PALESTINE POLYTECHNIC UNIVERSITY College of Information Technology and Computer Engineering Computer Engineering

Personalized Tracking Robot (PTR)

Project Members:

Raghida Thabainah

Zaina Abu Allan

Israa Amro

Supervisor: Dr. Alaa Halawani

Hebron - Palestine

2024

Acknowledgment

The Giver of Light, the Weaver of Paths, the Ever-magnificent, the Ever-Thankful, Alhamdulillah, praise be to Allah the Almighty of God the most Gracious and the most Merciful, for all the blessings you have given us. The ability, courage, endurance, and patience you put inside us strengthened us in completing this Project.

First and foremost, we want to remember and honor the brave people and martyrs in Gaza who spent their lives for freedom and justice. Their strength reminds us of challenges beyond our studies, and we salute their memory. They remain alive in our hearts and our history.

Our deepest appreciation to our mothers for their precious and continuous guidance and support throughout our lives, and for all the daily prayers that you have dedicated to us. To our fathers and family members who have supported us so far and who have been the source of inspiration, thank you for always being by our side.

Heartfelt appreciation to Eng. Dr. Alaa Halawani, our project supervisor, who has been a guiding light, leading us down the right path and offering invaluable direction. His continuous guidance, encouragement, and unwavering support throughout the semester have been instrumental to our success.

To those who stood by us through thick and thin, making awesome memories in tough and good times. A big thanks to our special friends Maria, Ayat, Haya, Raghad Asil, and, Niveen for always

being there, encouraging us, and making this journey truly special.

ABSTRACT

The custom tracking project is important and effective in the current era, designed primarily to help in tracking its owner using advanced technologies. The system utilizes a variety of ROS servers and tools to manage data and integrate different algorithms for owner identification and tracking. The PyzBar library is used to identify the owner through a QR code, while a particle filtering algorithm is implemented for the tracking process. The Kinect camera's internal sensors are employed to maintain the owner's safety, and an ultrasonic sensor is used to detect and overcome obstacles. Ultimately, the project resulted in a robot capable of tracking the owner based on a specific QR code, with the ability to navigate around obstacles and maintain a safe distance between itself and the owner.

Keywords: ROS, mobile robot, Tracking algorithm, Navigation, QR, Detection algorithm

الملخص

يعتبر مشروع المتتبع المتخصص شيئا مهما وفعالا في العصر الحالي، وهو مصمم بشكل أساسي للمساعدة في تتبع صاحبه باستخدام التقنيات المتقدمة. ويستخدم النظام مجموعة متنوعة من خوادم وأدوات ROS لإدارة البيانات ودمج الخوارزميات المختلفة لإجراء عملية تحديد هوية المالك وتتبعه.تم استخدام مكتبة PyzBar للتعرف الى المالك من خلال رمز QR، ثم استخدمت خوارزمية particle filter لتنفيذ عملية التتبع، وأيضا استخدمنا المستشعرات الداخلية لكاميرا Kinect للحفاظ على سلامة المالك، بينما استخدم مستشعر الموجات فوق الصوتية لتحديد والتغلب على العقبات.وفي النهاية تم الحصول على روبوت قادر على تتبع صاحبة بناء على QR محدد مع القدرة على التغلب على العقبات والحفاظ على مسافة امان بينه وبين المالك .

كلمات مفتاحية : نظام تشغيل الروبوت ، التعرف على QR ، تتبع المالك .

TABLE OF CONTENT

Contents

List of Figures

List of Tables

Chapter 1: Introduction

This section provides a comprehensive overview, detailing the motivation, importance, objectives, and a concise description of the problem statement, along with an outline of the report sections.

1.1 Overview of the project

We live in an era where technological development has become an important thing in our lives, and the presence of a robot is something that has facilitated many tasks for humans. This era is characterized by an unprecedented and remarkable integration between technological innovation and rapidly developing societal trends. From here came the need for the presence of robots that make it easier for humans to carry out complex tasks.

Hence, the Personal Tracking Robot (PTR) emerges as a pioneering and very important project to meet the growing need for personal assistance and companionship, as the primary function of this robot is to track its owner through its own QR, which can be used in several fields.

1.2 Motivation

The primary motivation behind undertaking the (PTR) project is society’s need for a paradigm shift in societal expectations, as people have come to recognize the importance of interaction between humans and machines, due to the accelerating pace of modern life, in addition to the increasing proportion of the elderly. The increasing number of individuals with specific requirements confirms the need for an adaptable, intelligent, and versatile tracking robot that can provide personalized assistance in real-time scenarios.

1.3 Importance

The project's importance extends beyond mere technological innovation. It represents a qualitative leap in the world of assistive robots, and its ability to revolutionize the way individuals are monitored and supported in various aspects of their lives, as its function is not limited only to functional requirements, but also deals with the emotional and psychological aspects of the well-being of individuals, and focuses on safety and privacy by providing a comprehensive solution. To meet the diverse needs of users and go beyond traditional tracking mechanisms, the project

(PTR) is evidence of the transformative power of technology in improving and facilitating human life.

1.4 Objectives

The primary goal of the project is to develop and deploy an advanced tracking robot capable of providing personal assistance through the seamless integration of advanced technologies such as computer vision and sensor systems, ensuring user privacy and safety, and enhancing the general well-being of individuals in need of support.

Therefore, the project aims also:

1- Accurately identify a specific individual via their QR, using the QR algorithm (Pyzbar)
2- Track the owner using the particle filter
3 - Overcoming obstacles using ultrasonics.

1.5 Problem Statement

In our current era, the need for a personal tracking robot that helps humans accomplish a specific task saves their owner time and effort, and protects them from performing dangerous tasks, has become extremely important and necessary at present. The research problem in the Personal Tracking Robot project revolves around developing and improving an autonomous robotic system capable of providing accurate tracking of its owner and personal assistance in dynamic environments. The challenge lies in the design:

This research came to explore innovative solutions to enhance the performance and user experience and address the complex interaction between robots, artificial intelligence, and human factors, to create an accurate and adaptable personal tracking robot for practical applications in various fields, including health care, commerce, and other different environments.

1.6 Requirements

1.6.1 Functional Requirements:

1- **User identification:** The individual must be identified accurately and reliably in various environments.

2- **Navigation system:** It should also be able to move in all spaces and different internal environments, whether in a corridor or a closed room, with ease and flexibility, and it must be able to detect obstacles and avoid them.

3- **Real-time tracking:** The PTR device must track an individual's movement in real-time, provide a continuous update of the owner's location with a latency of no more than one second, and be able to adapt to the owner in terms of direction.

1.6.2 Non-Functional Requirements:

1- **Safety:** The robot must maintain a distance of approximately 45 cm between the robot and its owner, and this distance may not be exceeded.

2- **COST:** We keep the project within a specified budget without compromising essential features and performance.

1.7 Project boundaries and constraints

1. Environmental conditions: The robot is designed and tested for operation in an indoor environment.
2. The robot can only navigate in a single-layer environment and cannot move up and down.
3. The brightness of the light may affect the QR code recognition speed

1.8 Short description of the system

Figure 1.1 shows us the external environment of the personal tracking robot, as the first component is the personal tracking robot that tracks its owner and provides assistance to him, and the second component is the target party that will use the robot and interact with it.

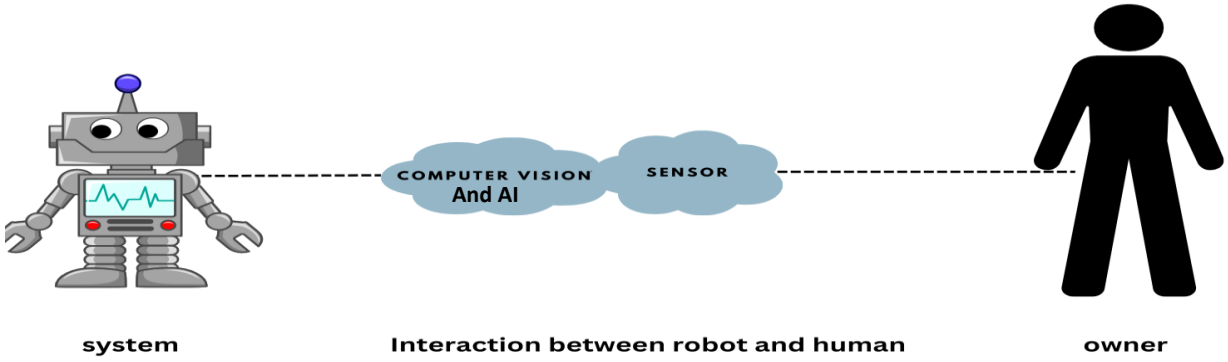

Figure 1.1: The external environment of a system

Chapter 2: Background

2.1 Overview

This chapter presents the theoretical background of the project and the hardware and software components that we will use to build the project, in addition to a review of previous studies.

2.2 Theoretical Background

This section will provide an overview of the pieces and techniques used:

2.2.1 Sensors

A sensor is a device that detects and responds to some type of input from the physical surrounding environment. Input such as light, heat, motion, obstacles, moisture, …etc, the output is generally a signal that is converted to a human-readable form [1].

2.2.2 Arduino microcontroller

The microcontroller is an integrated circuit (IC) or chip used to accept conditioned inputs from sensors, process the input, then process the required output. An Arduino microcontroller is based on the open-source hardware/software programming platform named Arduino which is based on the Atmel microcontrollers. it's free to be used and modified by the people. The Arduino microcontroller can be told what to do by connecting it to a PC with the help of a USB cable and then using the Arduino IDE to write the required code. This code can then be exported to the Arduino board and run [2].

2.2.3 Ultrasound frequency & Ultrasonic modules

The ultrasonic sensor used in this project is HC-SR04, it uses sonar to determine the distance of an object, this sensor can detect an obtained object from 2 cm to 400 cm in 30 degrees, and its sensitivity can be up to 3mm [3].

The ultrasonic modules detect the distance to an object by sending out a short burst of ultrasonic sound and then listening for the echo of it; by measuring the time it takes for the sound to return to the sensor the module can determine how far away an object is [4].

In each module, the left eye serves as a speaker and transmits the signal, while the right eye serves as a microphone and listens for the return signal. Each module has four pins. There is a ground pin, a voltage pin, a trigger input, and an echo output pin, the module starts a measurement when the trigger pin receives a 5V pulse for at least 10 μs... The sensor then sends out a cycle of eight 40 kHz ultrasonic bursts and waits for any reflections to return. When an ultrasonic burst is received back, the echo pin is set to 5V and delayed for a period proportional to the distance the burst traveled.

The distance to the object in front of the sensor can thus be calculated from the echo signal and the detection objects in a range utilizing the speed of sound.

2.2.4 Computer vision

Rasche [5] defined Computer vision as the field dedicated to interpreting image content. It involves classifying entire images, such as those uploaded to platforms like Facebook and Instagram. Additionally, computer vision focuses on recognizing objects within images, such as faces, cars, and license plates. It also plays a crucial role in detecting specific aspects within images, such as identifying cancer in biomedical images.

2.2.5 Image processing

Banda [6] defined Image processing as converting an image into digital form and performing various operations to enhance the image or extract valuable information. It is a type of signal processing where the input is an image, and the output is either an improved image or characteristics associated with that image.

Image processing algorithms are used to analyze and extract information from the camera input, which is very useful in this project, such as identifying and tracking a person. The most popular algorithms include OpenCV and the perception package in ROS.

2.2.6 Robot Operating System (ROS)

Kane [7] A ROS (Robot Operating System) is an open-source meta-operating system designed for robots. It offers typical operating system services such as hardware abstraction, low-level device control, implementation of common functions, inter-process messaging, and package management. Additionally, ROS provides tools and libraries for retrieving, building, writing, and executing code across multiple computers.

ROS can be chosen as the operating system in this project for several reasons:

1- **Modeling**: this is designed with a modular architecture that allows developers to break down complex robot behavior into small, reusable components called nodes. These nodes can interact with each other through well-defined interfaces, making it easier to build, test, and maintain complex robotic systems.

2- **Large Community**: ROS has a large and active community of developers, researchers, and users. The community provides resources, support, and an extensive library of existing software packages and tools that can be used to accelerate development and solve common robotics challenges.

3- **Platform Independence**: ROS is platform-independent, meaning it can run on various operating systems such as Linux, macOS, and Windows. This flexibility allows developers to choose the hardware and software platform that best suits their needs."

2.2.7 Obstacles Avoidance

Ultrasonics plays an important role in the process of overcoming obstacles, as this sensor works by emitting ultrasonic waves and measuring the time it takes for the echo to return after bouncing off the body, and from it, the distance between it and the obstacle is calculated [4].

How ultrasonic sensors work in the process of crossing obstacles:

1- Emission of sound waves: Ultrasonic sensors consist of two main components: a transmitter and a receiver. The transmitter sends out high-frequency sound waves.

2- Echo reception: When a sound wave hits an object, it reflects to the sensor. The receiver then receives this echo.
3- Timing: Measures the time it takes for the echo to return.
4- Distance Calculation: Based on the speed of sound, the distance to an object is calculated using the following formula:

$$\text{Distance} = \text{Time} \times \text{Speed of Sound} / 2$$

Based on this information, the code is written so that it takes the appropriate action to overcome obstacles, such as ordering the robot to stop or turn.

2.2.8 Navigation

Navigation involves directing a robot from one location to another within a given environment. This process includes determining the robot's path, avoiding obstacles, and finally reaching the intended destination. Many algorithms and technologies contribute to navigation, including path planning and object detection and recognition [8].

2.2.9 Mobile robot

A mobile robot is an autonomous machine that can move in different environments. Unlike stationary industrial robots, mobile robots are not fixed in one location but are designed to navigate different terrains and environments. They are a subset of robotics and information technology and come in many forms and applications.

2.2.10 Object Detection and Tracking

Object detection is essential to start the tracking process. It is applied consistently in every frame. A common approach to detecting moving objects is to use temporal information extracted from a series of images, for example, by calculating the difference between frames, learning a static background scene model and comparing it to the current scene, or finding high-motion areas. Another common way to detect objects is by placing a certain tag on the person, for example, a QR code, and then detecting the object by this tag [9].

Object tracking is a critical process in many applications, including robotics, surveillance, and augmented reality. It involves continuously identifying and monitoring the position of a specific object within a sequence of frames or over time. Advanced tracking algorithms, such as particle filters or deep learning-based methods, are often employed to achieve accurate and reliable tracking. These algorithms handle various challenges, including changes in the object's appearance, occlusions, and dynamic backgrounds. By continuously updating the object's position and adapting to new observations, object tracking systems can provide real-time data that is essential for navigation, decision-making, and interaction in complex environments [10].

2.2.11 particle filter

A particle filter, also known as a Sequential Monte Carlo (SMC) method, is a powerful algorithm used to estimate the state of a system that evolves. It is particularly useful in situations where the system is nonlinear and non-Gaussian, making it a popular choice for tracking applications. This algorithm is widely used in robotics, computer vision, and navigation systems. [11], The block diagram of the particle filter-based visual object tracking, as shown in Figure 2.1

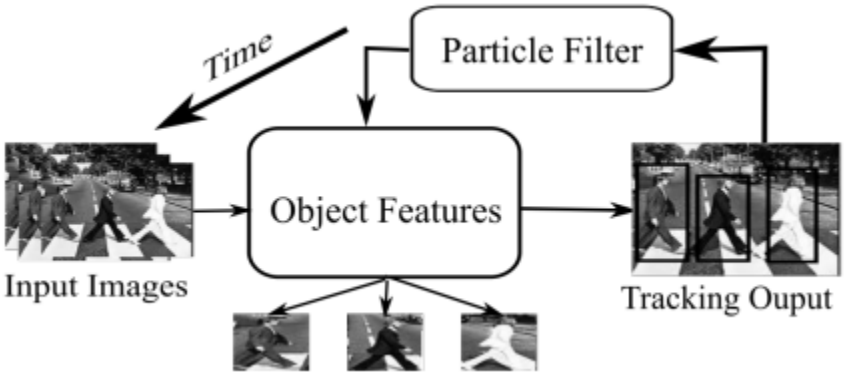

Figure 2.1: Block diagram of the particle filter-based visual object tracking.[11]

The particle filter represents the probability distribution of the system's state using a set of discrete particles. Each particle represents a possible state of the system and has an associated weight that indicates the likelihood of that state given the observations [11].

The process starts with the initialization of a set of molecules. Each particle represents a possible state (position and orientation) of the robot, and the system estimates the current state of the robot based on the initial set of particles. Next, the new position of the particle is predicted based on the robot's motion model. This prediction takes into account the control input or robot motion while reading the QR from the current frame or the image captured by the robot's camera.

Using the information in the QR, the system computes a monitoring model. This model provides a way to compare predicted observations (given particle positions) to actual observations. The particles are then weighted according to the observation model. Particles that are more consistent with actual observations are given higher weights, indicating that they are more likely to represent the actual state of the robot. Based on the updated weights, a resampling process is performed. Particles with higher weights may be selected, while particles with lower weights may be discarded. This step helps focus computing resources on the most promising particles. The current state of the robot is estimated based on the reconstructed particles. This estimate represents the most likely current state of the robot.

The process then returns to the state estimation step and continuously refines the robot's state estimate as new data is received and processed. This iterative approach allows the particle filter to adapt to changes in the environment and the robot's motion, providing robust localization and state estimates over time. Figure 2.2 shows the Flow char of the algorithm

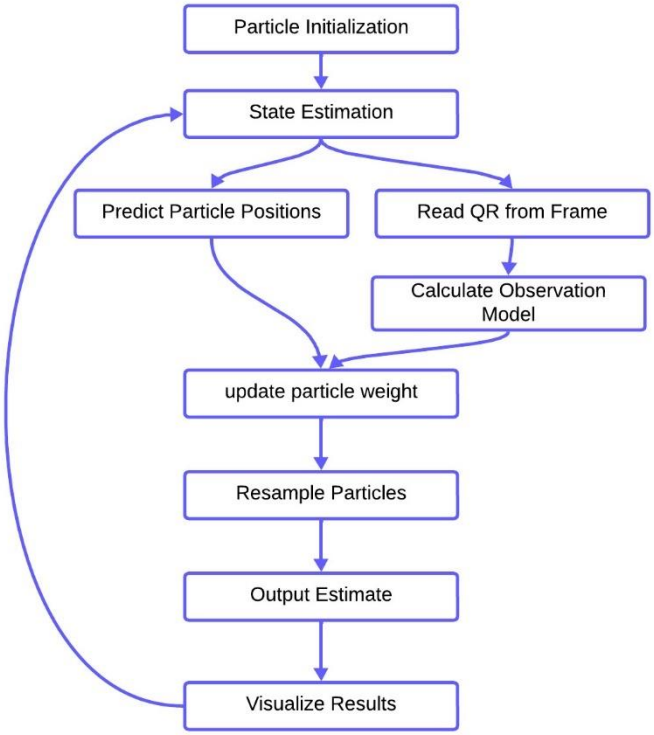

Figure 2.2: the Flow char of the particle fitter algorithm

2.2.12 **QR Algorithm**

Pyzbar is a Python library that is widely used for decoding barcodes and QR codes from images and video streams. It is a versatile tool that can handle various types of barcodes, including EAN, UPC, Code 128, and QR codes, among others. The library is built on top of the ZBar C library, providing a simple and effective interface for integrating barcode detection and decoding functionalities into Python applications.

Pyzbar operates by taking an image as input and processing it to detect and decode any barcodes present. The process begins with image acquisition, where the image can be obtained from a file, camera feed, or any other source that provides a compatible image format. Once the image is acquired, the following steps are performed [12]:

1- Image Conversion: The image is converted into a format suitable for barcode detection. Typically, this involves converting the image to grayscale, as barcode detection algorithms work more efficiently on single-channel images. Pyzbar uses the Python Imaging Library (PIL) or its fork, Pillow, to handle image conversion and manipulation.
2- Barcode Detection: Pyzbar scans the grayscale image for barcode patterns. The underlying ZBar library uses edge detection and pattern recognition techniques to identify regions in the image that resemble barcodes or QR codes. It employs a combination of thresholding, edge detection, and morphological operations to isolate potential barcode regions.
3- Decoding: Once a barcode is detected, Pyzbar attempts to decode it. The decoding process involves interpreting the patterns of bars and spaces (or modules in the case of QR codes) to extract the encoded information. Pyzbar supports a wide range of barcode formats and can decode multiple barcodes in a single image.
4- Output: The decoded information is returned as a list of objects, each containing the data encoded in the barcode, the barcode type, and the coordinates of the barcode's bounding box. This information can be used for various applications, such as inventory management, product lookup, or any other application requiring barcode scanning.

The following Figure 2.3 shows the Proposed Method's Flowchart

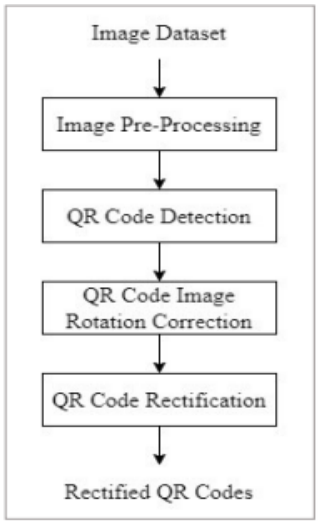

Figure 2.3: shows the Proposed Method's Flowchart [12]

2.3 Literature Review

Nowadays, in many applications, robots track and follow a person. As the robot is an innovative mobile robot capable of tracking its owner, bypassing obstacles, and avoiding tasks simultaneously, using the latest algorithms and image processing techniques, a mobile people-tracking robot will be implemented that will be able to follow the target person in practical and complex environments. Many studies and projects deal with and apply tracking technology.

2.3.1 Smart shopping cart for people with disabilities or limited mobility

The previous project was based on the idea of creating a shopping basket that follows its owner through a Pixie camera that is used to identify its owner, follow his/her movements throughout the shopping process, and avoid obstacles using ultrasonic sensors. A motor is used to move the robot in the right direction, and an alarm is used if the robot cannot go anywhere [13].

Naturally, our project is different and its goals are completely different, as different and advanced technologies will be used and it will be a robot whose work depends mainly on image processing and computer vision techniques.

The following Figure 2.4 shows the system structure

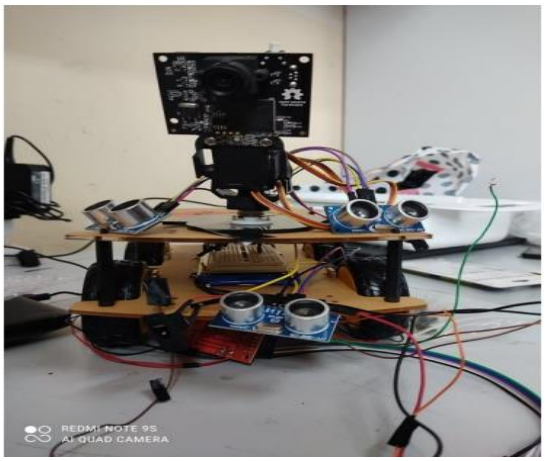

Figure 2.4: Robot structure [13]

2.3.2 A Human-Tracking Robot Using Ultra Wideband Technology

This project was implemented using Ultra-wideband (UWB) technologies that work to save energy and accuracy, prevent multi-path interference, and provide high security, in addition to reducing the complexity of the system. The person is identified by determining his position and direction.

UWB is a radio technology that uses surface power levels for short-range, high-bandwidth communications over a large portion of the radio spectrum. UWB technology is increasingly integrated into consumer electronics devices for applications such as secure keyless entry, real-time location tracking, and enhanced augmented reality experiences. For example, newer smartphone models are now equipped with UWB, which provides precise location services and seamless point-to-point data transfer.

Figure 2.5 shows the system structure

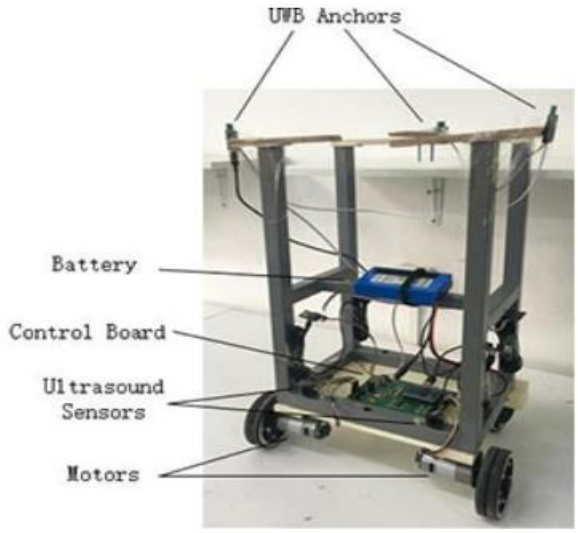

Figure 2.5: System architecture [14]

The hardware of the project includes "two wheels on two sides, with an additional wheel for directional movement. The sensors include the UWB anchors and ultrasound ones. There is a control board connecting all the sensors and computing the real-time motion signals to control the movement of the two wheels by the motors" [14].

2.3.3 Autonomous Person-Specific Following Robot

This robot is designed to help residents of nursing homes or patients in healthcare facilities by integrating the latest facial recognition technologies with tools for tracking unknown people. They used the sequential nearest neighbor algorithm, which is a mathematical method used to address a variety of classification tasks, and it is capable of dealing with bad or no lighting situations.

Chan et al tested their user-following method in a series of experiments in which the Aether robot had to identify, track, and follow users within five different scenarios. They monitored the position of the robot and that of people in its surroundings using a motion capture system called Vicon. The initial tests conducted by the researchers yielded highly promising results, with the new technique outperforming the existing face recognition and user-tracking tools it was compared to [15].

Figure 2.6 shows the system structure

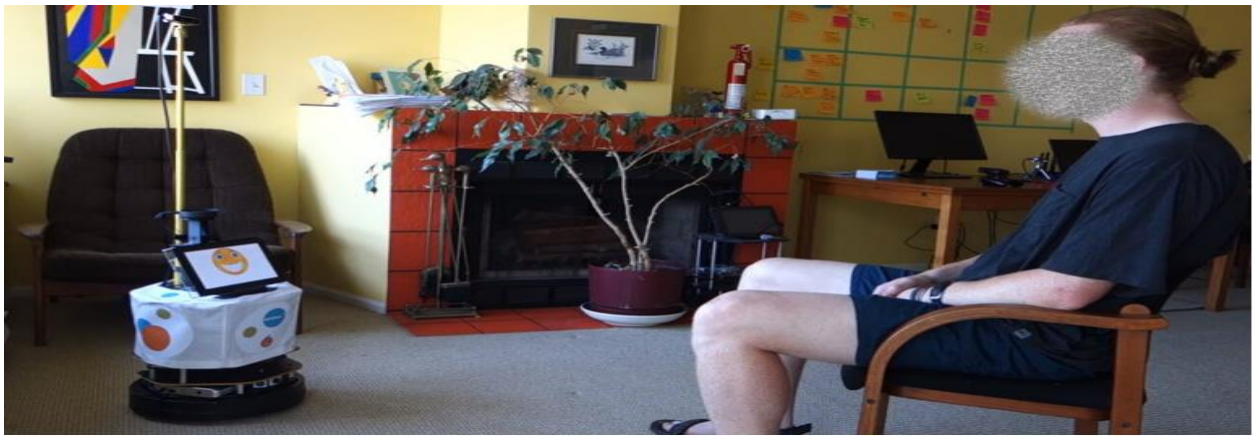

Figure 2.6: Robot structure [15]

Chapter 3: Design

3.1 Overview

This chapter will cover the overall design of the system and the integration of its components, showing the block diagram and schematic diagram, as well as details about the algorithms used.

3.2 Design Options

By comparing the available components and evaluating different hardware and software choices, the aim is to identify the most suitable components that align with the project requirements and objectives

3.2.1 Hardware Components Options

3.2.1.1 Processing Unit

The processing unit is the component that drives the system's functionality. It receives and processes sensor data, executes algorithms, and controls system components. When comparing and evaluating two choices, a Raspberry Pi 4 and a laptop, the selection of the processor depends on specific characteristics shown in Table 3.1.

Table 3.1: Comparison between Laptop and Raspberry Pi 4 and NVIDIA Jetson Nano

Feature	Laptop	Raspberry Pi	NVIDIA Jetson Nano
Performance	High processing power and memory	Lower processing power and memory	High processing power and memory
Size and Portability	Larger and less portable	Small, lightweight, and highly portable	Slightly larger and less portable
Cost	More expensive	More affordable	More expensive
Power Consumption	Higher power consumption	Very low power consumption	low power consumption

Operating System	Full-fledged OS (Windows, macOS, Linux)	Variety of OS (Raspbian, other Linux-based OS)	Ubuntu-based JetPack SDK
Connectivity	Built-in peripherals; wide range of ports and options	Requires external peripherals; fewer connectivity options	More USB 3.0 ports, HDMI 2.0, DisplayPort, and an M.2 slot for wireless expansion.

Initially, the Raspberry Pi was chosen as the processing unit, but we faced some problems, the most important of which was: extreme difficulty in the process of downloading the Kinect driver on the Raspberry Pi. After several attempts, we decided to move to the laptop to take advantage of the features of the Kinect camera and the sensors inside it, such as the depth sensor.

3.2.1.2 Obstacle Avoidance Sensor

There are several sensors used to measure distances that have shown their efficiency, but the comparison was between the most efficient as shown in Table 3.2.

Table 3.2: Comparison between LIDAR URG-04LX Sensor and HC-SR04 Ultrasonic Sensor

Feature	LIDAR URG-04LX Sensor	HC-SR04 Ultrasonic Sensor
Technology	Laser-based LIDAR	Ultrasonic (sound waves)
Primary Use	Robotics, industrial applications	Simple distance measurement, obstacle avoidance
Depth Sensing Range	0.06 - 4.0 meters	2 cm - 4 meters
Accuracy	Up to 10 mm (1 cm)	Varies, typically around 3 mm
Field of View (FoV)	240° (horizontal)	Narrow, conical beam (typically ~15°)
Resolution	High angular resolution (up to 0.36°)	Depending on pulse frequency, usually less detailed

Frame Rate	10 Hz	Dependent on programming, usually lower
Data Output	Distance values for the scanned area	Single distance measurement
Integration	USB/Serial interface, ROS compatible	Simple digital I/O, widely supported on microcontrollers
Size and Weight	Compact and lightweight	Very compact and lightweight
Cost	Generally, more expensive	Very affordable
Environmental Impact	Effective in various lighting conditions	May be affected by temperature and humidity
Power Consumption	Moderate	LOW

Based on the previous comparison, HC-SR04 will be chosen because of its breadth of scope, the accuracy of its results, and its lack of influence on environmental factors.

3.3.1.3 Mobile robots

There are many types of robots that we can use, but we took into account the ones available to us that could be most useful and achieve the goal as shown in Table 3.3.

Table 3.3: Comparison between Mobile Robot MP-500 and Kabuki Robot - Turtlebot2

Feature	Mobile Robot MP-500	Kabuki Robot (TurtleBot2)
Manufacturer	Adept Technology, Inc.	Willow Garage (originally), continued by Clearpath Robotics
Primary Use	Industrial applications, material handling	Research, education, and hobbyist applications
Platform Type	Autonomous Mobile Robot (AMR)	Open-source robotics platform
Payload Capacity	Up to 500 kg	Up to 5 kg
Dimensions	Approximately 800 x 800 x 350 mm	354 mm diameter, 390 mm height
Weight	Around 90 kg	6.3 kg (without payload)
Navigation	Laser-based SLAM, advanced navigation software	Basic navigation, supported by ROS
Sensors	LIDAR, ultrasonic, bumper sensors	Kinect sensor (for TurtleBot2), gyroscope, wheel encoders
Battery Life	Typically, 8 hours	2-3 hours
Speed	Up to 1.5 m/s	Up to 0.65 m/s
Software Integration	Proprietary software, integration with industrial systems	ROS (Robot Operating System) compatible

Kobuki was selected as the mobile robot platform for the project. It provides a Customizable hardware base with integrated sensors such as cameras, laser scanners, and inertial Units of measurement. Kobuki's compatibility with ROS allows for Various software libraries and algorithms to enable mapping, Localization, and path planning. We considered other options such as Mobile Robot MP-500, but Kobuki was the best fit for our project needs and goals. The Kobuki shown in Figure 3.1

Figure 3.1: Kobuki Robot - Turtlebot2

3.2.1.4 Camera

We have two options to choose from, which are Kinect and Pixy (CMUcam) Camera, and here is a comparison between them as shown in Table 3.4.

Table 3.4: Comparison between Kinect camera and Pixy (CMUcam) Camera

Feature/Aspect	Kinect XBOX360	Pixy (CMUcam) Camera
Primary Use	Motion sensing, depth perception, 3D scanning, gaming, and interaction.	Object recognition and tracking, robotics applications.
Technology	Uses an RGB camera, depth sensor, and infrared projector to map space in 3D.	Uses a camera sensor to detect objects based on color.
Resolution	Higher resolution for depth and image data.	Lower resolution, optimized for object detection.
Processing	More complex processing often requires a connected computer.	Simple, onboard processing for color-based object detection.
Ease of Use	Requires setup and calibration, more complex to use.	Designed for plug-and-play simplicity.

Flexibility	Highly versatile for different applications including full-body tracking.	Primarily focused on color-based object detection and tracking.
Cost	Generally, more expensive due to its advanced technology.	Relatively cheaper, designed for cost-effective robotics projects.
Integration	is Commonly used with gaming consoles, PCs, and complex robotics.	Easily integrates with microcontrollers like Arduino for hobbyist projects.
Field of View	A wide field of view is suitable for room-scale tracking.	The narrower field of view focused on detecting specific objects.
Software Support	Strong support in gaming and 3D modeling software.	Focused on robotics and educational programming.
Target Audience	Gamers, developers, and researchers in 3D modeling and interactive systems.	Hobbyists, educators, and robotics enthusiasts.

The Kinect camera was chosen for use because it is an RGB camera, which allows us to obtain images with a high resolution of up to 640 x 480 pixels, which enables us to obtain accurate results. It also contains a depth sensor that makes it easier for us to carry out the process of protecting the owner by maintaining a distance between the robot. And the owner.

3.2.1.5 Depth Sensor

A depth sensor is used to measure the distance between the sensor and objects in its environment. This technology enables the creation of three-dimensional representations by capturing depth information and provides simultaneous localization and mapping capabilities. It automatically detects any nearby object and measures the distance to it on the go. These features allow devices to move autonomously by making real-time decisions [16]. There are two available options for depth sensors to choose from, as shown in Table 3.5

Table 3.5: Comparison between Kinect Sensor and LIDAR URG-04LX Sensor

Feature	Kinect Sensor	LIDAR URG-04LX Sensor
Technology	Structured light / Time-of-Flight	Laser-based LIDAR
Primary Use	Gaming, entertainment, and some robotics applications	Robotics, industrial applications
Depth Sensing Range	0.5 m to 4.5 meters	4.0 meters
Accuracy	Varies with distance and conditions; typically around 1-3 cm	Up to 10 mm (1 cm)
Field of View (FoV)	Horizontal: 57°, Vertical: 43°	240° (horizontal)
Resolution	Depth map resolution of 640 x 480 pixels	Distance measurements at up to 0.36° angular resolution
Frame Rate	Up to 30 FPS	10 Hz
Data Output	Depth map, RGB image	Distance values for the scanned area
Integration	USB interface, compatible with various platforms and SDKs	USB/Serial interface, ROS compatible
Size and Weight	Relatively larger and heavier	Compact and lightweight
Cost	Relatively affordable	Generally, more expensive

Kinect was chosen because it can detect from 0.5m to 4.5 m, covering what is needed for mapping and navigation. It's not too expensive, and it has good accuracy. Moreover, as shown in Figure 3.2, the Kinect sensor captures RGB data using an IR depth sensor to enable depth measurement. In this process, the emitter releases infrared light beams, and the depth sensor reads the reflections of these beams and then translates it into depth information, measuring the distance between an object and the sensor

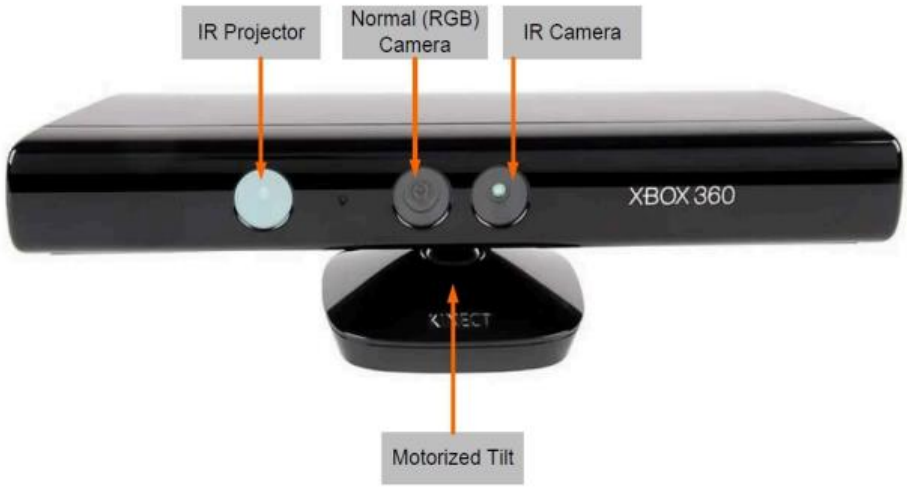

Figure 3.2: Kinect camera [17]

This ability allows the sensor to maintain a certain distance between itself and the owner, thus protecting the owner from injuries and collisions when stopping suddenly, as when this distance is exceeded, the robot stops.

3.2.2 Software Components Options

3.2.2.1 Robot Operating System

There are several other options for robot operating system frameworks that can be considered for the project, including:

1- **ROS Noetic**: ROS Noetic is a framework and toolset designed to develop robotic software. It supports component-based architecture and programming in various languages [18].

2- **MRPT**: is a collection of C++ libraries and algorithms for mobile robotics applications [19]. The differences between MRPT and ROS Noetic are shown in Table 3.6

Table 3.6: Comparison between ROS Noetic and MRPT

Feature	ROS Noetic	MRPT (Mobile Robot Programming Toolkit)
Platform	Robot Operating System (ROS)	C++ library
Primary Use	Robot development, research, deployment	Robotics research, SLAM, navigation, computer vision
Language	Python and C++	C++
Supported Systems	Ubuntu 20.04 LTS, Debian	Cross-platform (Windows, Linux, macOS)
Communication	ROS topics, services, actions	Custom data structures and serialization
SLAM Capabilities	Extensive support via packages (e.g., Gmapping)	Built-in SLAM support (e.g., Kalman filters)
Navigation	Navigation stack (move_base, AMCL)	Navigation modules
Sensor Integration	Extensive (LiDAR, cameras, IMUs)	Various sensors, more custom integration needed
Visualization	Rviz, rqt	MRPT apps (mrpt::gui, mrpt::opengl)
Community Support	Large, active community	Smaller community, good documentation
Ease of Use	Steeper learning curve	Easier C++ integration

Deployment	Nodes and launch files	Direct C++ project integration
Compatibility	ROS2 compatibility	Independent, can be used with ROS
Development	Frequent updates	Less frequent updates

There are two programming languages available for development within the ROS noetic framework: C++ and Python. C++ is a powerful and efficient programming language widely used in robotics, Python was chosen because it offers a wide range of libraries and tools for development and prototyping

3.2.2.2 Open CV

OpenCV is very useful for visualization and analysis purposes. It is primarily responsible for image preprocessing and enhancing the quality of images recorded by the Kinect camera, this could involve reducing noise, adjusting contrast, and optimizing the images for analysis. The library provides algorithms for image segmentation, pattern recognition, and feature extraction [18].

3.2.2.3 Particle Filter Algorithm

In this project, we use a particle filter algorithm to achieve accurate and reliable tracking of the owner's location. The particle filter, a powerful tool for state estimation in dynamical systems, works by maintaining a set of particles, each of which represents a possible state for the location of a QR code. As the robot moves, the algorithm predicts the new locations of these particles based on the robot's motion model. Feedback from the Kinect camera is then used to update the particle weights, with particles most consistent with the observed QR code position being assigned higher weights, ensuring robust tracking even in complex and dynamic environments.

3.2.2.4 pyzbar library

In this project, we employ the pyzbar library to handle the crucial task of QR code recognition. Pyzbar is a versatile and efficient library capable of decoding various types of barcodes and QR codes from images. By integrating pyzbar with our Kinect camera, we capture real-time images of the environment, which are then processed to detect the QR code carried by the owner. The library

decodes the QR code's content, providing essential information about the owner's position. This decoded information is used as input for our particle filter algorithm, facilitating accurate tracking. The ease of integration and the reliability of pyzbar in decoding QR codes in different lighting conditions and orientations make it an ideal choice for our project, ensuring consistent and precise detection that is critical for the robot's ability to follow its owner seamlessly.

3.3 General Block Diagram

The Raspberry Pi 4 was previously scheduled to be adopted as the primary processing unit in the system, and as shown in Figure 3.3, an overview of the main components that make up the system. The Raspberry Pi is the core component of the system, as it receives data, processes it, and then issues commands. Data enters the system through the action camera, which can be opened for live streaming to identify the owner (using the owner's QR code) in real time to maintain efficiency and speed. Meanwhile, ultrasonic sensors will avoid obstacles and maintain a safe distance between the robot and the owner. The power unit provides the power needed for the system to operate efficiently.

However, we decided to use a laptop instead of the Raspberry Pi, and a Kinect instead of an action camera due to its depth sensor and the incompatibility of the Kinect with the Raspberry Pi. Therefore, we switched to a laptop before completing the implementation of the ultrasonic sensors on the Raspberry Pi.

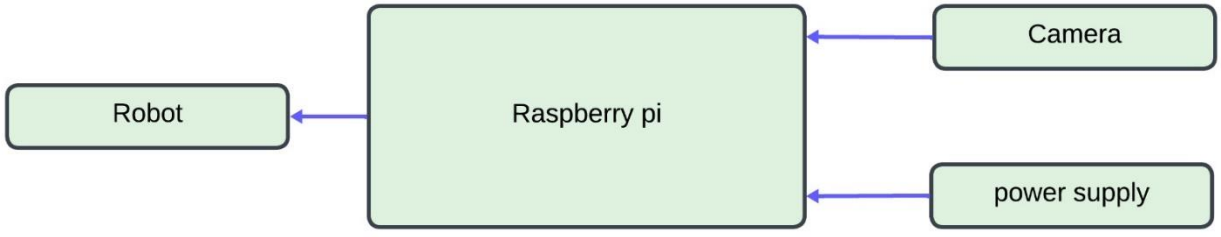

Figure 3.3: Basic system components using Raspberry Pi

The laptop was then adopted as the primary processing unit for the system, and as shown in Figure 3.4, an overview of the main components that currently make up the system. The computer is the basic element in the system, as it receives data, processes it, and then issues commands. Data enters the system through a Kinect camera, which can be opened for live streaming to identify the owner (using the owner's QR code) in real time to maintain efficiency and speed. Meanwhile, the camera has a depth sensor, which we will use to maintain a safe distance between the robot and the owner, while we will use ultrasonic sensors to avoid obstacles. The power unit provides the power needed for the system to operate efficiently.

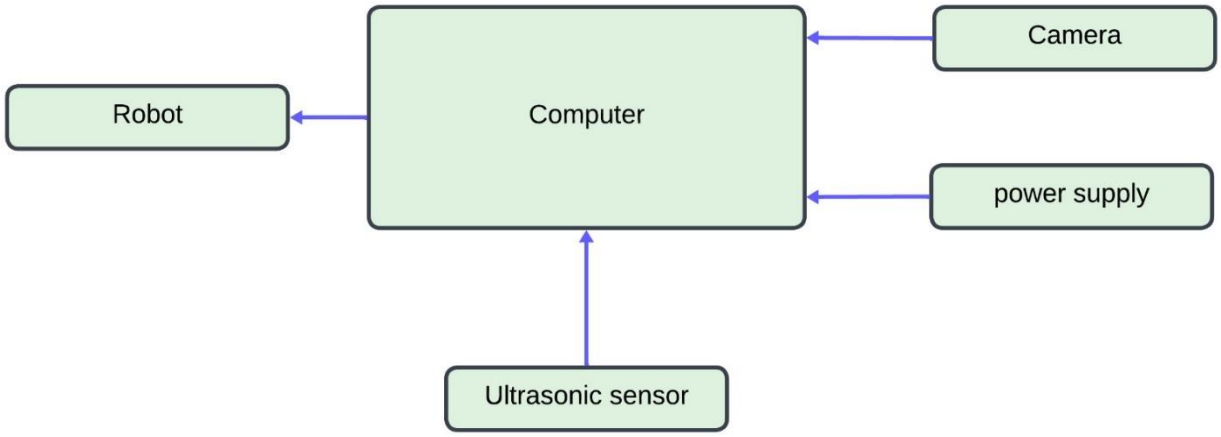

Figure 3.4: Basic system components using Laptop

3.4 General Flowchart

The following section describes the steps the system will follow, as shown in Figure 3.5 below.

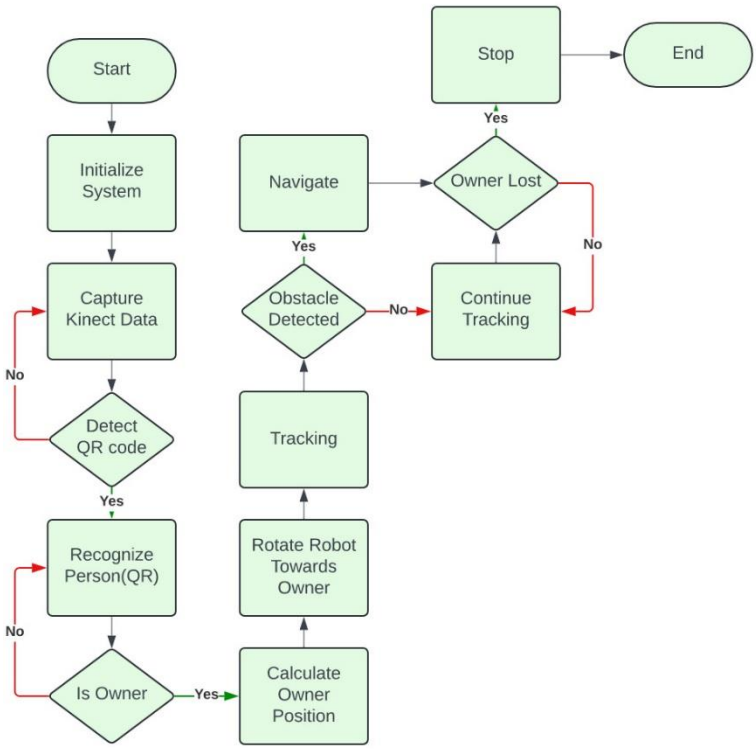

Figure 3.5: General Flowchart

Initially, the robot is activated. Then, the system is configured, and a live-streaming camera is turned on. The live stream is continuously monitored to check for any QR codes within the camera's frame or range using QR code detection algorithms (ZBar library). If no QR code is detected in the frame, the camera keeps running, attempting to capture useful data. If a QR code is found, the robot checks whether it is the owner's QR code. If not, the camera continues to wait until the owner's QR code appears. If the person is the owner, their location and distance from the robot are calculated using the depth sensor in the Kinect camera. The robot then rotates to face its owner and follows them using a particle filter algorithm. Meanwhile, the ultrasonic sensor scans the surrounding area within its range to ensure the path is clear. If no obstacles are found, the robot continues to follow the owner. If obstacles are encountered, it navigates around them and checks

if the owner is lost. If the owner is not lost, the robot continues to follow them. However, if the owner is lost, the robot stops, and the program ends.

3.5 System Schematics

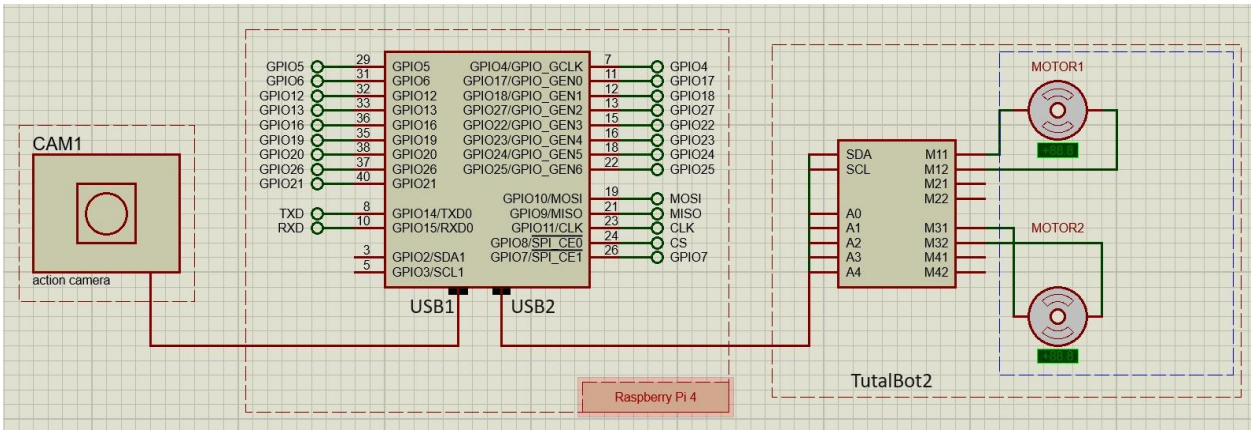

Figure 3.6: System Schematics using Raspberry Pi 4

As shown in Figure 3.6, using the Proteus software, we successfully identified the circuit connection for our project (before modification). Simply put, the system inputs are provided through an action camera. We connect the action camera to our Raspberry Pi 4 via a USB port and supply the camera with the necessary power to keep the live feed active throughout the system's operation. The camera must remain active to detect our QR code (carried by the designated person). The system outputs are configured based on specific conditions, which generally cause the robot to move in certain directions according to these conditions.

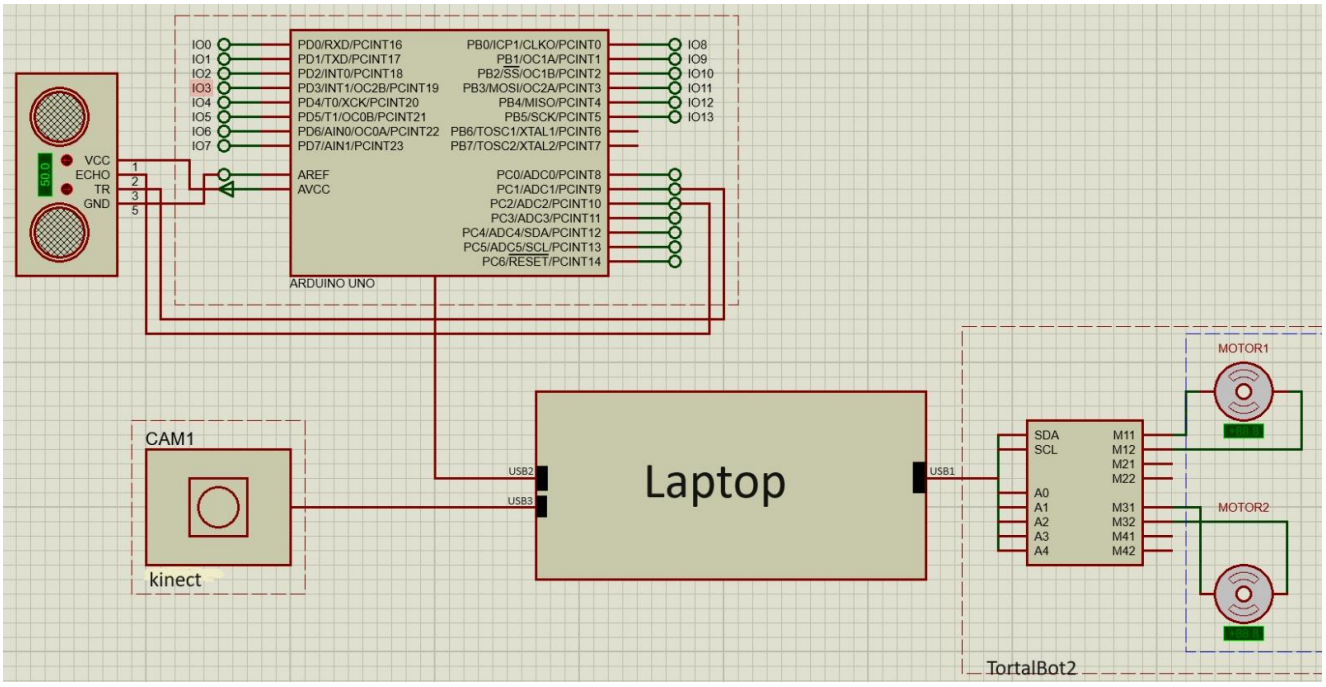

Figure 3.7: System Schematics using a laptop

As we can see in Figure 3.7, using Proteus software, we successfully identified the circuit connection for our project. Simply, the system inputs are provided through a Kinect camera. We connect the Kinect camera to our laptop via a USB port and supply the necessary power to the camera to keep the live feed active throughout the system's operation. The camera must remain active to detect our QR code (carried by the person in question). The system outputs are configured based on specific conditions, which generally result in the robot moving in certain directions according to these conditions. After detecting our QR code, the robot enters a tracking phase where it follows the QR code: if the person carrying the QR code moves to the right, the robot moves to the right; if the person moves to the left, the robot moves to the left; and if the person moves forward, the robot follows forward, and so on. However, if the QR code is not found, the robot remains in place until it detects the QR code again. In case an obstacle prevents the robot from moving, the ultrasonic sensor is used to avoid this obstacle (avoiding collision at least). In the figure, we replaced the robot with two DC motors connected to the driver and then to the laptop.

Chapter 4: Implementation

4.1 Overview

This chapter describes the implementation part of the project in more detail. It dives deep into the different hardware components of the system and its software with all of its modules.

4.2 Hardware Implementation

4.2.1 Laptop

The laptop is the main component in this project, which connects the other elements

1- The TurtleBot is connected to the laptop via a USB cable as shown in Figure 4.1.

Figure 4.1: System components

2- The laptop is connected to the Kinect Sensor via a USB cable, and it receives power through an adapter utilizing the TurtleBot's 12V/5A cable, as shown in Figure 4.2

Figure 4.2: Connect Kinect sensor to TurtleBot

4.2.2 Ultrasonic Sensor with Arduino

The ultrasonic sensor is used to overcome obstacles that the camera cannot see

1- The ultrasonic sensor is connected to the Arduino as shown in Figure 4.3

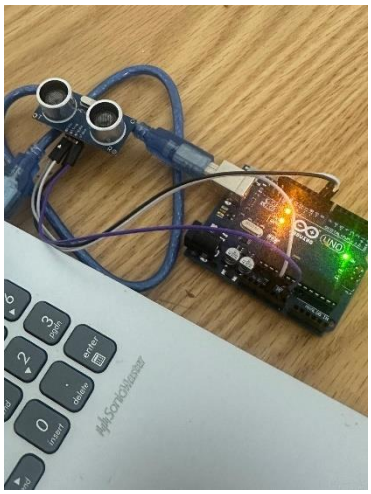

Figure 4.3: Connect Ultrasonic sensor to Arduino

Connect the Ultrasonic Sensor to the Arduino

- VCC to Arduino 5V
- GND to Arduino GND
- Trig to Arduino digital pin 9

- Echo to Arduino digital pin 10

4.3 Software Implementation

4.3.1 Arduino environment implementation

This process is intended to prepare the Arduino for work. It starts by downloading the latest version from the original page website [20]. When the download is finished, we get the decompressed file. After that, we double-click on the folder to open it, it is ready to use, and building our system in the Arduino software environment is prepared to use.

4.3.2 Download USB drivers

We need USB drivers for the FTDI chip on the board to connect our Arduino board to the computer. In addition, we need to install the USB drivers before plugging in the Arduino for the first time. This demands downloading the latest version from this page site [21]. After that, we need to select the port and select the microcontroller type " Arduino Mega 2560".

4.3.2 Detection algorithm

4.3.2.1 OpenCV Setup

We need to install the OpenCV library to capture a QR code frame from the video stream and read this QR. To install it we will enter the following commands in the Visual Studio terminal:

```
Sudo apt-get update
Sudo apt-get install python3-opencv
Pip3 install opencv-contrib-python
```

4.3.2.2 ZBar Setup

We need to install the ZBar library to decode the QR code and convert it to the respective information. To install it we will enter the following commands into the Visual Studio terminal:

```
sudo apt-get install libzbar-dev
sudo pip install zbar
```

4.3.3 Tracking algorithm

We need to install the NumPy library to use it in the tracking process. To install it, we will enter the following commands in the Visual Studio terminal

```
pip install numpy
```

4.3.4 Installing Ubuntu Mate 20.04 Operating System

Ubuntu 20.04 Mate, ensuring stability and compatibility with ROS noetic [22].

4.3.5 Installing ROS Noetic

The system runs on ROS Noetic. It contains all the necessary packages like ROS_enviroment and catkin to operate the TurtleBot robot and Kinect sensor [23].

installation commands:

1- The following line of command will install the latest ROS Noetic on Ubuntu-mate 20.04

```
$Wgethttps://raw.githubusercontent.com/qboticslabs/ros_install_noetic/
master/ros_install_noetic.sh && chmod +x ./ros_install_noetic.sh && ./ros_install_noetic.sh
```

```
$ sudo apt install ros-noetic-desktop-full
```

2- Environment setup

```
$ echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
```

4.3.6 TurtleBot Installation

This involves installing essential TurtleBot packages that didn't come with the basic ROS Noetic installation. These additional packages include things like ***turtlebot_apps, launch files, turtlebot_viz***, and ***turtlebot_bringup***, they provide extra functionalities and tools that enhance the capabilities of TurtleBot for various tasks and applications [24]. The following command is used to get the all required packages:

```
$ git clone https://github.com/hanruihua/Turtlebot_on_noetic.git
```

4.3.7 Obstacles Avoidance

Ultrasonic was used to overcome obstacles that the camera does not see, by measuring the distance between it and the object opposite it and sending a message to ROS, and after the process of identifying the obstacles, they are avoided.

4.3.7.1 Install and Configure rosserial

1- Installing libraries that provide tools and applications for interaction between ROS and Arduino and allow communication via serial interfaces between ROS and Arduino, by writing the following commands on the terminal:

```
sudo apt-get install ros-noetic-rosserial-arduino
sudo apt-get install ros-noetic-rosserial
```

1- **Install the** rosserial **library on the Arduino:**

- Open the Arduino IDE.
- Go to Sketch -> Include Library -> Manage Libraries.

- Search for rosserial and install the rosserial library.

2- **Upload the Arduino sketch to your Arduino board**.

4.3.7.3 Run rosserial to Communicate with the Arduino

1- Connect the Arduino to your computer via USB**.**

4.3.7.4 Run rosserial_python Node Again

rosrun rosserial_python serial_node.py _port:=/dev/ttyACM0 _baud:=57600

4.4 Implementation Issus

4.4.1 Raspberry Pi Issus

The Raspberry Pi was supposed to be used as a processing unit, but we encountered many problems, including:

1- The stage of downloading Raspbian as an operating system for the Raspberry Pi

At this stage, we faced many problems as it was difficult to find a driver for the robot because most of the projects implemented on this robot were using the ROS system. After a long search process, we found a library that could be used as a driver library for the robot by taking advantage of the move command as it gives each wheel Speed.

2- After merging the tracking code and QR code with the driver, we faced the problem of inaccurate results in determining QR locations; This is due to the inability to control speed and direction simultaneously because direction depends on speed. To solve this problem, we divided the camera range into 5 sections and printed an average for each location. Each location had a different warp speed. Through experimentation, we learned where each location led (right, left, forward), and created an if statement for each location. Figure 4.5 shows the result of the Raspberry Pi tracking process.

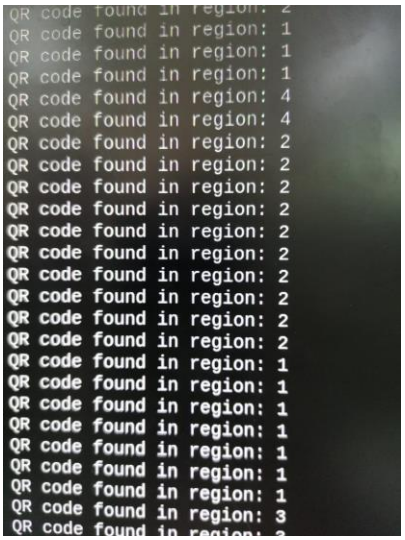

Figure 4.4: Result using Raspberry Pi

3- The inability to operate the Kinect camera on the Raspberry Pi. We tried to operate it for 3 weeks, but all attempts failed. We prefer using the Kinect because it contains sensors that protect the owner by leaving a certain distance between the robot and the owner.

Therefore, we switched from using the Raspberry Pi to a laptop. First, we tried running Diver, which was previously used on the Raspberry Pi, but it did not work on the computer, and the solution was to use ROS.

4.4.2 Kinect Camera Issus

1- **Kinect Camera Identification:**

We faced issues with the Kinect connection, it's using a different ID than the default one, causing another device to connect to the USB port instead. Resolved this by adjusting the device's default ID to match the designated Kinect ID within the OpenNI file.

```
<arg name="device_id" default="#2" />
```

sets the device ID for the camera. The value is set to "#2", which suggests using the second device found. It can be specified in various formats (serial number, bus, and address) to identify the camera device uniquely.

2- Speed adjustment for QR detector:

While moving, the robot's movement was fast. Therefore, it is difficult for the robot to detect the QR. This problem was solved by reducing the robot's speed by changing the maximum speed x (Max_vel_x =0.6) of the robot to 0.1 m/s.

3- Camera-Robot Synchronization:

Faced delays between camera input and robot movement. Successfully fixed the synchronization gap, improving real-time coordination by adding this line of code.

```
if rospy.Time. now() - self.last_image_time < self.min_interval:
    return
self.last_image_time = rospy.Time.now()
```

4.4.2 Open CV Issus

While attempting to install the opencv-python package, a FileNotFoundError was encountered. The error message indicated that the file config-3.py was missing from the _skbuild/linux-aarch64-3.11/cmake-install/python/cv2/ directory. This failed to build the wheel for opencv-python.

To fix this problem, we followed these steps:

1- Uninstall the existing opencv-python package:

```
pip uninstall opencv-python
```

2- Reinstall opencv-python without using cached files:

```
pip install --no-cache-dir opencv-python
```

4.4.2 Tracking Algorithm Issus

Running the Python script projectcod.py results in an import error for NumPy and OpenCV due to a missing shared object file (libopenblas.so.0), The following error message appeared:

```
ImportError: libopenblas.so.0: cannot open shared object file: No such file or directory
```

To fix this problem, we followed these steps:

1- Install NumPy and OpenBLAS:

```
pip install numpy
sudo apt-get install libopenblas-dev
```

2- Activate the Virtual Environment:

```
source /home/zaina/project1/venv1/bin/activate
```

3- Set Environment Variable:

```
export LD_LIBRARY_PATH=/usr/lib/openblas:$LD_LIBRARY_PATH
```

4- Reinstall NumPy:

```
pip uninstall numpy
pip install numpy
```

4.4.2 System Issus

Continuous Movement Without QR Code Detection Problem

TurtleBot was moving continuously without waiting for the QR code to be detected. This was because the motion command was sent in a continuous loop without checking whether the QR code was detected.

We fixed this issue by creating a global variable qr_detected to track whether a QR code has been detected or not. TurtleBot's movement is then conditional on the value of this variable. The robot moves forward only when qr_detected is true.

Chapter 5: Testing and Validation

5.1 Overview

This chapter explains the project component testing methodology and displays the project system implementation outcomes.

5.2 ROS Installation

You can check the installed version by opening a terminal and running:

```
roscore
```

ROS is installed correctly if you see: started core service [/rosout]

5.3 Testing the TurtleBot

Ensure the TurtleBot is connected to the laptop using the cables and connectors. Check the power connection and ensure that the robot is receiving power. After running the following command, a sound will be emitted to indicate a power connection:

```
roslaunch turtlebot_bringup minimal.launch
```

5.4 Testing the Kinect

5.4.1 Default 3D sensor

Check the default 3D sensor of TurtleBot by printing an environment variable and confirming the output:

```
$ echo $TURTLEBOT_3D_SENSOR

# Output: Kinect
```

5.4.2 Test OpenNI Driver

Launch the OpenNI driver for ROS, initializing the OpenNI camera and making its data available for further processing in the ROS ecosystem.

```
$ roslaunch openni_launch openni.launch
```

5.4.3 Test Kinect Stream

Open a new terminal, and check the list of topics being published:

```
$ rostopic list
```

This command will display the RGB camera data being published on that topic in the terminal as shown in Figure 5.1:

```
$ rostopic echo /camera/rgb/image_raw
```

Figure 5.1: RGB camera data stream

5.4.4 Test Image Data

Using the following command will display a live video stream from the Kinect as shown in Figure 5.2:

```
rosrun image_view image_view image: =/camera/rgb/image_color
```

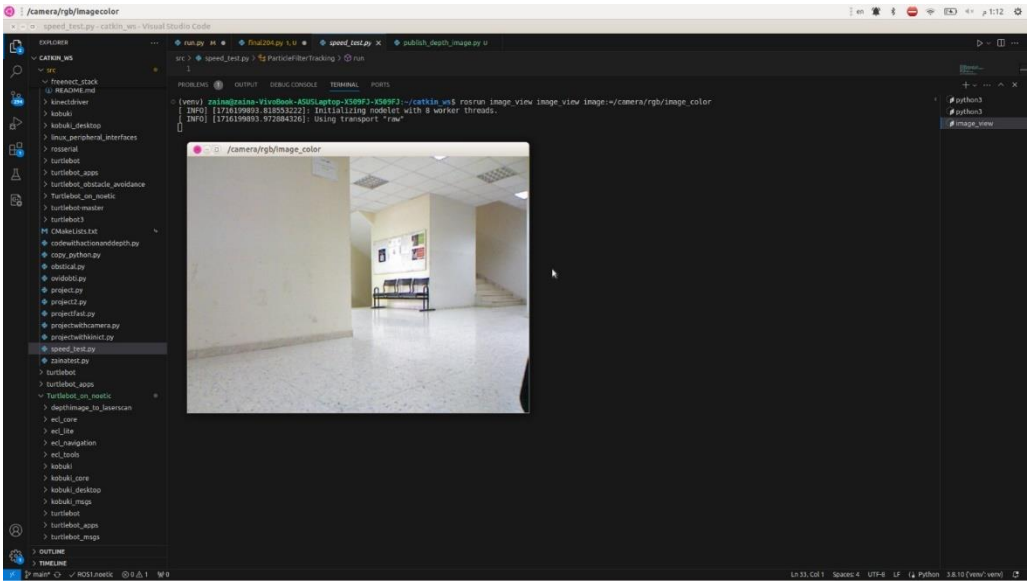

Figure 5.2: Image Data

5.4.5 Test Depth Data

This command test of depth data showed a visual representation of distance where objects closer to the Kinect appear dark as shown in Figure 5.3:

```
rosrun image_view image_view image:=/camera/depth/image
```

Figure 5.3: Test Kinect depth data

5.5 Test Obstacles Avoidance

Echo the ROS Topic (testing ultrasonic)

Check the topic /ultrasonic_distance for the sensor readings:

rostopic echo /ultrasonic_distance

Create a Directory for Logs:

mkdir -p ~/logs ```

The following Figure 5.4 shows the result of the implementation:

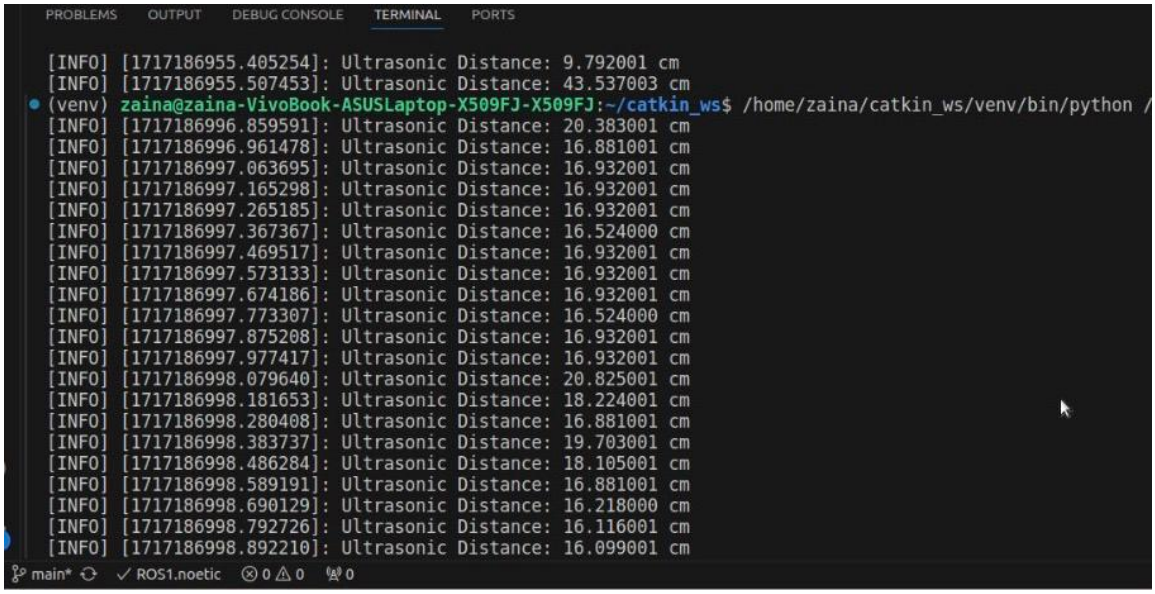

Figure 5.4: Execution result Ultrasonic code

5.6 Test Detection Algorithm

The Zbar library was used for the process of identifying the QR that the owner carries, and the following Figure 5.5 shows the result of the implementation, where the QR was identified.

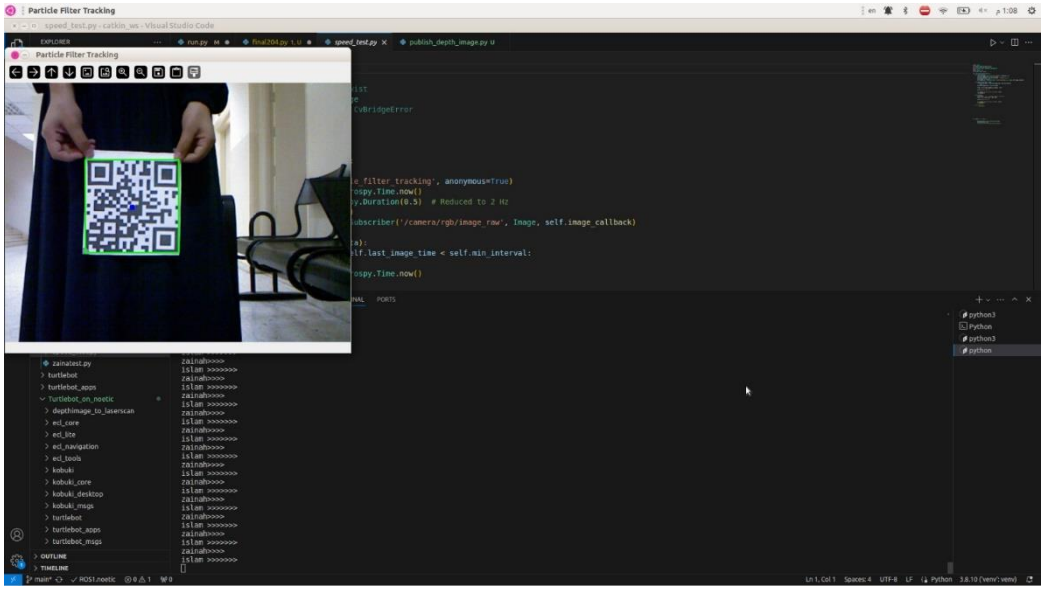

Figure 5.5: Result of using zbar library

5.7 Test Tracking Algorithm

The particle filter algorithm was used in the tracking process and its accuracy in the tracking process was verified, as the accuracy rate was approximately 90%, where a cross is placed in the middle and this represents the tracking process, and the following figure 5.6 shows the implementation result.

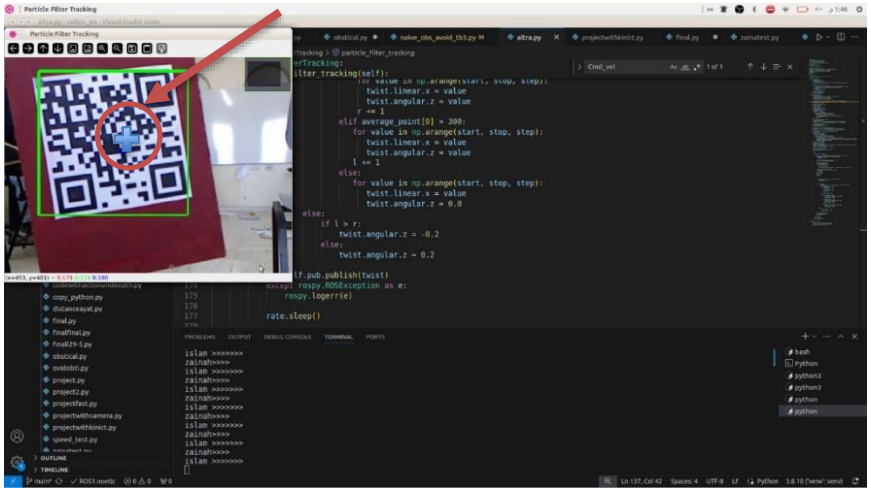

Figure 5.6: The result of using a particle filter

5.8 Overall System Testary

We tested the entire system in three stages: we tested the robot's movement during tracking and detection operations, we checked the sensors as they detected and crossed obstacles, and finally, we tested the overall stability and responsiveness of the system.

During the first phase, we tested the robot's ability to move while tracking, by observing how the robot interacts with QR, and whether it can follow it or not. In a different direction, and how quickly it responds to a change in the location of QR.

Therefore, the robot must first recognize the QR, and then begin the tracking process, and when the owner moves in its direction, the robot must change its direction, and if the robot finds an obstacle in its path, it must overcome it by turning and changing direction, and then completing the process Tracking.

Finally, we combined all the system components and modify all the code parts so that the system would function as one unit and achieve all the features that the system was designed to.

5.9 System Validation

In this section, we check the complete system component to make sure it meets the functional requirements under various test scenarios. These include:

Case 1: The robot moves in an environment crowded with people but there is only one person holding a QR code. It has difficulty recognizing QR when the light is insufficient at certain points.

This problem was solved by enlarging the size of the QR that the person was carrying, and thus the effect of light became somewhat less.

Case 2: The robot moved again and was able to detect the QR, but it did not recognize it accurately, so the robot’s movement was random and inaccurate.

This problem is solved in section 4.4.2.

Case 3: The robot is now moving in the same environment and can detect and track the QR, but when it encounters an obstacle, it does not cross it but collides with it.

This problem was overcome by using ultrasonics in the obstacle avoidance process.

Case 4: The robot moves again, recognizes the QR tracks it accurately, and changes its direction based on the direction in which the owner moves. When he encounters an obstacle in the road, he overcomes it. It also maintains a safe distance between itself and the owner, and when the robot approaches the owner at a distance less than this distance, it stops.

Chapter 6: Conclusion and Future Work

6.1 Conclusion

This project develops an autonomous robot that can track its owner using a QR code it carries. The system uses a Kinect camera to capture image and depth data, while the Pyzbar library recognizes and decodes the QR code to determine the owner's location. A particle filtering algorithm processes this data and accurately estimates the owner's position and movement. Ultrasonic sensors detect obstacles in real-time, allowing the robot to safely navigate around them. Kinect's depth sensor ensures that the robot maintains a safe and constant distance from the owner and stops when this distance is exceeded. This integrated approach enables robust and reliable following behavior and seamless interaction between the robot and its owner in dynamic environments.

6.2 Future work

In the future development of this project, several improvements can be implemented to improve system functionality, these considerations include:

1- Using machine learning models, so that the recognition process is more accurate without being affected by lighting conditions.

2- Enable the robot to recognize more than one person based on certain conditions specific to each person.

3- Enabling the robot’s ability to interact externally with its owner and in different circumstances

4- Using more modern sensors to overcome obstacles

5- Interact with the individual in a personal way and provide assistance based on his context and preferences.

Reference

[1] sensor Device, Pooja Gupta, April 2021". [Online].

[2] Arduino Microcontrollers, Arduino&tech, April 2021.[Online]

[3] UltrasonicDistanceSensorHCSR04, sparKfun, from: "https://www.sparkfun.com/products/15569". April 2021. [Online]

[4] How HC-SR04 Ultrasonic Sensor Works & Interface It With Arduino, Learn Electronics, February 2022.

[5] C.Rasche, "Computer Vision", *ResearchGate*, October 2019.

[6] F.Banda, "OVERVIEW OF IMAGE PROCESSING OVERVIEW OF IMAGE PROCESSING Contents", *ResearchGate*, December 2020.

[7] J. M. O'Kane, "A Gentle Introduction to ROS*", University of South Carolina*, April 24, 2018.

[8] N.Kumar, Z.Vsmossy, " Robot navigation with obstacle avoidance in the unknown environment", *Date accessed*, October 28, 2023.

[9] F.Porikli, A.Yilmaz, " Object Detection and Tracking", *ResearchGate*, January 2012

[10] C.K.Kumar, K.Rawal, "A Brief Study on Object Detection and Tracking", Journal of Physics, 2022

[11] A.Abdul, R.Abu Rumman, N.Feroza, I. Zahidul," A Particle Filter Based Visual Object Tracking: A Systematic Review of Current Trends and Research Challenges", *International Journal of Advanced Computer Science and Applications, Vol. 14*, No. 11, 2023.

[12] F.C. Ferano, J.K. Olajuwon, G.P. Kusuma, "QR Code Detection And Rectification Using Pyzbar And Perspective Transformation", *Jatit, Vol.100*. No 21,15th November 2022.

[13] A.Sonkrot, A.Taradeh,” Smart shopping cart for people with disabilities or limited mobility”, 2022

[14] T. FENG, Y, YU, L. WU, Y. BAI, Z. XIAO, A.Z. LU, “A Human-Tracking Robot Using Ultra Wideband Technology”, *IEEE Access, Vol.6*, publication July 25, 2018.

[15] W.P.Chan , S.radmard , Z.Q.Hew, J.Morris, E.Croft, “Autonomous Person-Specific Following Robot”, *Cornell University*, Vol.2, 11 Nov 2020

[16] terabee.com, Depth sensors: Precision & personal privacy, January 4, 2024.

[17] allaboutcircuits.com,Teardown Tuesday: Microsoft's Xbox 360 Kinect, December 3, 2023.

[18] wiki.ros.org, ROS Noetic Ninjemys, December 3, 2023.

[19] docs.mrpt.org, MRPT, May 22, 2023.

[20] Arduino site, November 2021. [Online].

[21] FTDI chip , November 2021. [Online].

[22] Ubuntu releases, Ubuntu MATE 20.04.6 LTS (Focal Fossa), October 20, 2023.

[23] ros.org, ROS, October 20, 2023.

[24] ros.org, Turtlebot Installation, October 20, 2023.