-8.4, 8.4, 0) \\ bottom-left-point = (-8.4, -8.4, 0) \end{gathered} \tag{2.3}$$

For the sake of simplicity, the values at 2.3 are for a QR code with a global position value of (0,0,0).

Figure 2.6: This is a picture of a QR code that has the width/height of 16.8cm and its four corners points in the image coordinate system in pixels are known as it appears.

After choosing some pairs of 3D points with their corresponding 2D projections, use equation 2.2 for each pair as follows:

$$s \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x_{w,i} \\ y_{w,i} \\ z_{w,i} \\ 1 \end{bmatrix}$$

This yields two equations per point:

$$u_i = \frac{f_x(r_{11}x_{w,i} + r_{12}y_{w,i} + r_{13}z_{w,i} + t_x)}{r_{31}x_{w,i} + r_{32}y_{w,i} + r_{33}z_{w,i} + t_z} + c_x$$

$$v_i = \frac{f_y(r_{21}x_{w,i} + r_{22}y_{w,i} + r_{23}z_{w,i} + t_y)}{r_{31}x_{w,i} + r_{32}y_{w,i} + r_{33}z_{w,i} + t_z} + c_y$$

Finally, use the later equations to solve for R and t by using one of the PnP(Perspective-n-Point) algorithms. At the end, we will have the values of R and t which is the camera's extrinsic matrix that describes an object's rotation and translation relative to the camera. So, this is how to get the relative pose.

PnP PnP is a classic approach in computer vision for determining the pose of a calibrated camera with respect to a set of known 3D points in the real world coordinate system and their corresponding 2D projection points. So in a nutshell a PnP algorithm takes three inputs: 3D points, 2D projection points, and intrinsic matrix to solve the camera's pose. There are several different PnP algorithms such as Direct Linear Transformation(DLT), Efficient PnP(EPnP), and RANSAC-PnP.

Camera Calibration Camera Calibration is the process of calculating the camera's focal length, principal point, and distortion coefficients. The basic idea behind camera calibration is to take multiple photos by the camera to a known pattern, then try to accurately locate some points at these photos. Then compare these estimated points to their physical locations of points in the calibration pattern (usually on a flat surface) in the real-world coordinate system. By comparing the 2D points in the images to their known 3D locations in the real-world coordinate system, we will be able to calculate accurate values of the camera's focal length, principal point, and distortion coefficients.

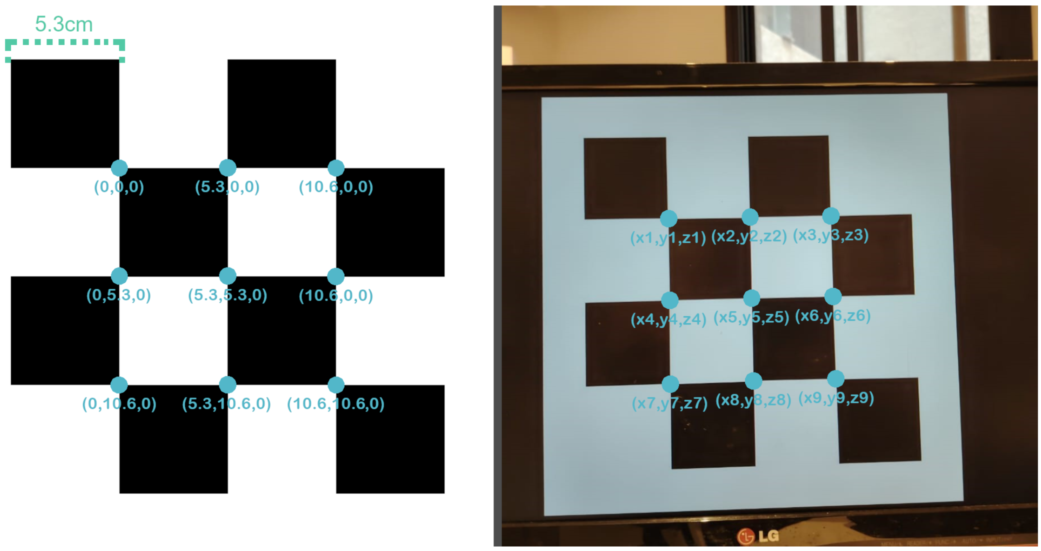

Figure 2.7: The left image represents the calibration pattern's real world inner points locations. While on the other hand, the right image shows the estimated inner points locations.

Libraries:

- **Camera Libraries**: There are several libraries for cameras in Android devices, such as Camera2, and CameraX. It is recommended for most developers to use CameraX library, since it supports the vast majority of Android devices, and provides a simple and easy to use API - unlike Camera2 - that spares users from dealing with compatibility issues and other low level details. See [20] for more info.

- **Computer Vision Libraries**: temporarily empty...

While the pose estimation approach offers significantly higher accuracy, it requires greater computational resources due to additional processes, such as camera calibration to determine intrinsic parameters. This increased complexity makes the method more resource-intensive compared to the divided environment approach. However, it is suitable for applications where continuous and precise localization is essential.

2.5 Android Platform

Android is a versatile, open-source software platform that powers a wide range of devices and applications. Its robust design and layered architecture make it ideal for building scalable and user-friendly solutions. At its core, Android provides essential functions like memory management, hardware control, and system-level services, all of which ensure seamless operation across diverse devices.

The platform is designed to be modular, allowing developers to easily integrate advanced features and functionalities. For example, it offers interfaces that enable interaction with device hardware, such as cameras or Bluetooth modules, without requiring in-depth knowledge of the underlying systems. Additionally, Android's framework simplifies the development process by providing reusable tools and components for creating intuitive user interfaces, managing notifications, and handling background tasks.

The architecture's flexibility and widespread support have established Android as a leading platform for mobile development, providing the foundation for creating powerful, user-centered applications [3].

2.6 TalkBack: Accessibility on Android

TalkBack is Google's built-in screen reader for Android devices, designed to enable eyes-free interaction and improve accessibility for visually impaired users. Its key features include:

- **Gesture-Based Navigation**: Multi-finger gestures (for Android R and above) enable easy navigation through menus and interfaces.
- **Voice Feedback**: Reads on-screen content aloud to guide users through apps and system interactions.
- **Braille Support**: Includes a TalkBack braille keyboard for text input without additional hardware.

- **Tutorials and Help**: Offers an in-app tutorial to introduce new users to gestures and functionality, ensuring an accessible learning experience.

TalkBack enhances accessibility, allowing users to perform tasks such as browsing the web, sending messages, and using apps without relying on visual input. [5]

2.7 ESP32

Embedded System Processors (ESP) are versatile microcontroller units (MCUs) widely used in Internet of Things (IoT) and embedded applications. Developed by Espressif Systems, these modules combine powerful processing with Wi-Fi and Bluetooth connectivity, making them ideal for smart home devices, wearables, and industrial automation.

Figure 2.8: ESP32-WROOM Development board

2.7.1 ESP Series Overview

The ESP series includes different models tailored to specific application needs for performance, power efficiency, and connectivity. Key types include:

- **ESP8266 Series:** A cost-effective option with a 32-bit LX6 single-core processor (up to 160 MHz), supporting 2.4 GHz Wi-Fi and peripheral interfaces like UART, I2C, and PWM. Its low power consumption suits battery-powered applications [4].

- **ESP32 Series:** Offers more power and connectivity. The ESP32-S2 has a single-core LX7 processor, while the ESP32-S3 features a dual-core LX7 processor (up to 240 MHz), with 2.4 GHz Wi-Fi and Bluetooth 5.0. It includes security features like secure boot and flash encryption [4].

- **ESP32-S2 Series:** A low-power option with a single-core LX7 processor (up to 240 MHz) and Wi-Fi support. It includes USB OTG and advanced security features, making it suitable for secure IoT applications [4].
- **ESP32-S3 Series:** Designed for AI and neural network applications, it features a dual-core LX7 processor (up to 240 MHz), with Wi-Fi and Bluetooth 5 (LE). It has added vector instructions for AI tasks and comprehensive security options [4].

2.7.2 Key Features of ESP Modules

ESP modules provide a range of features for various applications:

- **Wireless Connectivity:** All modules support 2.4 GHz Wi-Fi; some also include Bluetooth 5.0 (LE) [4].
- **Peripherals and Interfaces:** Offers GPIO, SPI, I2C, I2S, UART, PWM, ADC, DAC, and USB OTG (on select models), supporting a wide variety of sensors and devices [4].
- **Power Efficiency:** Low-power modes and fine-grained control make ESP modules suitable for battery-powered devices, with the ESP32-S2 optimized for ultra-low-power applications [4].

With their range of types, processors, and features, ESP modules offer a flexible solution for IoT and embedded systems, meeting the needs of modern, connected applications [4].

2.8 Ultrasonic Ranging Module

The HC-SR04 ultrasonic ranging module is a widely utilized sensor for non-contact distance measurement. Its capability to measure distances ranging from 2 cm to 400 cm, with an accuracy of up to 3 mm, makes it a versatile tool for various applications in robotics, automation, and IoT systems. This module integrates ultrasonic transmitters, a receiver, and control circuitry within a compact form factor [13].

Working Principle The HC-SR04 operates by emitting an ultrasonic pulse at 40 kHz through its transmitter. This pulse propagates through the air and, upon encountering an obstacle, reflects back toward the sensor. The module measures the time taken for the echo to return to the receiver. The distance to the object can be calculated using the formula:

Figure 2.9: HC-SR04 Module

$$\text{Distance} = \frac{\text{Time taken by pulse (s)} \times \text{Speed of sound (340 m/s)}}{2}$$

To ensure accurate results, a measurement cycle of over 60 ms is recommended to avoid interference between consecutive signals.

The HC-SR04 provides an effective and economical solution for non-contact distance measurement, with applications in diverse systems requiring proximity sensing [13].

2.9 Coin Vibration Motor

The C1026B Series coin vibration motor is a compact, permanent-magnetic DC motor designed for generating tactile feedback through vibrations. Its small size and high efficiency make it ideal for applications in wearable devices, handheld electronics, and consumer gadgets.

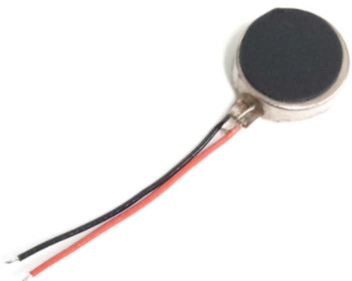

Figure 2.10: C1026B Series Coin Vibration Motor

Operating Conditions The motor operates effectively within a voltage range of 2.7 V to 3.3 V DC, with a rated voltage of 3.0 V DC. It can function in environmental conditions ranging from -20°C to 60°C with ordinary humidity, making it robust for various settings.

Working Principle The vibration is generated by the rapid rotation of an eccentric mass attached to the motor shaft. When powered, the imbalance caused by this mass induces vibrations, which are transmitted to the device housing the motor. This principle is commonly used in applications requiring tactile feedback or silent alerts.

Performance Characteristics The motor achieves a rated speed of 9,000 RPM at its rated voltage while consuming a maximum current of 90 mA. It is designed to start at a voltage as low as 2.3 V DC. With a mechanical noise limit of 50 dB(A), it ensures quiet operation.

The C1026B Series motor provides a reliable and efficient solution for haptic feedback and vibration alert mechanisms in compact electronic devices [12].

2.10 Related Work

Navigation solutions for visually impaired individuals utilize a range of technologies, including Bluetooth Low Energy (BLE), LiDAR, and artificial landmarks. Each of these technologies offers unique advantages and limitations that impact their effectiveness in real-world applications. In this section, we explore different solutions that incorporate various technological approaches, highlighting how each addresses the challenges faced by visually impaired users in navigating indoor environments.

PathFinder, proposed by Kuribayashi et al. [18], is a map-less navigation system developed to assist blind individuals in navigating unfamiliar indoor environments. Unlike other systems that rely on prebuilt maps, PathFinder uses real-time detection of intersections and signs to guide users. The system includes a suitcase-shaped robot that blind users control through a handle interface, receiving audio feedback about their surroundings. PathFinder's core navigation capabilities are powered by LiDAR for detecting intersections and image processing via a high-resolution camera to recognize directional and textual signs. Developed using a participatory design approach with blind users, the system focuses on addressing the most relevant challenges of navigating unknown spaces. In a user study involving seven blind participants, PathFinder demonstrated significant effectiveness in helping users navigate unfamiliar environments with greater confidence than when using traditional aids like canes or guide dogs. Despite requiring more user effort than map-based systems, participants appreciated PathFinder's flexibility and adaptability to environments without prior mapping. The study concluded that PathFinder offers a valuable solution for blind individuals, particularly in situations where prebuilt maps are unavailable.

Ahmetovic et al. [8] presents NavCog, NavCog is a smartphone-based navigation system designed to help visually impaired individuals with turn-by-turn guidance in indoor and outdoor environments. The system uses Bluetooth Low Energy (BLE) beacons for accurate localization, employing a K-nearest neighbor (KNN) algorithm to compare current signal strengths with previously recorded beacon signal fingerprints. NavCog provides real-time auditory navigation instructions to guide users along a predetermined path and also notifies them of nearby points of interest (POI) and accessibility issues such as stairs or obstacles.

One of NavCog's key features is its ability to be easily deployed in large, complex environments without requiring significant infrastructure modifications. In a study conducted with six visually impaired participants on a university campus, NavCog successfully guided users through both familiar and unfamiliar spaces, receiving positive feedback on its navigation features. The study emphasizes NavCog's flexibility and ease of use, making it a promising tool for visually impaired individuals navigating new environments.

The work by Fraga et al. [19] presents an indoor navigation system has been developed for visually impaired individuals. This system uses QR code markers and computer vision techniques to provide real-time navigation instructions. Users scan QR codes placed in different locations to receive guidance along optimal paths. The system cross-references the scanned information with a prebuilt database. Audio feedback informs users of their current location and the next steps in the navigation process. If the user deviates from the planned route, the system recalculates the path. Importantly, this system does not require internet connectivity, making it practical for offline use. Additionally, the system includes a collision avoidance feature based on a monocular depth estimation algorithm, which predicts distances to obstacles using 2D images. Experimental results in a controlled environment demonstrated that the system accurately guided users and effectively detected obstacles with acceptable depth estimation margins. This study highlights the potential of QR code-based navigation for enhancing mobility for visually impaired individuals and suggests future integration into smartphones or AI accelerators for improved usability.

In conclusion, our solution distinguishes itself by employing QR codes as a cost-effective navigation tool that requires only a standard smartphone camera. This straightforward approach minimizes the need for additional hardware, thereby increasing accessibility for users. Unlike systems such as PathFinder and NavCog, which depend on specialized technologies like LiDAR and Bluetooth beacons, our method aims to simplify navigation by leveraging existing smartphone capabilities. Furthermore, we intend to integrate optional remote or Wi-Fi cameras to enhance the navigation experience without imposing significant costs. Our primary objective is to deliver a practical and user-friendly solution specifically tailored for visually impaired individuals, ensuring ease of use in real-world settings through thorough testing in various indoor environments.

Chapter 3

Methodology

3.1 System Architecture Overview

This section provides an overview of the architecture of our indoor navigation system, designed to assist visually impaired users with real-time, context-aware guidance within indoor environments. The system leverages mobile technology, backend services, and sensory input devices to deliver seamless and accessible navigation. Below, we briefly introduce the key components and their roles, supported by a use case diagram and system architecture illustration.

3.1.1 High-Level View of Components

The system comprises four main components, each explained briefly below. Detailed descriptions of each component are provided in subsequent sections.

- **Mobile Application**: The mobile application acts as the central interface for users and the processing unit for the system. It facilitates QR code scanning, provides navigation instructions via TalkBack, processes obstacle detection data, and communicates with the backend server to retrieve location-specific data.

- **Localization System**: The localization system determines the user's global position and orientation within the environment. It uses QR code scanning and positional data retrieval from the backend to calculate the user's current position relative to the environment.

- **Guidance System**: The guidance system provides real-time navigation assistance by retrieving and delivering location-based instructions. It allows users to select destinations and offers step-by-step guidance through TalkBack, dynamically updating as the user progresses along the path.

- **Obstacle Detection System**: The obstacle detection system ensures safety by alerting users to nearby obstacles. Using four ultrasonic sensors and vibration motors, it delivers tactile feedback proportional to the proximity of detected obstacles, enhancing the user's awareness of their surroundings.

3.1.2 Use Case Diagram

The following use case diagram (see Figure 3.1) illustrates the interactions between the user and the system components, including QR code scanning, destination selection, navigation guidance, and obstacle detection:

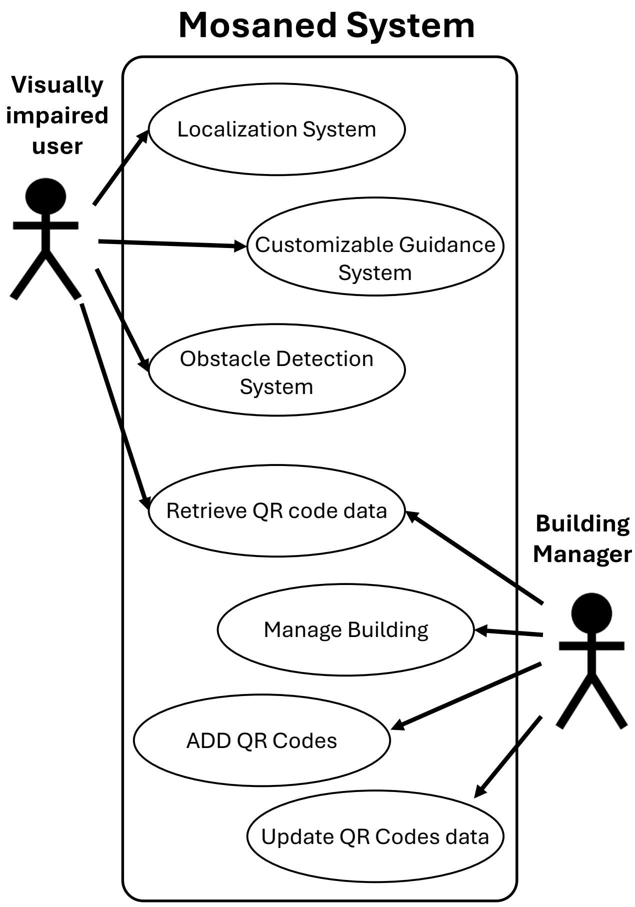

Figure 3.1: Use Case Diagram of Mosaned System

3.1.3 System Architecture Diagram

The system architecture diagram (see Figure 3.2) provides a high-level view of how the components are connected and interact, where the arrows show the flow of data.

This high-level overview introduces the core components and their functions within the system. Subsequent sections provide detailed explanations of each component, their implementation, and integration.

3.2 Database System

The database system serves as the foundational framework for managing data storage, facilitating requests, and delivering instructions in response to QR code scans. It plays a dual role by serving both visually impaired users and building managers. For visually impaired users, the database ensures timely access to navigation instructions, location-specific guidance, and other essential information linked to QR codes. For building managers, it provides a centralized

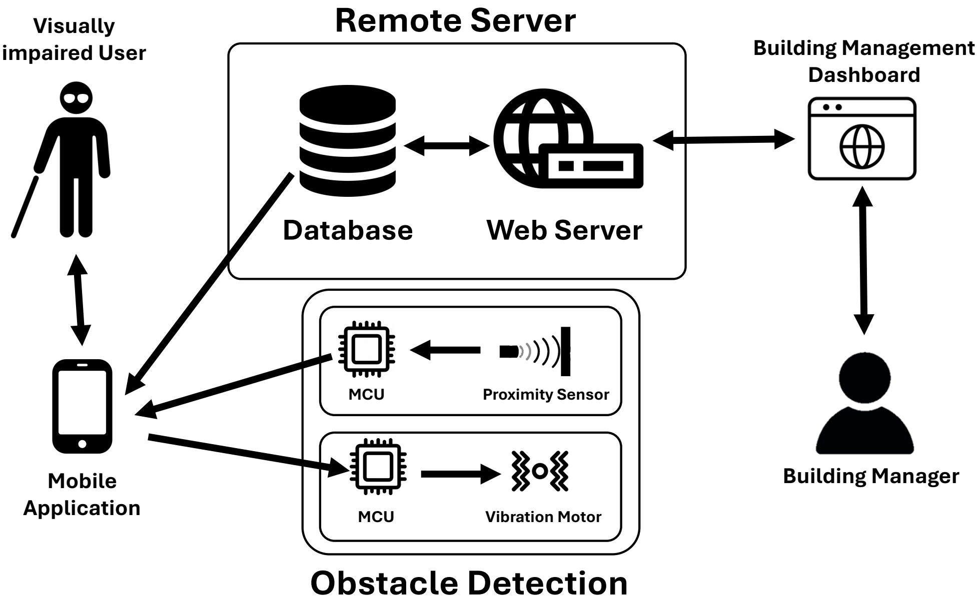

Figure 3.2: System Architecture Diagram of Mosaned System

platform to manage building layouts, QR code global position, and instructions, allowing seamless updates and configuration.

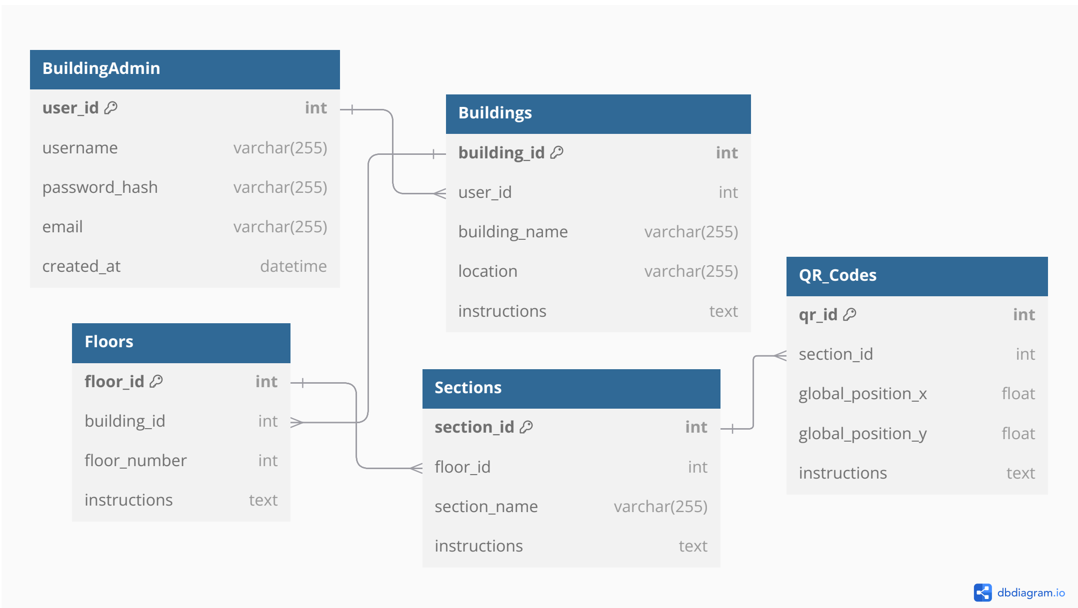

Figure 3.3: Entity-Relationship Diagram (ERD) of the Mosaned Database System

The database structure, shown in Figure 3.3, organizes data into five interconnected tables to ensure efficient management of users, buildings, floors, sections, and QR codes. By maintaining this structured hierarchy, the database supports both the localization system and the customizable guidance system, ensuring accuracy and real-time functionality.

Building managers interact with the database through a Building Management Dartboard, enabling them to configure and manage metadata such as building-specific instructions and QR code positions. The metadata and features of the database are detailed in Appendix A.2.

3.3 Building Management Dashboard

The Building Management Dashboard is a web-based tool that allows building administrators to manage the indoor navigation system effectively. This dashboard provides an intuitive interface to control and update QR code instructions, organize building layouts, and ensure the system remains up-to-date and reliable.

Key features include:

- **User Authentication:** Allows administrators to securely log in and access only their managed buildings, ensuring privacy and data security.
- **Building and Floor Management:** Displays a list of buildings and their respective floors, allowing administrators to organize and view all QR codes assigned to each area.
- **QR Code Instruction Editing:** Administrators can add, update, or delete instructions for each QR code in the building, tailoring guidance to specific areas and ensuring data accuracy.
- **Generate QR Codes:** Provides the ability to generate printable QR codes, where each QR code encodes a unique ID corresponding to its entry in the database. This feature allows administrators to create physical QR codes for deployment, ensuring they align with the stored global positions and associated instructions in the system.
- **Bulk Updates:** Allows administrators to perform bulk updates on QR codes, such as applying changes to multiple codes within a single section or floor, saving time and effort.

The dashboard enables efficient oversight and maintenance of the QR code-based navigation system, allowing real-time updates without the need to modify physical QR codes. Additionally, the export feature ensures that administrators can easily produce hard copies of QR codes in a section, floor, building, or a single QR code in a printable format.

3.4 Mobile Application

The Mobile Application serves as the central component for visually impaired users, providing essential functionalities for navigation assistance in indoor environments. Designed with accessibility in mind, the application performs core tasks of localization, guidance, and real-time feedback, integrating seamlessly with backend services and onboard device features. The application is designed to use the mobile device's native camera, speakers, and microphone, ensuring simplicity and ease of use.

The application's functionality includes:

- **QR Code Scanning and Camera Processing:** The mobile application uses the phone's integrated camera to capture frames and detect QR codes in real time. Upon detecting a QR code, the app decodes its unique ID and retrieves corresponding navigation instructions from the backend database.

- **Localization and Guidance System Implementation:** The mobile application determines the user's position by analyzing the data from detected QR codes. Based on this information, the app provides step-by-step navigation guidance to assist users in navigating indoor spaces effectively.

- **Backend Connectivity:** The application connects to a backend server to retrieve necessary data, such as navigation instructions and location updates. This ensures users receive accurate and up-to-date directions in real time.

- **Objects Avoidance Alerts:** The application integrates with an obstacle detection system that uses ultrasonic sensors. Alerts are generated as vibration feedback when nearby obstacles are detected, enabling users to avoid collisions and navigate safely.

- **User-Friendly Interface:** The application interface is designed to be accessible for visually impaired users, incorporating features like TalkBack compatibility. The app provides audio feedback for navigation and tactile feedback for alerts. It ensures intuitive interaction by using larger buttons, high-contrast visual elements, and minimal manual input requirements. All critical features are accessible via simple touch-based navigation, supporting ease of use for individuals with visual impairments.

This design ensures that visually impaired users can navigate indoor environments confidently and independently. The mobile application serves as the core processing unit, coordinating localization, guidance, backend connectivity, and proximity-based vibration alerts, while leveraging accessible design principles to maximize usability.

3.5 Localization System

The basic idea behind our localization system implementation is to capture a frame at real time by a camera, then detect and decode the QR code in it. After that, we will be able to restore the QR code's global position from the server using the decoded data, which is described at...(write where) . Now, we only need to calculate the camera's position relative to the QR code, and then add the result together with the QR code's global position as follows:

$$user\_global\_position = QR\_global\_position + camera\_relative\_position$$

This is the pose estimation approach which is mentioned previously at section 2.4.1.3.

As we just mentioned, the QR code's global position is stored at a server so there is nothing to calculate here, but this is not the case of the camera's relative position. The process of calculating the later is somewhat complicated and not short, and there are several things that need to be calculated first, such as the QR code corners locations at the image frame, camera matrix, and the camera distortion coefficients. There are multiple libraries for localization purposes as we mentioned previously at 2.4.1.3, and we choose OpenCV due to its Android devices support, ease of use, simple API, and performance.

Everything related to the localization will be running at a separate thread from the main thread. This enables the System to perform other tasks simultaneously and independently.

3.5.1 Camera Calibration

We mentioned at...(write where) that users can chose the camera they want to use from the settings after connecting it to the phone. This approach rise the need of calibration system that users can use since the cameras' parameters are distinct and not known. Each camera need to be calibrated at least once at the first time. Then the camera parameters will be stored at the phone storage for reusing in the future. At first, users need to specify the number of rows and columns at the pattern and the width/height of each square in real world units. Then they need to take multiple photos to the pattern at different angles and distances and start the calibration process. After this point, everything else will be done automatically without the need of the users actions no more.

The camera calibration system's implementation is the same as what was mentioned previously at section 2.4.1.3. First the program will iterate through each photo trying to estimate their inner corners points locations using the following OpenCV function:

```
boolean findChessboardCorners(
Mat image,
Size patternSize,
MatOfPoint2f corners
)
```

Params:

- **image:** The current photo of the pattern.
- **patternSize:** Number of inner corners per a chessboard row and column.
- **corners:** Output array of detected corners.

After that, we used the OpenCV function cornerSubPix to refine the detected points. Finally, we used the following OpenCV function for calibration:

```
double calibrateCamera(
List<Mat> objectPoints,
List<Mat> imagePoints,
Size imageSize,
Mat cameraMatrix,
Mat distCoeffs,
List<Mat> rvecs,
List<Mat> tvecs
)
```

Params:

- **objectPoints:** List of real world inner points locations for each image.
- **imagePoints:** List of estimated inner points locations for each image.
- **imageSize:** Size of the calibrated pattern photo.
- **cameraMatrix:** Output matrix for the camera intrinsic values.
- **distCoeffs:** Output matrix for the camera distortion coefficients.

- **rvecs:** Output list contains the rotation of the calibration patterns for each image.
- **tvecs:** Output list contains the translation of the calibration patterns for each image.

3.5.2 Camera's Relative Pose

The first step is to capture frames using a camera at real time. We used cameraX to do so since it is the best for Android devices as we mentioned at 2.4.1.3. Then, each captured frame will be passed to a function for QR code detecting, decoding, and pose estimation, so finally we can calculate camera's relative position. For QR code detecting, there are bunch of useful libraries we can for QR code detecting and decoding as we mentioned previously at 2.3.4. We choose Google's ML kit due to its simplicity and good performance. After successfully detecting the QR code, Google's ML kit will calculate the QR code corners locations for us.

Finally, we use OpenCV's solvePnP to calculate the QR pose function as follow:

```
boolean solvePnP(
MatOfPoint3f objectPoints,
MatOfPoint2f imagePoints,
Mat cameraMatrix,
MatOfDouble distCoeffs,
Mat rvec,
Mat tvec
)
```

Params:

- **objectPoints:** The QR code's four corners locations relative to its center in real world units, such as cm, mm, etc.
- **imagePoints:** The locations of the QR code's corners inside the captured frame in pixels.
- **cameraMatrix & distCoeffs:** These two are crucial for calculating the QR code's pose since they represent the intrinsic values of a camera and its distortion coefficients.
- **rvec & tvec:** These are output vectors for the code's rotation(rvec) and translation(tvec) relative to the camera.

imagePoints is calculated after successfully detecting the QR code by Google's ML kit.

Now we got the QR code's translation vector relative to the camera, we just multiply it with -1 to get the camera translation relative to the QR code.

3.6 Customizable Guidance

The Customizable Guidance system provides navigation assistance to visually impaired users through the mobile application. By utilizing location-specific data, the system retrieves tailored navigation instructions from a central database when a QR code is scanned. This allows users to receive context-aware guidance, such as directions to their destination or specific information relevant to their current location. The system ensures an intuitive and accessible experience as users move throughout a building.

3.6.1 Location-Based Instructions and Information

The Customizable Guidance system operates through the following steps:

- **QR Code Scanning:** When a user scans a QR code using the mobile application's camera, the system identifies the QR code's unique ID.
- **Instruction Retrieval:** The unique ID is sent to a remote database, where it is used to fetch specific instructions or information associated with the QR code's location.
- **Accessible Presentation:** The retrieved instructions are displayed on the mobile screen and are accessible through assistive features like TalkBack, ensuring visually impaired users can hear the information read aloud.

This approach allows visually impaired users to navigate buildings with ease, offering real-time, location-specific assistance that is both customizable and intuitive.

Destination Path Guidance Since the user's exact position and rotation are known through the localization system (See 3.5), the guidance system can determine the optimal path between the user and their desired destination. The system works as follows:

1. The user selects a destination from a displayed list of zones within the building. The list is navigable and selectable using TalkBack, allowing visually impaired users to choose their destination as they typically interact with their phones.

2. Once a destination is selected, the system searches the database for its corresponding location and determines the optimal path.

3. The path is fragmented into several milestones, starting from the user's current position and ending at the selected destination.

4. For each milestone, the system calculates the distance and angle between the user's current position and the next milestone:

$$\text{difference} = \text{milestone\_position} - \text{user\_global\_position}$$

$$\text{distance} = \sqrt{\text{difference.x}^2 + \text{difference.y}^2}$$

$$\text{angle} = \tan^{-1}\left(\frac{\text{difference.y}}{\text{difference.x}}\right)$$

5. Based on these calculations, the guidance system informs the user, via TalkBack, to move in a specific direction and for a specified distance.

6. If the next milestone is on a different floor, the system informs the user that they are standing in front of the stairs or elevator and instructs them to proceed to the correct floor.

7. The system continues to guide the user by repeating steps 3 to 5 until the destination is reached.

This method ensures a seamless and accessible experience for visually impaired users by leveraging their familiarity with TalkBack for destination selection and providing step-by-step guidance to navigate the path effectively.

3.7 Obstacle Detection

The obstacle detection system is designed to enhance the safety and navigation capabilities of visually impaired users by alerting them to nearby obstacles in real time. This system combines ultrasonic sensors, vibration motors, and the mobile application, enabling a seamless flow of information between hardware components and the app. By utilizing the processing capabilities of the mobile application, the system can also be expanded to include additional hardware in the future.

3.7.1 System Overview

The obstacle detection system comprises the following components:

- Four ultrasonic sensors, each facing a specific direction: forward, backward, right, and left.
- Four vibration motors, placed in corresponding positions: forward, backward, right, and left.
- Mobile application to process the sensor data and control vibration feedback.

3.7.2 Ultrasonic Sensor Data Collection

The ultrasonic sensors are positioned to monitor one direction each (forward, backward, right, and left). These sensors measure the time it takes for an ultrasonic pulse to travel to an obstacle and reflect back. This raw time-of-flight data is sent directly to the mobile application for processing.

This approach ensures that the mobile can leverage its processing power to calculate distances and perform additional analyses if required.

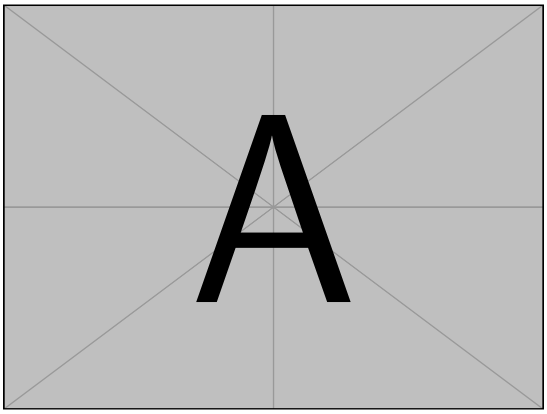

Figure 3.4: Schematics of Ultrasonic Sensors and Mobile Application Integration

3.7.3 Data Processing and Mobile Application

Upon receiving the raw time-of-flight data from the sensors, the mobile application processes the data to compute the distance to obstacles in each direction. The distance is calculated using the following equation:

$$d = \frac{v \cdot t}{2}$$

Where:

- d: Distance to the obstacle (in meters).
- v: Speed of sound in air (approximately $343\,\mathrm{m/s}$).

- t: Round-trip time of the ultrasonic pulse (in seconds).

The division by 2 accounts for the fact that the pulse travels to the obstacle and back.

The mobile application determines whether the distance is within the actionable range (default is less than 2 meters). If the obstacle is within range, the application calculates the intensity level for the corresponding vibration motors. The intensity I is adjusted to account for the vibration motor's operating voltage threshold of 2.7V, as follows:

$$I = \begin{cases} 2.7 + (0.6 \cdot (1 - \frac{d}{2})), & \text{if } d \leq 2 \\ 0, & \text{if } d > 2 \end{cases}$$

Where:

- I: Intensity of vibration (ranges from 2.7 to 3.3V for actionable distances).
- d: Distance to the obstacle (in meters).
- 3.3: Maximum vibration intensity.
- 2.7: Minimum voltage required for motor activation.

This ensures that:

- When the obstacle is at the minimum measurable distance of $d = 0.03\,\text{m}$, the vibration intensity is at its maximum: $I = 3.3$.
- When the obstacle is at $d = 2\,\text{m}$, the vibration intensity is $I = 2.7$, which is just enough to activate the vibration motor.
- When $d > 2\,\text{m}$, no voltage is applied ($I = 0$), and the motor remains idle.

The application sends these calculated intensity values to the vibration motors for tactile feedback.

3.7.4 Vibration Feedback Mechanism

The vibration motors receive the calculated intensity values, which are mapped to an analog voltage ranging from 2.7 to 3.3V using Pulse Width Modulation (PWM). The voltage controls the vibration intensity, ensuring a proportional response based on the proximity of obstacles.

$$\text{Analog Voltage} = I \quad (\text{where } 2.7 \leq I \leq 3.3).$$

Using PWM, the mobile application controls the vibration motors as follows:

- If $I = 3.3$, the PWM signal generates the maximum voltage, resulting in the strongest vibration.
- If $I = 2.7$, the PWM signal generates the minimum voltage required for vibration, providing low-intensity feedback.
- Intermediate values of I produce proportional vibrations, allowing the user to gauge obstacle proximity intuitively.

This method ensures that the vibration motors are only activated when necessary and provides a clear indication of the obstacle's distance.

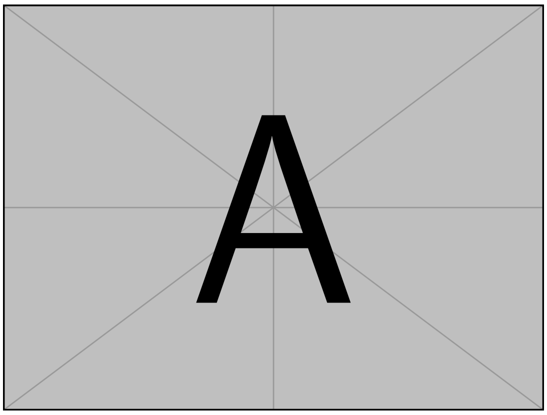

Figure 3.5: Schematics of Vibration Motors and PWM Control

The communication channel between the mobile application and the ESP32 microcontrollers is designed to be flexible, allowing additional hardware to be integrated easily. For instance, other types of sensors could be added to the system, with the mobile leveraging its processing power to analyze and act on the new data.

3.8 Experiment Setup

This section describes the setup of the system for conducting experiments, as part of the methodology. Once the camera calibration process is complete—an initial step that requires assistance—the system is ready for operation. The following outlines the arrangement and configuration of the components for the experiment.

3.8.1 Mobile Phone and Stand Configuration

The mobile phone, which serves as the central processing unit, is mounted on a stand equipped with a four-wheel base for stability and mobility. The stand is adjustable in height and is set at approximately 1.5 meters to align the phone's camera with the placement of the QR codes on the walls. The camera is positioned to face the walls perpendicularly, ensuring optimal detection of the QR codes.

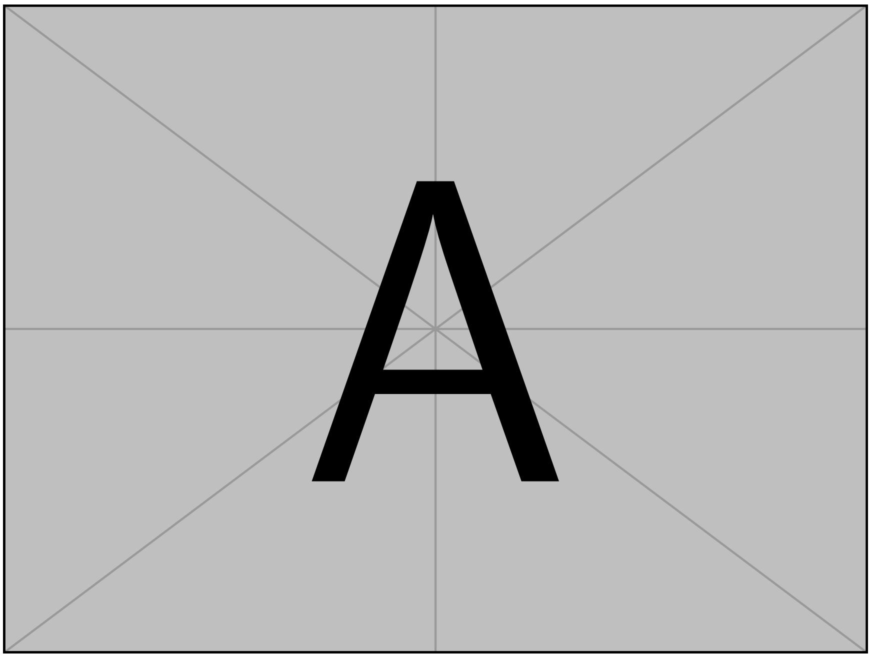

Figure 3.6: Mobile Phone Mounted on Adjustable Stand

3.8.2 QR Code Placement

For the experiment, QR codes are strategically positioned at a height of 1.5 meters from the ground to match the camera's alignment on the stand. This ensures that the system can accurately scan and decode the QR codes without requiring additional adjustments during the experiment.

3.8.3 Wearable Obstacle Detection System

The wearable obstacle detection system comprises the ultrasonic sensors, vibration motors, and ESP32 microcontrollers integrated into a belt. The configuration is as follows:

- The four ultrasonic sensors are placed in forward, backward, right, and left orientations on the belt to monitor the user's surroundings.
- The four vibration motors are positioned correspondingly to provide tactile feedback in the same directions.

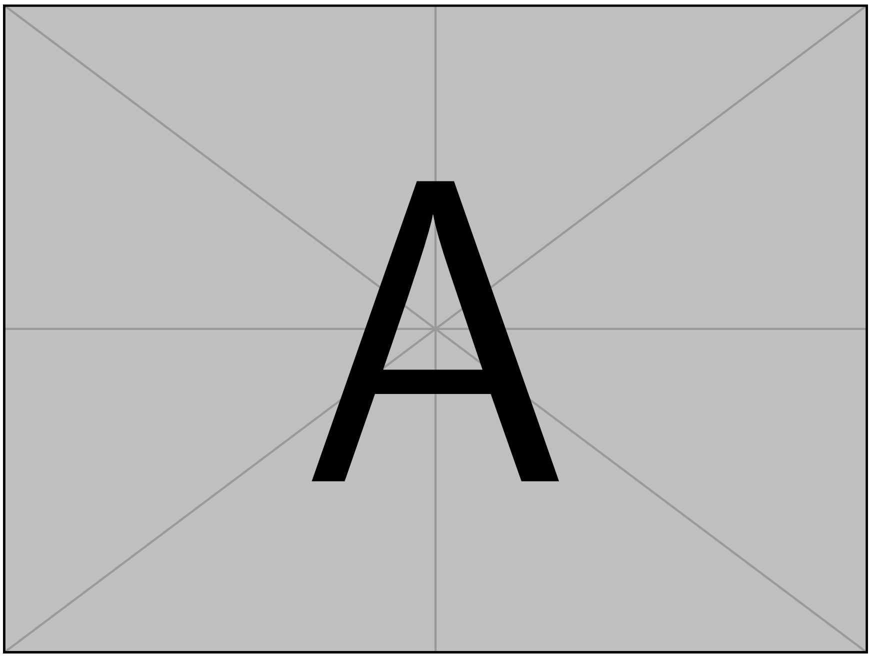

Figure 3.7: QR Code Placement on Wall

- Both ESP32 microcontrollers are powered by a compact power bank, with all wiring securely hidden within the belt for a clean and unobtrusive design.

The belt is worn by the user, allowing them to experience the obstacle detection and feedback system in a practical, hands-free manner.

3.8.4 Back-end Infrastructure Hosting

The backend database and web host are deployed on a remote server to ensure seamless data access and scalability during the experiment. This configuration is necessary because the system relies on two key components hosted on the remote server:

- **Building Management Dashboard:** A web-based dashboard allows building managers to efficiently update, add, or modify QR code-related data, including global positions and navigation instructions. This dashboard provides centralized control over building layouts, ensuring that updates are reflected across the entire system in real time.
- **Database:** The database stores essential information about QR codes, such as their global positions and associated instructions. It serves as the central data repository, accessed by both the mobile application and the building management dashboard to retrieve or update QR-related data.

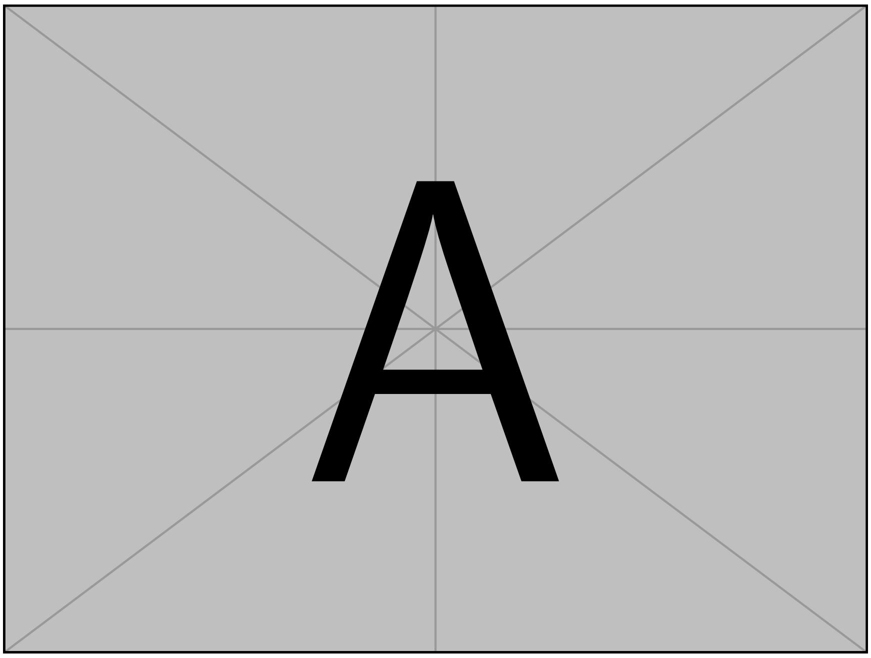

Figure 3.8: Wearable Obstacle Detection System with Integrated Belt

This remote hosting approach ensures seamless integration between the components, providing high availability and reliability. It also supports centralized management, allowing building managers to make updates remotely without requiring changes to local hardware or application configurations.

3.8.5 Power Supply and Connectivity

The system relies on portable power sources to maintain functionality throughout the experiment:

- The mobile phone operates independently on its internal battery.
- The ESP32 microcontrollers are powered by a compact power bank attached to the wearable belt.
- Bluetooth Low Energy (BLE) ensures seamless communication between the ESP32 devices and the mobile application.
- The mobile application connects to the remote server Wi-Fi access point mobile data for backend interactions.

This setup guarantees portability, ease of use, and robust connectivity for real-world testing and experimentation.

This configuration ensures that the components are aligned and integrated effectively, enabling accurate testing and evaluation of the system in a controlled environment. The setup is designed to reflect a real-world use case as closely as possible, while allowing for flexibility and adjustments as needed during the experiment.

3.9 Test & Evaluation

To validate the effectiveness and usability of the proposed system, two evaluations will be conducted: an assessment of the overall user experience and a focused evaluation of the localization system.

3.9.1 Evaluation of Overall Experience

This evaluation focuses on the user's journey, excluding localization, as it is tested separately. A group of six participants will be blindfolded, with three using the system and three relying on traditional navigation methods. The evaluation will take place in an unfamiliar building. The participants' experiences will be assessed using the following survey:

Survey Questions

1. How confident did you feel navigating the building?
2. How easy was it to identify obstacles and avoid them?
3. Did you find the instructions clear and helpful?
4. How comfortable were you using the system during the task?
5. Would you recommend this system to others with similar needs?

Where the possible answers will be collected on a likert scale of Strongly Disagree, Disagree, Neutral, Agree, and Strong Agree.

3.9.2 Evaluation of Localization System

To test the localization system's accuracy, the evaluation will take place in a controlled 4x4 meter room. Four QR codes will be placed on the walls encoding their global positions. The system will estimate its location based on the detected QR codes and compare the results to the actual known positions.

Evaluation Procedure

1. The user stands at predefined positions in the room, and the system records the estimated positions based on QR code data.

2. The recorded positions are compared to the actual known coordinates.

3. The error in localization is calculated as the Euclidean distance between the estimated and actual positions.

This evaluation will determine the system's precision and reliability in providing accurate localization data under ideal conditions. These results, combined with the user experience feedback, will provide a comprehensive assessment of the system's effectiveness.

Chapter 4

Results and Discussion

4.1 Results

This chapter presents the results obtained from the system.

Scenario	**Expected Result**	**Actual Result**
Scenario 1	Success	Success
Scenario 2	Failure	Success

Table 4.1: Results of Various Scenarios

Table 4.1 shows the results obtained during testing in various scenarios.

Chapter 5

Conclusion and Future Work

5.1 Conclusion and Future Work

The project concludes by summarizing the findings and suggesting future work.

References

[1] Michail K., Brennan C., Sabrina C., Anand A., Camden W., & Nikolaos V. (2021). Fiducial Markers for Pose Estimation. Journal of Intelligent & Robotic Systems (2021) 101:71. https://link.springer.com/article/10.1007/s10846-020-01307-9

[2] Tiwari, V., "QR Code Technology and Its Applications," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 6, no. 4, pp. 123-127, 2016.

[3] Android, "Official Android Website," Available: https://www.android.com/.

[4] Espressif Systems, "ESP Modules," Available: https://www.espressif.com/en/products/modules.

[5] Google Accessibility, "Use TalkBack on Android Devices," Available: https://support.google.com/accessibility/android/topic/10601571?hl=en&ref_topic=3529932.

[6] E. Chian, "A Single Camera 3D Functions," Medium, Available: https://eugene-chian.medium.com/a-single-camera-3d-functions-fdec7ffa9a83.

[7] H. Zhang, C. Zhang, W. Yang and C. -Y. Chen, "Localization and navigation using QR code for mobile robot in indoor environment," 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), Zhuhai, China, 2015, pp. 2501-2506, doi: 10.1109/ROBIO.2015.7419715.

[8] D. Ahmetovic, C. Gleason, C. Ruan, K. M. Kitani, H. Takagi and C. Asakawa, "NavCog: a navigational cognitive assistant for the blind," *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services*, Florence, Italy, 2016, pp. 90-99, doi: 10.1145/2935334.2935361.

[9] IBM, "What Is an API (Application Programming Interface)?," *IBM Topics*, 2024. [Online]. Available: https://www.ibm.com/topics/api. [Accessed: 28-Oct-2024].

[10] Visual Paradigm, "What Is Entity Relationship Diagram?," *Visual Paradigm*, 2024. [Online]. Available: https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/. [Accessed: 28-Oct-2024].

[11] Luca C., Gionata C., Francesco F., Alessandro F., Gianluca I., Andrea M. Federica V. (2017). A QR-code Localization System for Mobile Robots: Application to Smart Wheelchairs. https://ieeexplore.ieee.org/document/8098667

[12] Vybronics, "VC0820B006F Coin Vibration Motor Datasheet," Available: `https://www.vybronics.com/wp-content/uploads/datasheet-files/Vybronics-VC0820B006F-datasheet.pdf`.

[13] SparkFun Electronics, "Ultrasonic Ranging Module HC-SR04 Datasheet," Available: `https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf`.

[14] J.-I. Kim, H.-S. Gang, J.-Y. Pyun, and G.-R. Kwon, "Implementation of QR Code Recognition Technology Using Smartphone Camera for Indoor Positioning," Energies, vol. 14, no. 10, p. 2759, 2021, doi: 10.3390/EN14102759.

[15] S. -J. Lee, G. Tewolde, J. Lim and J. Kwon, "QR-code based Localization for Indoor Mobile Robot with validation using a 3D optical tracking instrument," 2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), Busan, Korea (South), 2015, pp. 965-970, doi: 10.1109/AIM.2015.7222664.

[16] M. Kalaitzakis, B. Cain, S. Carroll, A. Ambrosi, C. Whitehead, and N. Vitzilaios, "Fiducial Markers for Pose Estimation: Overview, Applications, and Experimental Comparison of the ARTag, AprilTag, ArUco, and STag Markers," Journal of Intelligent & Robotic Systems, vol. 101, no. 71, pp. 1-26, 2021, doi: 10.1007/s10846-020-01307-9.

[17] Aqeel Anwar, "What are Intrinsic and Extrinsic Camera Parameters in Computer Vision?", A detailed explanation of intrinsic and extrinsic camera parameters with the help of visualizations. https://towardsdatascience.com/what-are-intrinsic-and-extrinsic-camera-parameters-in-computer-vision-7071b72fb8ec

[18] M. Kuribayashi, T. Ishihara, D. Sato, J. Vongkulbhisal, K. Ram, S. Kayukawa, H. Takagi, S. Morishima and C. Asakawa, "PathFinder: Designing a Map-less Navigation System for Blind People in Unfamiliar Buildings," *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*, New York, NY, USA, 2023, Article 41, pp. 1–16, doi: 10.1145/3544548.3580687.

[19] A. L. Fraga, X. Yu, W.-J. Yi, and J. Saniie, "Indoor Navigation System for Visually Impaired People using Computer Vision," *2022 IEEE International Conference on Electro Information Technology (eIT)*, Grand Forks, ND, USA, 2022, pp. 257-260, doi: 10.1109/eIT53694.2022.9800294.

[20] https://developer.android.com/media/camera/choose-camera-library

[21] S. Tiwari, "An Introduction to QR Code Technology," 2016 International Conference on Information Technology (ICIT), Bhubaneswar, India, pp. 39-44, 2016, doi: 10.1109/ICIT.2016.021.

[22] S.G. Leitch, Q.Z. Ahmed, W.B. Abbas, M. Hafeez, P.I. Laziridis, P. Sureephong, and T. Alade, "On Indoor Localization Using WiFi, BLE, UWB, and IMU Technologies," Sensors, vol. 23, no. 20, pp. 8598, 2023, doi: 10.3390/s23208598.

[23] "QRCode Monkey - Free QR Code Generator," QRCode Monkey, Available: `https://www.qrcode-monkey.com`.

[24] "QR TIGER - Free QR Code Generator," QR Tiger, Available: `https://www.qrcode-tiger.com`.

[25] "PyQRCode Documentation," PyQRCode, Available: `https://pyqrcode.readthedocs.io`.

[26] "ZXing Decoder," ZXing Project, Available: `https://github.com/zxing/zxing`.

[27] "Core Image Framework," Apple Developer Documentation, Available: `https://developer.apple.com/documentation/coreimage`.

[28] "Segno - Python QR Code Generator," Segno Documentation, Available: https://segno.readthedocs.io.

[29] "AVFoundation Framework," Apple Developer Documentation, Available: https://developer.apple.com/documentation/avfoundation.

[30] https://www.youtube.com/watch?v=-L-WgKMFuhE&list= PLFt AvWsXl0cq5Umv3pMC9SPnKjfp9eGW

[31] "QRickit QR Code Decoder," QRickit, Available: https://qrickit.com.

[32] OmniVision Technologies. (2008). 5-Megapixel SOC Camera Module. https://cdn.sparkfun.com/datasheets/Sensors/LightImaging/OV5640_datasheet.pdf

[33] OmniVision Technologies. (2006). OV7670 Camera Module Datasheet. https://web.mit.edu/6.111/www/f2016/tools/OV7670_2006.pdf

[34] OmniVision Technologies. (2016). OV2640 Camera Module Datasheet. https://www.uctronics.com/download/OV2640_DS.pdf

Appendix A

A.1 The divided environment method

The idea of dividing an environment into smaller pieces is widely used in some industries in different ways and purposes. One of these industries is the video game industry. A huge amount of video games divide the game world into a chunk of squares, triangles, hexagons, and other shapes depending on the game's need and performance. But the difference here is that they do not use this technique for localization since the positions for all the objects in the games are known and stored at the RAMs already. They use the technique to categorize the pieces such as walkable ground, water, lava, rocks, and so on. Then these pieces are used along with their categories to find a proper path between two points. Different path finding algorithms can be used, but the most popular and simple one is A* algorithm. See this [30] incredible youtube tutorial made by Sebastian Lague, that explain an implementation of the A* algorithm and how it can be optimized.

A.2 Metadata and Features of the Database

The database system is composed of five main tables, each designed to manage specific elements of the Mosaned system. This section details the metadata and features associated with each table.

- **BuildingAdmin:**
  - **user_id (Primary Key)**: A unique identifier for each user of the system, enabling individual management of buildings and associated data.
  - **username**: The name chosen by the user for login purposes.
  - **password_hash**: A secure hash of the user's password used for authentication.
  - **email**: The user's email address, utilized for notifications and account recovery.
  - **created_at**: A timestamp indicating when the user account was created.
- **Buildings:**
  - **building_id (Primary Key)**: A unique identifier for each building within the system.
  - **user_id (Foreign Key)**: Links the building to the associated user in the `BuildingAdmin` table, ensuring that each building is managed by the correct user.
  - **building_name**: The name of the building.
  - **location**: The physical address or description of the building's location.
  - **instructions**: General instructions related to the building, providing contextual information for navigation and management.
- **Floors:**
  - **floor_id (Primary Key)**: A unique identifier for each floor within a building.
  - **building_id (Foreign Key)**: Associates the floor with its corresponding building, establishing a clear hierarchical structure.
  - **floor_number**: Indicates the number of the floor (e.g., 1 for the first floor, 2 for the second, etc.).
  - **instructions**: Specific instructions related to that floor, assisting users with effective navigation.
- **Sections:**

- **section_id (Primary Key)**: A unique identifier for each section on a floor.
- **floor_id (Foreign Key)**: Links the section to its respective floor, maintaining the organizational hierarchy.
- **section_name**: The name or description of the section within the floor.
- **instructions**: Instructions specific to the section, enhancing the user's ability to navigate the area.

- **QR_Codes:**
  - **qr_id (Primary Key)**: A unique identifier for each QR code, ensuring no duplicates exist within the system.
  - **section_id (Foreign Key)**: Connects the QR code to its specific section.
  - **global_position_x**: The X-coordinate representing the global position of the QR code.
  - **global_position_y**: The Y-coordinate indicating the QR code's position, further aiding in navigation.
  - **instructions**: Specific instructions directly linked to the QR code, providing targeted guidance for users when the QR code is scanned.