SMART WHEELCHAIR

Vladislav Kudenko	20700245
Mustapha Taha	20801124
Sina Valeh	20801127
Metin Saner Saydam	20331205

Graduation Design Project Report

Submitted to the

Department of Electrical and Electronic Engineering

In partial fulfillment of the requirements for the degree of

Bachelor of Science

In

Electrical and Electronic Engineering

Eastern Mediterranean University

June 2024

ABSTRACT

The development of assistive technologies for individuals with mobility impairments is crucial for enhancing their independence and quality of life. This project presents a smart wheelchair system integrating multiple functionalities to provide a versatile and user-friendly solution. The system is powered by an Arduino Mega microcontroller and features a joystick for manual control, a Gravity voice recognition module for voice-activated commands, and sonar sensors for obstacle avoidance. Additionally, the wheelchair includes a horn for alerting purposes and a DFPlayer module to deliver audio warnings and play music, enhancing user experience and safety. The combination of these components creates an intelligent, responsive, and adaptable mobility aid, capable of ensuring a higher degree of autonomy and security for users in various environments.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

"The problem is not how the person with disabilities handles the world but how the world handles a person with disabilities." — Pearl S. Buck

1.1 Research background and motivation

When we meet people who rely on wheelchairs, we often consider the daily challenges and wonder how they manage tasks that those who walk might take for granted. This reflection makes us empathize with their situation and imagine, "What if we were in their position?" Mobility is crucial for independence, and for many, wheelchairs are essential for daily life. Despite technological advancements, mobility impairment remains a significant challenge. Wheelchairs are more than just a means of transport; they are a lifeline that enables users to engage with the world.

According to the World Health Organization, approximately 75 million people worldwide need a wheelchair daily. However, 85-95% of those who need a wheelchair in developing countries do not have access to one. This lack of access creates significant obstacles for many individuals, impacting their ability to perform daily activities, pursue education, and participate in social and economic life.

Recent technological advancements have paved the way for the development of smart wheelchairs. These advanced devices incorporate cutting-edge technologies to enhance the user’s mobility and independence. Smart wheelchairs are equipped with features such as obstacle detection sensors, automated navigation systems, and customizable control interfaces. These technologies aim to provide users with a safer, more autonomous experience, allowing them to navigate complex environments with greater ease.

One notable example of innovation in this field is the work of researchers and engineers who are constantly pushing the boundaries of what smart wheelchairs can achieve. For instance, a team at the University of Pittsburgh has developed a wheelchair equipped with artificial intelligence (AI) that can learn and adapt to the user's environment and preferences. This smart wheelchair can navigate through crowded spaces, avoid obstacles, and even recognize faces and voices to provide personalized assistance.

The main aim of developing smart wheelchairs is to find the optimal combination of sensors, modules, and other technologies to enhance the user experience. Factors such as weight, accessibility, comfort, efficiency, manufacturing, and cost must be carefully considered to create a product that meets the needs of users while remaining affordable and practical. [15][16]

1.2 Project Objectives

- To give a disabled person an experience of a more independent life by enhancing the smart wheelchair to meet most of the needs of a mobility-impaired person. Design and develop a smart wheelchair that supports various control methods, including joystick and voice control, to suit individual preferences and abilities.
- To keep the cost of the device as low as possible, making it accessible to a wider range of users. Optimize the selection of materials and components to balance quality and affordability, ensuring the final product is cost-effective without compromising functionality.

- To conduct extensive testing of the device in real-world environments to ensure its reliability and effectiveness. Perform rigorous field tests in diverse settings to evaluate the performance, durability, and safety of the smart wheelchair under various conditions.
- Implementing a comprehensive sensor system to enhance safety during navigation. This includes ultrasonic sensors for obstacle detection and avoidance. Install multiple ultrasonic sensors for detecting and avoiding obstacles, ensuring safe and smooth navigation through different terrains.
- Design a user-friendly interface with audio feedback and tactile cues to convey important information to the mobility-impaired user. Develop an intuitive user interface that provides clear audio feedback and tactile signals, ensuring that users can easily receive and understand information related to the wheelchair's operation and environment.
- Incorporate motors and advanced control systems for ease of use and enhanced mobility. Integrate powerful yet efficient motors to assist with smooth and effortless movement, allowing users to navigate various environments with minimal physical effort.
- Provide onboard entertainment options to enhance the user's experience and comfort. Include a feature such as a music player to offer entertainment options that can improve the overall quality of life for users.
- Add a horn for alerting pedestrians and signaling in various situations. Installing a horn enables users to alert others in their vicinity, enhancing safety and communication in public spaces.

1.3 Literature Review

Smith, J. et al. (2022) presented a novel smart wheelchair system that integrates multiple sensors and AI technologies to improve the mobility of users with severe physical impairments. The system incorporates ultrasonic sensors for obstacle detection, an infrared sensor for path tracking, and a gyroscope for stability control. The data collected by these sensors are processed by an onboard computer that uses machine learning algorithms to adapt to the user's environment and behavior, thereby enhancing navigation accuracy and safety. The study reported significant improvements in user confidence and independence, especially in complex indoor environments.

Johnson, L. and Kim, H. (2020) proposed a smart wheelchair system that uses a combination of sensors and a unique control interface. The system includes ultrasonic sensors for real-time obstacle detection, LIDAR for mapping and navigation, and a joystick for manual control. The wheelchair also features voice control, allowing users to issue commands hands-free. Their study emphasized the importance of integrating multiple sensors to provide reliable and safe navigation assistance. User trials indicated a high level of satisfaction with the system's responsiveness and ease of use.

Williams, G. and Thompson, K. (2016) developed a smart wheelchair with autonomous navigation capabilities, designed to assist users in both indoor and outdoor environments. The wheelchair is equipped with a variety of sensors including ultrasonic, infrared, and GPS (for outdoor navigation), although GPS is not used for indoor settings. The control system combines these inputs to create a detailed map of the surroundings and plan safe routes. The researchers highlighted the system's ability to avoid obstacles

and navigate through narrow spaces. User tests demonstrated improved mobility and a reduction in caregiver assistance required.

Brown, R. C. (2010) reviewed various smart wheelchair systems, highlighting the evolution of technologies and methodologies used in their development. The review covered a range of sensor technologies including ultrasonic, infrared, and laser-based systems, as well as control interfaces such as joysticks, touchscreens, and voice recognition. Brown noted the increasing trend toward incorporating AI and machine learning to enhance adaptive navigation and user personalization. The review concluded that while significant advancements have been made, challenges remain in ensuring affordability and accessibility for a wider population. [16] [17] [21] [22]

CHAPTER 2: METHODOLOGY

2.1 Workflow Description

The main workflow of this smart wheelchair project encompasses a series of steps designed to ensure the safety and convenience of individuals with mobility impairments. The process begins with turning on the system and deciding what type of control the user wants to use. Manual control via a joystick, allows users to navigate their environment intuitively and with precision. Concurrently, the voice recognition module enables hands-free operation by interpreting user commands, offering an alternative control method that enhances accessibility and ease of use.

For using the joystick control, the first step is to check, if the joystick is active for forward and backward directions. If the joystick is disabled for forward and backward we should check for the obstacle presence. If the obstacle is present, we should pass the obstacle to continue and start the checking process again. If the joystick is not active and there is no obstacle, it means that there is a problem with the system, so the user should call for help.

In case, if the joystick is active, it means there is no obstacle, so the user can move to the destination. Then it is checked, if the destination is reached. If no – the process should be started from the joystick checking process again. If yes – the process can be finished.

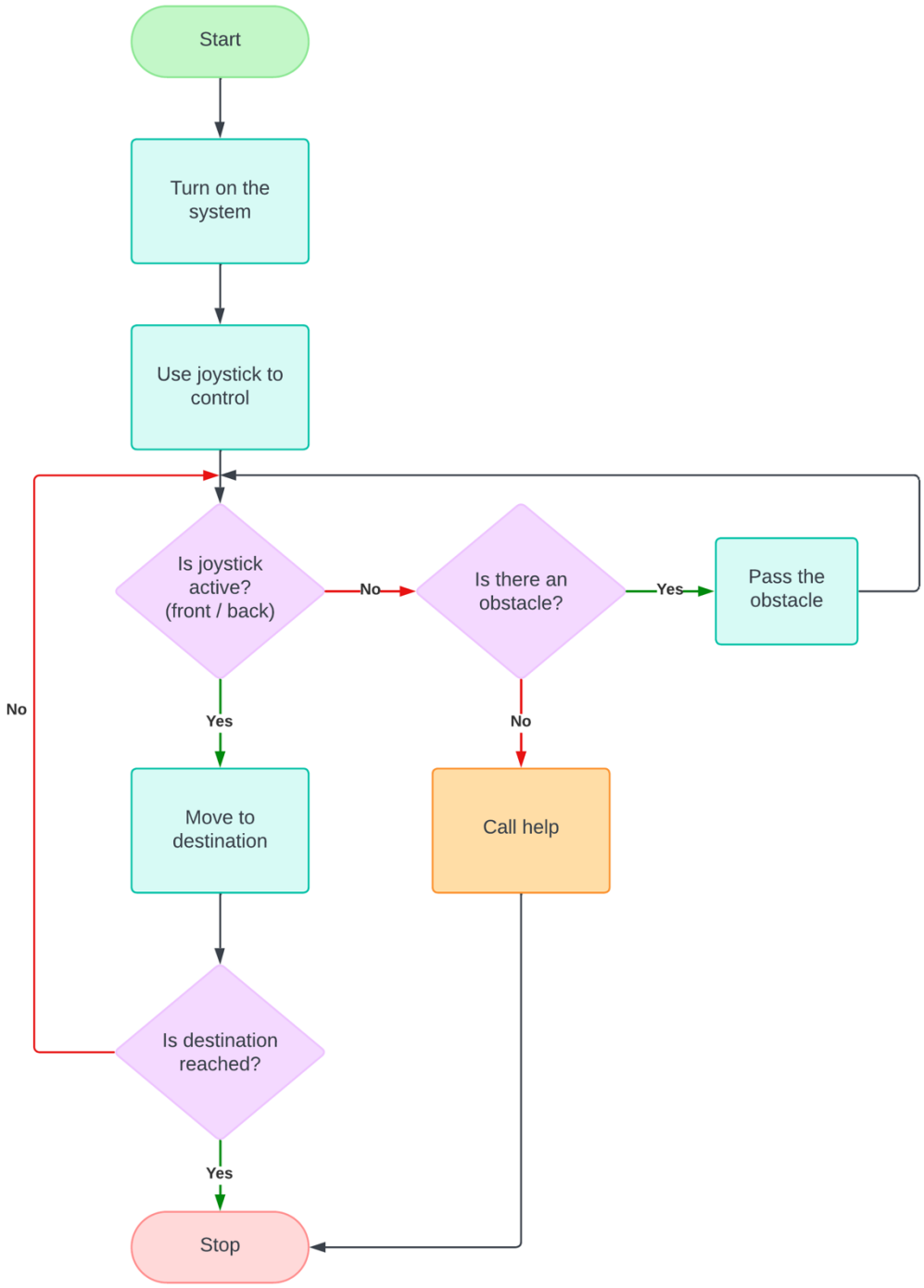

Figure 1: The joystick control workflow of the project.

For using the voice control, the first step is to check if the voice control is active for forward and backward directions. If the voice control is disabled for forward and backward, the user should check for the obstacle's presence. If the obstacle is present, the obstacle should be passed to continue and start the checking process again. If the voice control is not active and there is no obstacle, it means that there is a problem with the system, so the user should call for help.

In case the voice control is active, it means there is no obstacle, so the user can move to the destination. Then it is checked if the destination is reached. If not, the process should be started from the voice control checking process again. If yes, the process can be finished.

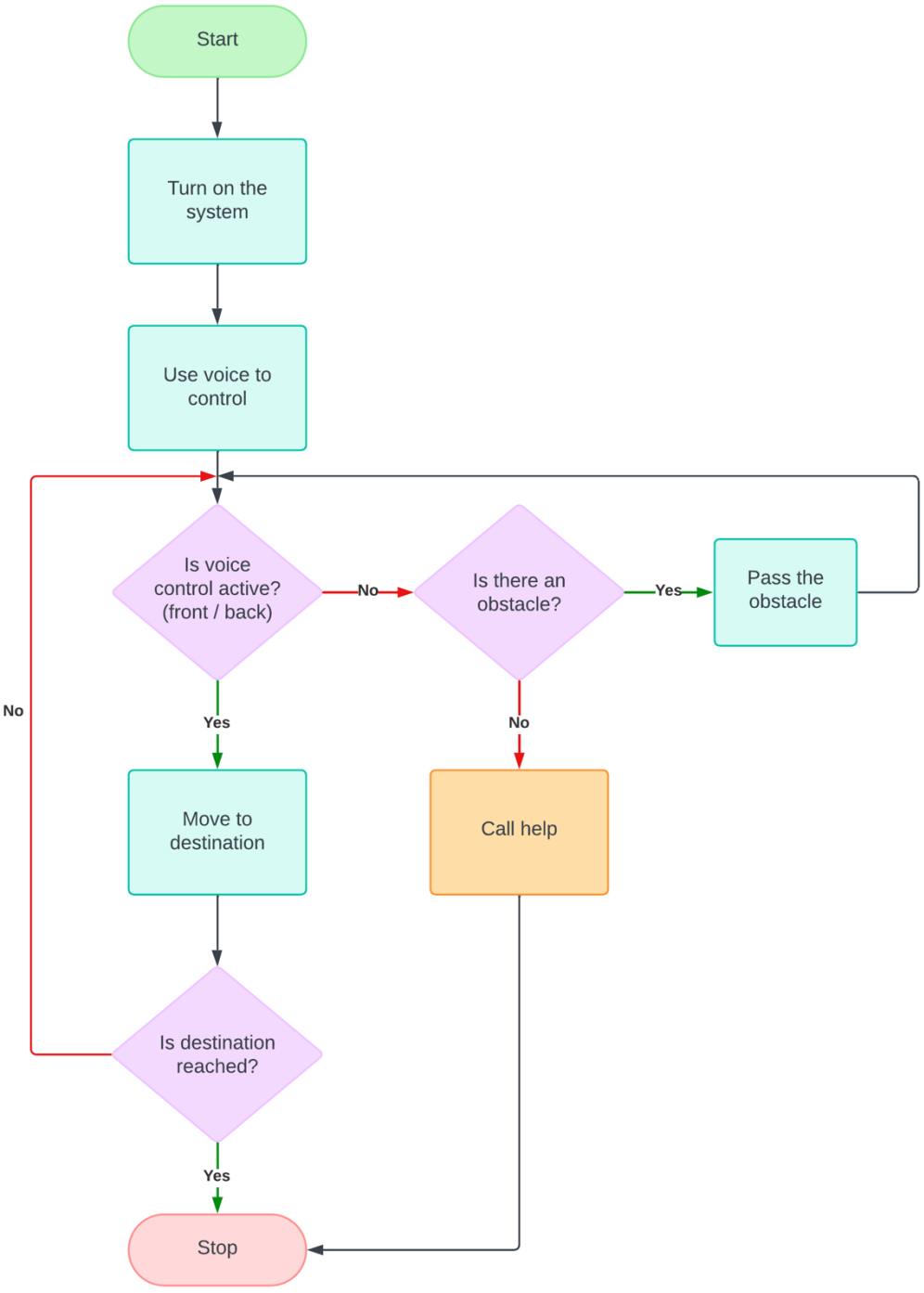

Figure 2: The voice control workflow of the project.

2.2 Manual and Voice Control

The manual control process within the workflow begins with the joystick input, where users can maneuver the wheelchair with precision. The system continuously checks for user input and responds in real-time to ensure smooth navigation. Simultaneously, the voice recognition module listens for specific commands, offering an alternative means of control. When a voice command is recognized, the system interprets the command and adjusts the wheelchair's actions accordingly. This dual-control approach empowers users to choose the most convenient method of operation, enhancing their autonomy and flexibility.

2.3 Obstacle Avoidance

The obstacle avoidance process is initiated when the sonar sensors detect potential obstacles in the wheelchair's path. The system assesses the proximity of these obstacles and issues an audio warning via the DFPlayer module to alert the user. If an obstacle is detected within a critical distance, the system automatically adjusts the wheelchair's trajectory to avoid collision. This proactive measure ensures that users can navigate safely in various environments without constant manual intervention.

2.4 Audio Alerts and Entertainment

The DFPlayer module serves a dual purpose by providing audio alerts and entertainment. In case of critical alerts, such as obstacle warnings or system errors, the module plays pre-recorded messages to inform the user promptly. Additionally, users can enjoy music playback, enhancing their overall experience while using the

wheelchair. The horn feature further supports user safety by enabling them to alert pedestrians and others in their vicinity.

2.5 Hardware components

2.5.1 Motors

As our project is electrical, we are supposed to use an electric motor. For example, Electric DC motors offer reliable and efficient mechanical power conversion. They convert electrical energy into mechanical motion, enabling devices like electric bicycles, scooters, and wheelchairs to move. For a smart wheelchair project, an electric DC motor provides the necessary propulsion, ensuring smooth and controlled movement. We decided to choose a DC motor because it’s simpler to operate and manage than AC motors and we have experience working with DC motors before.

For the wheels to make our wheelchair go, we will be using 12 V DC motors placed at the back. We picked these motors because they provide enough power to move the wheelchair with a person in it. Also, 12 V is a common and easy-to-use power level. This choice makes sure the motors get the right amount of power and avoids any complications.

Using 12 V is handy because it's widely used, making it easy to find the right power supplies. It also makes our wheelchairs more reliable and straightforward. We're sticking with what's familiar and practical. What's cool about these DC motors is that we can control their speed using something called Pulse Width Modulation (PWM). We've got experience using PWM to control DC motors from past projects. This means we can easily manage how fast or slow the wheelchair moves, making it simple and efficient to control. [1]

The MY1025Z2 Electric DC Motor, available on Amazon, is a high-quality, reduction-geared motor specifically designed for applications requiring significant torque and power. This motor is commonly used in electric bicycles, but its specifications make it ideal for integration into a smart wheelchair system.

Figure 3: MY1025Z2 Electric DC Motor.

Key Features:

1. High Torque: Equipped with a reduction gearbox, the motor delivers high torque, essential for moving a wheelchair with ease.
2. Durable Construction: Built with high-quality materials, ensuring longevity and reliability under continuous use.
3. Sprocket Compatibility: Includes a chain sprocket, making it easy to integrate with various drive systems.

4. Efficient Power Consumption: Designed to maximize efficiency, reducing the load on the wheelchair's battery system.

Technical Specifications:

- Model: MY1016Z3
- Motor Type: Brushed DC motor
- Rated Voltage: 12V
- Rated Power: 250W
- Rated Speed: 2950 RPM
- Reduction Ratio: 8.9:1
- Output Sprocket: 9-tooth sprocket for #410 chain
- Weight: Approximately 2.8 kg
- Dimensions: 15.5 cm x 10 cm x 7.5 cm

Integrating the MY1016Z3 Electric DC Motor into a smart wheelchair can provide robust and reliable propulsion. Here are some practical applications and benefits:

1. Smooth Navigation: The high torque and precise control offered by the motor ensure smooth acceleration and deceleration, crucial for safe and comfortable wheelchair movement.
2. Variable Speed Control: The motor supports variable speed settings, allowing users to adjust the wheelchair's speed according to their preferences and environmental conditions.

3. Energy Efficiency: The efficient design of the motor minimizes power consumption, extending the battery life of the wheelchair and reducing the need for frequent recharges.

The MY1025Z2 Electric DC Motor operates through the interaction of its internal components, including the armature, brushes, and gearbox. Here's a step-by-step breakdown of its operation:

1. Electric Input: Electrical power is supplied to the motor from the wheelchair's battery.
2. Armature Rotation: The current flows through the motor's armature, creating a magnetic field that interacts with the field magnets, causing the armature to rotate.
3. Gear Reduction: The rotation of the armature is transferred to the output shaft through a reduction gearbox. This gearbox reduces the high-speed rotation into lower-speed, high-torque output.
4. Sprocket Drive: The output shaft is connected to a sprocket, which drives the wheelchair's chain and wheel system, converting the motor's rotational energy into forward or backward movement.

So, in simple terms, we chose these motors because they provide the power we need, are easy to work with, and our team knows how to control them effectively using PWM. This ensures our wheelchair is both strong and easy to manage. [2] [5]

2.5.2 Controller

To give a brain to the wheelchair thus making it smart, a Microcontroller was implemented into the circuit, microcontrollers offer the intelligence to power common goods. This Microcontroller serves as a whole computer system in a single chip, though compact, this device has a CPU, memory, and peripheral interfaces to accomplish tasks efficiently and precisely. For our wheelchair to process data from the inputs in real-time, and act upon the sent information, such as doing specific tasks like motor control, a microcontroller is essential. After Evaluating several options, the Arduino Mega 2560 was chosen as the core microcontroller for our project.

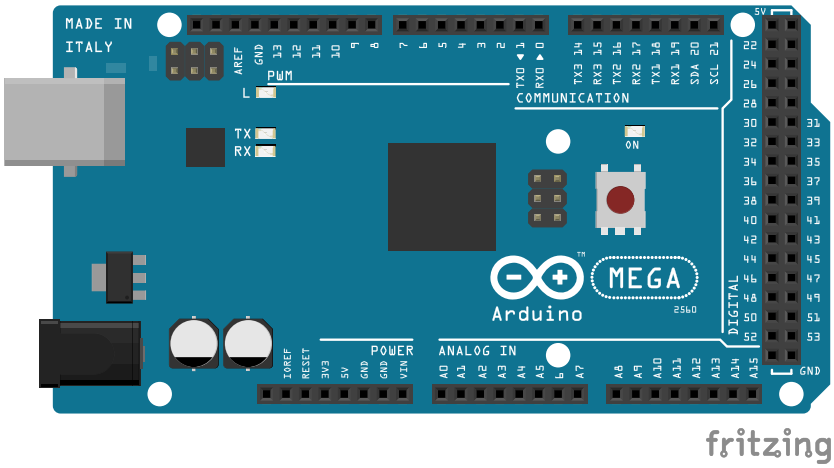

Figure 4: Arduino Mega board.

The main reasons go back to the specifications of Arduino Mega over other Microcontrollers such as Arduino UNO such as:

1. Memory Capacity:

   Mega offers 256 KB of flash memory, with 8 KB used by the bootloader this is suitable for our code (approximately 550-line code) where we feared another microcontroller may not have sufficient memory. The Arduino Mega's expanded

memory capacity allows for more complex programming and better performance in handling large-scale projects like a smart wheelchair.

2. Input/Output Pins:

Mega offers 54 Digital Pins (including 16 PWM) and 16 Analog Pins, which leaves space for not only implementing all the required components but also expanding the circuit to add more and more resources to make it scalable and smarter.

3. Processing power:

The ATmega2560 microcontroller provides sufficient processing power to handle multiple tasks simultaneously, including sensor data processing, motor control, and user interface operations, and a 16 MHz clock speed, it is essential to provide sufficient processing power for the wheelchair to work in real-time and react to any disturbance quickly.

In summary, the Arduino Mega Controller emerges as a practical and functional choice for integrating the various components of our smart wheelchair. Its user-friendly design, compatibility, reliability, upgrade capability, and cost-effectiveness collectively make it a suitable candidate for the brain of our project. [3]

2.5.3 Power system

To power our smart wheelchair system, we've chosen a 12 V 9 Ah battery, and here's why it's the ideal choice for our project.

The 12 V rating is a perfect match for our system's needs. This standard voltage level simplifies the setup, preventing any compatibility issues and making it easy to find the right charging equipment. The 9 Ah capacity of the battery is like its fuel tank. With 9 Ah, we have more than enough energy to run the entire smart wheelchair system. This

ample capacity ensures a reliable and extended runtime, allowing the wheelchair to operate smoothly without frequent recharging. The 12 V 9 Ah battery strikes a balance between size and weight. It's compact enough to fit into the wheelchair without adding excessive bulk, and its lightweight nature ensures that the wheelchair remains manageable and practical for daily use.

Being a rechargeable battery is a significant advantage. It allows for extended usage without the need for frequent battery replacements. When it's time to recharge, the process is straightforward, adding to the convenience of our smart wheelchair.

Figure 5: 12 V 9 Ah battery.

As mentioned before, we need less voltage to run our controller and input devices. To step down the voltage and avoid a high current that can damage devices we can use a buck converter.

The buck converter, also known as a step-down converter, is widely used in power electronics to convert higher input voltage to a lower output voltage. It plays a crucial role in diverse applications, such as portable devices and automotive systems,

where specific components or subsystems require a lower voltage. The key advantage of the buck converter lies in its simplicity, allowing efficient voltage conversion with a minimal number of components.

In the operation of a buck converter, controlled energy transfer occurs from the input to the output using switches, an inductor, and a capacitor. A high-side switch (typically a MOSFET) and a low-side switch (usually a diode) regulate the current flow through the inductor. By adjusting the duty cycle of the high-side switch, the average output voltage can be controlled in proportion to the input voltage.

When the high-side switch is turned on, it permits current to flow through the inductor, storing energy in its magnetic field. This stored energy is then transferred to the output, charging the output capacitor, and supplying power to the load. When the high-side switch is turned off and the low-side switch is turned on, the inductor's magnetic field collapses, releasing the stored energy and sustaining the current flow to the load. The buck converter operates within a closed-loop control system, where feedback continuously compares the output voltage to a reference voltage, ensuring stable and regulated output voltage despite changes in input voltage or load conditions.

So, after some research, we decided to pick a buck converter from a company called “Drok”, because they have a good reputation on the market and the variety of options they provide. The DROK DC Buck Module is an adjustable buck converter step-down voltage regulator that supports a wide input voltage range from 6V to 32V and provides an adjustable output voltage range from 1.5V to 32V. It is capable of delivering up to 5A of current and features an LCD display, USB port, and a protective case.

Figure 6: DROK buck converter.

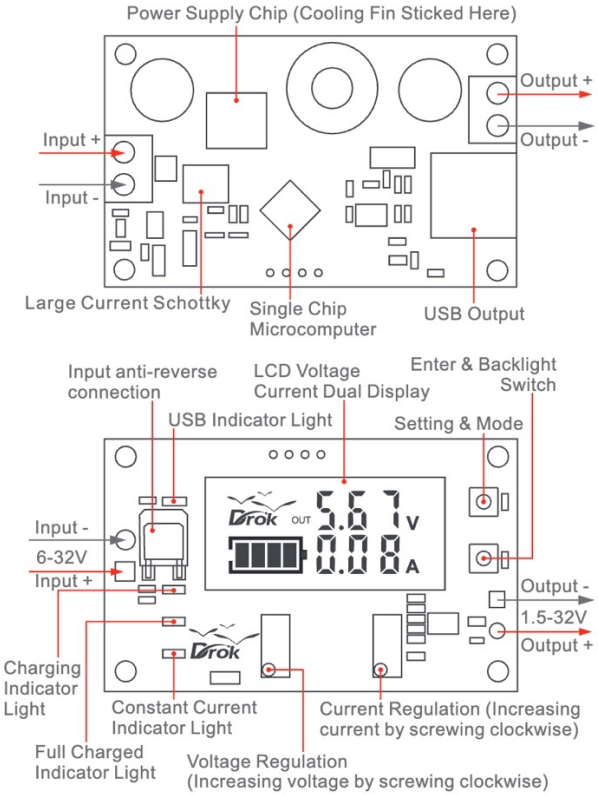

Figure 7: DROK DC buck converter main components.

DROK DC buck converter offers a wide input voltage range of 6V to 32V and an adjustable output voltage range of 1.5V to 32V, making it versatile for various power needs. The module supports a high current capacity of up to 5A, ensuring ample power delivery to connected devices. An LCD display provides real-time voltage and current readings, enhancing usability and monitoring. Additionally, the module includes a USB port, which allows for convenient charging of USB devices. Its protective case ensures the module is safeguarded from physical damage and short circuits, contributing to its reliability and durability in long-term use.

The Arduino Mega operates at 5V and has a recommended input voltage of 7-12V via the Vin pin. The DROK module can step down the main battery voltage to these levels efficiently, ensuring stable operation of the Arduino Mega. The project may require different voltages for various sensors and actuators, and the DROK module's adjustable output makes it versatile for supplying power to different components with varying voltage requirements. The Arduino Mega itself has moderate power consumption, but additional modules and sensors can increase the demand. The 5A capacity of the DROK module ensures there is ample current available for all components. The LCD display simplifies the process of setting and monitoring voltages, which can help during the development and debugging stages of the project. The protective case and reliable voltage regulation make the DROK module a safe choice for long-term use in a smart wheelchair project, minimizing the risk of electrical faults.

In summary, the 12V 9Ah battery ensures reliable and efficient operation of our smart wheelchair with its compatibility, ample capacity, and ease of recharging. Integrating the Drok DC Buck Module provides adjustable voltage, high current capacity,

and essential features like real-time monitoring, USB charging, and protective casing. Its compatibility with Arduino Mega enhances functionality and safety, reinforcing our commitment to a dependable and user-friendly smart wheelchair system. [8] [25] [26]

2.5.4 Motor Control

Motor control is a fundamental aspect of modern engineering that involves regulating the motion and operation of motors. This encompasses starting, stopping, and adjusting the speed, torque, and direction of motors. Effective motor control systems are crucial for applications ranging from simple household appliances to complex industrial machinery. These systems utilize various techniques and components to achieve precise control over motor functions, ensuring efficient and reliable operation.

2.5.4.1 H-bridge circuit

At the heart of many motor control systems is the H-Bridge circuit. An H-Bridge allows a voltage to be applied across a load in either direction. It is a fundamental building block for DC motor control and consists of four switches that direct the current flow. By closing specific switches, the direction of the current—and hence the direction of the motor rotation—can be controlled.

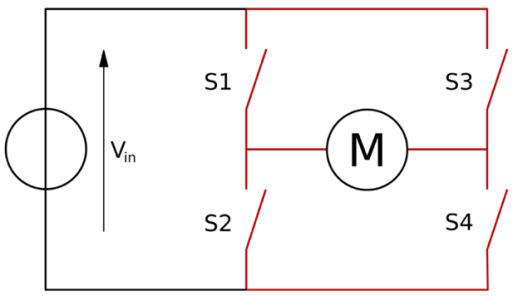

Figure 8: H-Bridge circuit.

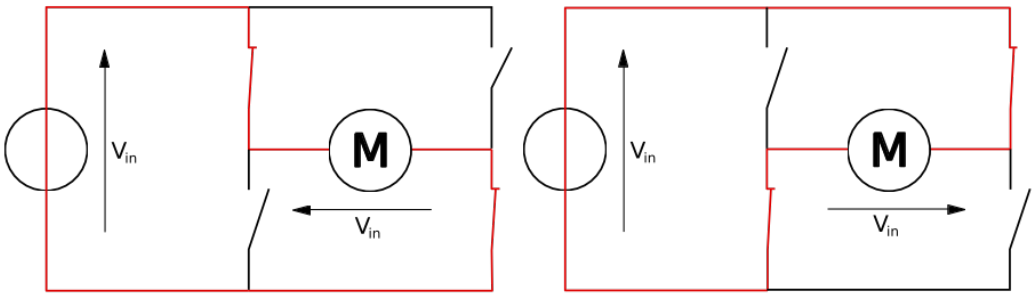

Figure 9: 2 states of H-Bridge circuit.

The simplicity of the H-Bridge design belies its power and versatility in controlling DC motors. Each switch can be a transistor, MOSFET, or any other electronic switch capable of handling the desired current. [1]

2.5.4.2 Pulse Width Modulation (PWM)

Pulse Width Modulation (PWM) is a technique used to control the speed of the motor by varying the duty cycle of the voltage pulses. By adjusting the ratio of the on-time to the off-time, the effective voltage supplied to the motor can be controlled, which in turn adjusts the motor speed.

The average voltage (and thus the motor speed) can be controlled by adjusting the width of the pulses. A higher duty cycle results in a higher average voltage and a faster motor speed, while a lower duty cycle results in a lower average voltage and a slower motor speed.

PWM is preferred for its efficiency, as it reduces power loss in the switching components and allows for fine control over the motor speed. It is widely used in motor control applications due to its simplicity and effectiveness. [1]

Using the function “analogWrite()” in Arduino IDE we can create a PWM signal to get the desired output voltage:

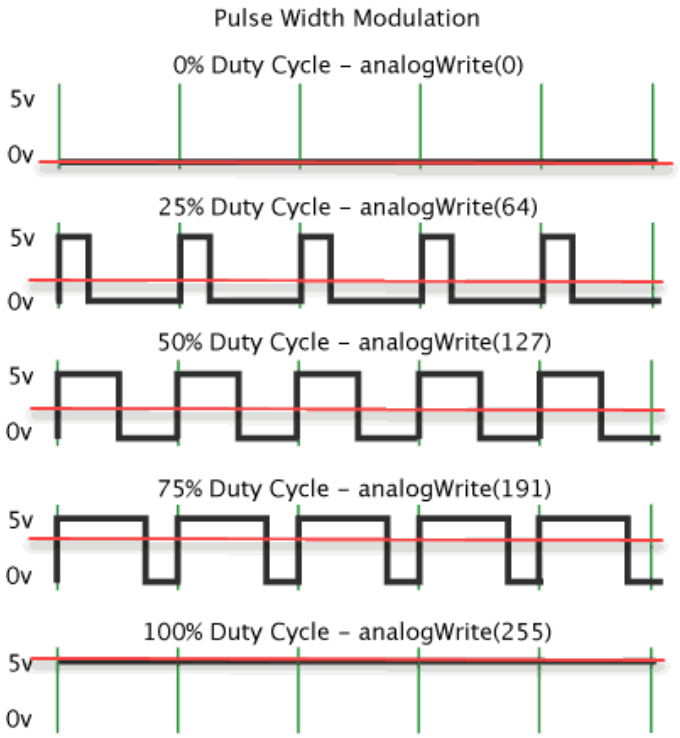

Figure 10: PWM output graphs with different duty cycles using “analogWrite()”.

2.5.4.3 Motor Driver

Before we started researching for our project, we didn't know how to control a powerful motor. This motor had 250 W power with a 12 V rate, which means it could draw a huge amount of current. Regular motor drivers like the L293n used in small robotics projects couldn't handle this much current and would burn out. After looking into it, we found a good solution called the "IBT-2" motor driver.

The IBT-2 motor driver is a high-current H-Bridge module designed for driving DC motors with a current capacity of up to 43A. It utilizes dual BTS7960 chips, making it suitable for applications requiring robust performance and protection features.

Key Specifications:

- Input Voltage: 6V-27V for motor, 5V for logic
- Control Method: PWM or logic-level inputs
- Protection Features: Over-temperature, over-current, under-voltage, and short-circuit protection
- Size: 50mm x 50mm x 43mm
- Weight: ~66g

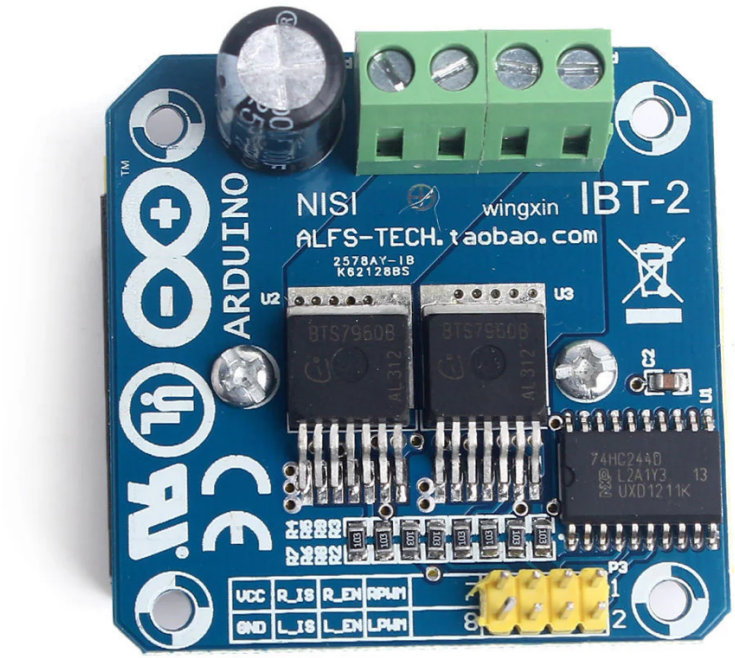

Figure 11: IBT-2 motor driver.

The IBT-2 module is designed to interface easily with microcontrollers, allowing for straightforward integration into various projects. Its compact size and high current capacity make it a versatile choice for a wide range of motor control applications.

The IBT-2 motor driver receives control signals from a microcontroller, which are buffered by the SN74HC244DW (U1) to ensure they are strong enough to drive the BTS7960 chips (U2 and U3). The BTS7960 chips then control the current flow through the motor, allowing it to rotate in either direction depending on the input signals.

When the control signal sets IN1 high and IN2 low, U2 drives the motor in the forward direction. Conversely, when the control signal sets IN1 low and IN2 high, U3 drives the motor in the reverse direction. By using PWM signals on the control inputs, the motor speed can be adjusted by varying the duty cycle of the signals.

To use the IBT-2 motor we should provide a power that we desire to use to the B+ pin as a positive terminal and B- pin as a negative terminal. The next step is to connect the motors to M+ and M- pins of the modules. In order to power the logic signals of the driver we should connect the Vcc and GND pins of the module to 5V and GND pins accordingly on Arduino Mega. As we need to use both directions of motors we do the action that enables it - provide a positive voltage to pins R_EN and L_EN on the motor driver. The PWM signals are supplied to RPWM for forward and LRPM for backward direction operation.

In conclusion, the IBT-2 motor driver is an excellent choice for our project. It offers robust protection against excessive current, high temperatures, and short circuits, ensuring safety and reliability. Its efficient motor control minimizes power waste, making it an energy-conscious option. The IBT-2 is user-friendly, with accessible

information and instructions that simplify setup and integration with our microcontrollers. Additionally, its cost-effectiveness provides the necessary power without straining our budget. Overall, the IBT-2 motor driver meets our requirements for handling high current and delivering precise motor control, making it the right choice for our project. [13]

2.5.4.4 Joystick control

In the realm of modern technology, the concept of control has evolved into a sophisticated and essential aspect of user interaction with machines and devices. Control, in this context, refers to the ability to influence the behavior and performance of a system or device using various input methods. Among these methods, joysticks stand out as one of the most intuitive and versatile tools available. Joysticks provide a direct and tactile means of controlling a wide array of devices, from video games and remote-controlled toys to advanced machinery and assistive technologies like smart wheelchairs.

Control, in common sense, involves the manipulation and regulation of devices to achieve desired outcomes. This can be as simple as steering a toy car or as complex as piloting a drone. Effective control systems ensure that the user can translate their intentions into precise actions performed by the device. Key aspects of control include:

1. Intuitiveness: Controls must be easy to understand and use, allowing users to operate devices with minimal training.

2. Responsiveness: Controls should provide immediate feedback and response to user inputs, ensuring smooth and accurate operation.
3. Precision: High precision is critical, especially in applications requiring fine motor skills and detailed movements.

Joysticks are among the most effective tools for achieving intuitive and precise control. They convert the user's physical movements into electrical signals, which are then processed by the device's control system to perform specific actions. The versatility of joysticks makes them suitable for various applications, ranging from recreational use in gaming to critical functionalities in assistive technologies.

1. Ergonomics and Design: Modern joysticks are designed with user comfort in mind, featuring ergonomic grips and adjustable resistance to reduce fatigue and enhance control over long periods. This is especially important in applications like smart wheelchairs, where the user might rely on the joystick for daily mobility.
2. Technological Advancements: Joysticks have benefited from significant technological advancements. Traditional potentiometer-based joysticks have largely been replaced by those utilizing Hall Effect sensors. These sensors detect the position of the joystick by measuring changes in a magnetic field, providing greater accuracy and durability. This technology ensures that the joystick remains precise and reliable over extended use, as it is less susceptible to wear and tear.

3. Applications in Assistive Technologies: In the context of smart wheelchairs, joysticks provide a vital interface for users with limited mobility. These joysticks must be highly responsive and easy to use, allowing for precise navigation through various environments. The integration of joysticks with advanced control systems can significantly enhance the user's independence and quality of life.

As technology continues to advance, the potential for joysticks in various applications expands. Innovations such as haptic feedback, which provides tactile sensations to the user, can further enhance the user experience by giving real-time feedback about the device's actions. Wireless connectivity and integration with other smart technologies also open new possibilities for remote operation and customization. [4]

2.5.4.4.1 JH-D400X-R4 joystick

The JH-D400X-R4 Hall Effect Sensor Joystick, available on platforms like Amazon, presents an advanced solution for enhancing the maneuverability and responsiveness of a smart wheelchair. Leveraging Hall Effect technology, this joystick ensures superior accuracy and durability compared to traditional potentiometer-based joysticks.

Figure 12: JH-D400X-R4 Hall Effect Sensor Joystick.

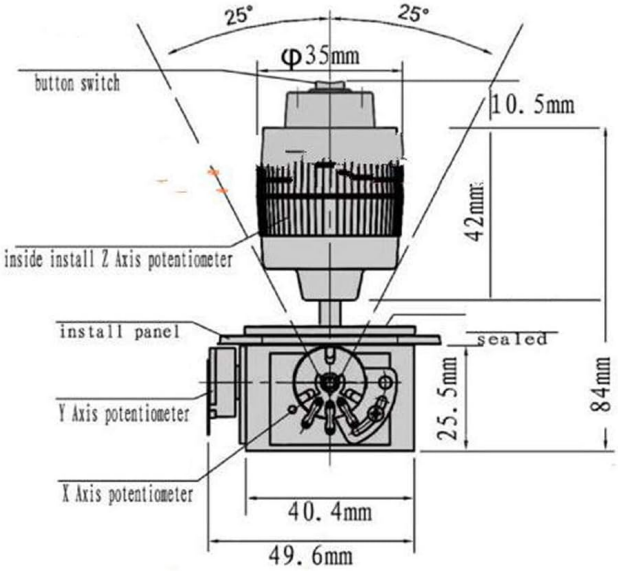

Figure 13: JH-D400X-R4 Joystick dimensions.

Key Features of the JH-D400X-R4 Joystick:

1. Precision Control: The JH-D400X-R4 joystick provides highly accurate input detection, critical for navigating tight spaces and making smooth turns in a wheelchair.
2. Durability: With no mechanical parts to wear out, JH-D400X-R4 joysticks offer enhanced longevity. They are resistant to environmental factors such as dust and moisture, which can affect traditional joysticks.

Ergonomic Design: Designed for comfort, the JH-D400X-R4 joystick can be used for extended periods without causing fatigue, making it ideal for wheelchair users who rely on their device throughout the day.

Technical Specifications of the JH-D400X-R4 Joystick

- Sensor Type: Hall Effect
- Model: JH-D400X-R4
- Output: Analog signals corresponding to joystick position
- Power Supply: Typically operates within a 5V range
- Connection: Easily interfaces with microcontrollers, such as Arduino Mega, for integration into various control systems.
- Button: A momentary switch push button is present at the top of the joystick. It can be used

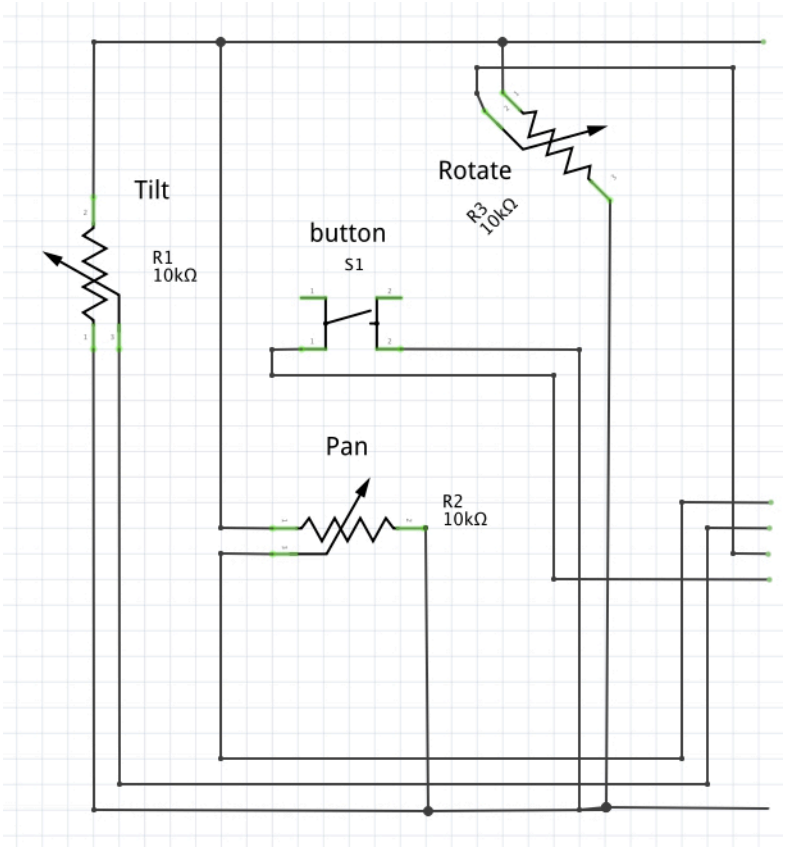

Figure 14: JH-D400X-R4 Joystick circuit diagram.

The JH-D400X-R4 joystick is straightforward to integrate with an Arduino Mega for a smart wheelchair project. Here's a step-by-step outline:

1. Wiring: Connect the joystick's power and ground pins to the Arduino Mega's 5V and GND pins, respectively. Connect the output pins to the analog input pins on the Arduino.
2. Coding: Utilizing the Arduino IDE to write a program that reads the analog signals from the joystick. The code maps these signals to control commands for the wheelchair's motors.

3. Calibration: Implement software calibration to ensure that the joystick’s neutral position corresponds to zero movement of the wheelchair. This involves reading the initial values of the joystick and adjusting the control signals accordingly.
4. Testing: Conduct thorough testing in various environments to ensure reliable operation. Adjust the sensitivity and response curves in the software to match the user's preferences.

In conclusion, the JH-D400X-R4 joystick is a sophisticated component that enhances the functionality and reliability of smart wheelchairs. Its precision, durability, and ease of integration with systems like Arduino Mega make it a valuable addition to any assistive technology project. By leveraging this technology, users can achieve smoother, more intuitive control over their wheelchairs, significantly improving their mobility and independence. [2] [4]

2.5.5 Obstacle avoidance system

In our smart wheelchair project, the Obstacle Avoidance System will rely on clever sonar sensors to help navigate safely. A good thing is that these sensors are smart about using power. They don't use too much energy, so the wheelchair's battery lasts longer. This makes the wheelchair not just smart but also efficient.

Ultrasonic sensors are devices that use sound waves to measure the distance to an object. They operate by emitting an ultrasonic sound wave at a frequency higher than the human ear can detect (typically around 40 kHz). When this sound wave encounters an object, it bounces back to the sensor, which then calculates the distance based on the time

it took for the echo to return. The model we picked is HC-SR04, which is the most popular module for electronics projects and suits most of the applications.

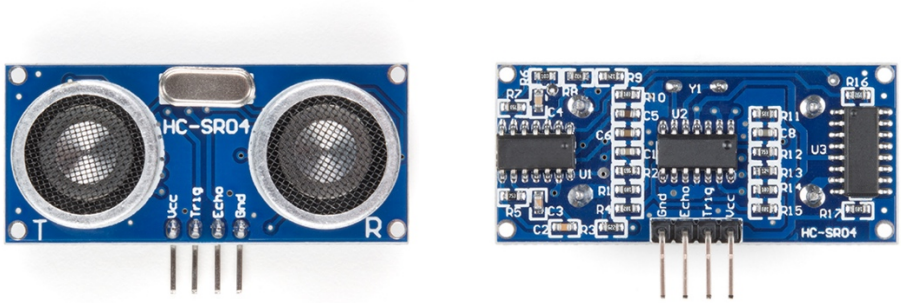

Figure 15: HC-SR04 ultrasonic sensor module.

HC-SR04 technical specifications:

- Operating voltage: 5V
- Operating current: 15mA
- Operating frequency: 40Hz
- Max range: 4 meters
- Min Range: 2 cm

The working principle of the sensor involves several steps. First, a short 10 µs pulse is sent to the Trig pin to start the measurement. The sensor then emits an ultrasonic sound wave at 40 kHz. This sound wave travels through the air and bounces back when it hits an obstacle. The sensor detects the reflected sound wave at the Echo pin. Finally, the time difference between sending the trigger pulse and receiving the echo pulse is used to calculate the distance to the obstacle.

In our project, four sensors were used, two in front and two in back, so the chair is aware of every side and can act upon detecting obstacles and alert the user for suspecting any irregular sound wave.

Continuous distance measurements can be fed into the Arduino Mega for real-time processing, allowing immediate adjustments to the wheelchair's path or speed based on the proximity of obstacles. Sensor data can also be used to alert the user of nearby obstacles through visual or auditory signals, enhancing safety and awareness. Furthermore, the wheelchair can dynamically adjust its speed or stop completely if an obstacle is detected within a critical distance, preventing accidents and ensuring user safety.

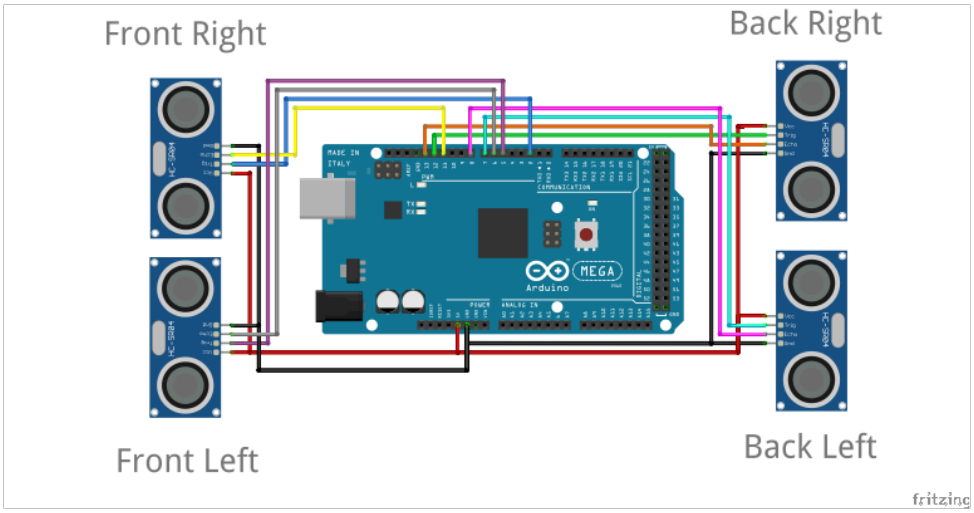

Figure 16: Sonar Sensor circuit diagram.

In conclusion, integrating HC-SR04 ultrasonic sensors into our smart wheelchair enhances safety and efficiency. The sensors' low power usage conserves battery life, and their placement—two in front and two in back—ensures comprehensive obstacle

detection. The Arduino Mega processes real-time data, allowing the wheelchair to adjust its path and speed promptly. Users are alerted to nearby obstacles through visual or auditory signals, improving awareness and safety. This smart obstacle avoidance system not only prevents collisions but also boosts user confidence in navigating various environments, offering a reliable and advanced mobility solution. [19]

2.5.6 Sound system

Sound systems are vital in assistive technologies, offering immediate and intuitive information delivery. In our smart wheelchair project, we decided to use DFPlayer Mini MP3 Player that enhances user experience and safety with high-quality audio for alerts, notifications, and instructions. This compact module is cost-effective, easy to integrate, and supports multiple control modes and file systems. By incorporating the DFPlayer Mini, we ensure users receive clear auditory feedback, improving navigation and overall interaction with the wheelchair.

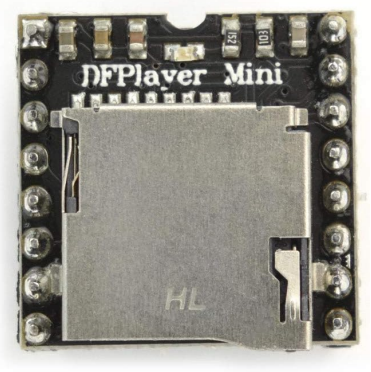

Figure 17: DF player mini.

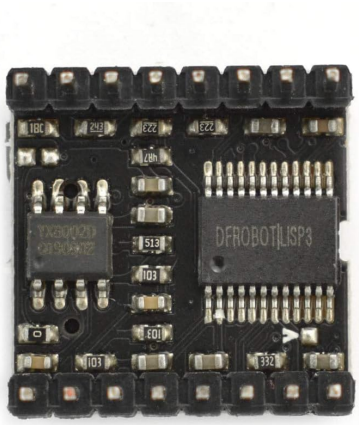

Figure 18: DF player mini hardware.

DF player mini Technical Specifications:

- Sampling Rates (kHz): 8/11.025/12/16/22.05/24/32/44.1/48
- DAC Output: 24-bit, supports dynamic range of 90dB and SNR of 85dB
- File System Support: Fully supports FAT16 and FAT32, with a maximum support of 32GB for TF card, 32GB for U disk, and 64MB for NOR FLASH
- Control Modes: I/O control mode, serial mode, and AD button control mode
- Audio Data Management: Supports up to 100 folders, each capable of holding up to 255 songs
- Volume Control: 30-level adjustable volume, 6-level EQ adjustable

Working Principle of DF player mini in Our Project

1. Integration with Arduino Mega:
    - The DFPlayer Mini is connected to the Arduino Mega via serial communication.

- The Arduino Mega serves as the central controller, managing inputs from sensors and outputs to the speaker, motors, and other components.

2. Control via Arduino Mega:
    - Commands are sent from the Arduino Mega to the DFPlayer Mini to control audio playback.
    - The Arduino processes data from ultrasonic sensors and other inputs to trigger specific audio responses.

DF player mini applications in Smart Wheelchair:

1. Music Playback:
    - Allows users to play their favorite music for entertainment and relaxation during use.
2. Horn Sound:
    - Provides an audible horn sound to alert pedestrians and other vehicles, enhancing safety in crowded or busy areas.
3. Obstacle Avoidance Mode:
    - Integrates with ultrasonic sensors to provide voice alerts when obstacles are detected, helping the user navigate safely.

In our project, the DFPlayer Mini is connected as follows:

- DF Player Mini to Arduino Mega:
    - TX (DFPlayer) to RX (Arduino Mega)
    - RX (DFPlayer) to TX (Arduino Mega)

- VCC (DFPlayer) to 5V (Arduino Mega)
- GND (DFPlayer) to GND (Arduino Mega)

- Speaker to DF Player Mini:
  - SPK1 (DFPlayer) to Speaker positive
  - SPK2 (DFPlayer) to Speaker negative

Figure 19: 4 Ohm 3W Speaker.

- Noise Reduction: A 1K resistor is attached to the TX pin for noise reduction.

Figure 20: 1 kOhm Resistor.

- MP3 Management: The order of MP3 files copied to the micro SD card that affects the playback sequence.

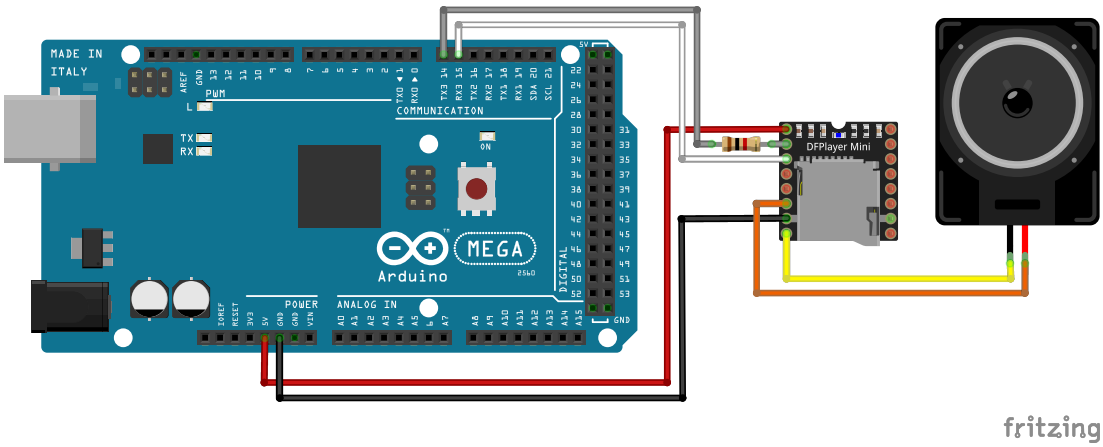

Figure 21: DFPlayer mini Circuit connection.

By integrating the DFPlayer Mini MP3 Player into the smart wheelchair, we enhance the user interface with auditory feedback, ensuring that users receive clear and timely information about their environment and the wheelchair's status. This is crucial for both safety and convenience, providing an additional layer of interaction that complements visual and tactile feedback systems. [6]

2.5.6 Voice control system

Voice recognition technology, also known as speech recognition, allows computers and electronic devices to understand and process human speech. This technology converts spoken language into digital data, enabling voice-activated control systems. For a smart wheelchair project, voice recognition can significantly enhance user accessibility and ease of control. Users with mobility impairments can issue voice

commands to navigate, control speed, and perform other functions without physical interaction with the wheelchair controls.

2.5.6.1 Gravity Voice Recognition Module

The Gravity Voice Recognition Module, SKU SEN0539-EN, is an advanced module designed by DF Robot for seamless integration into various projects requiring voice command functionalities. This module supports both I2C and UART communication protocols, making it versatile and compatible with a wide range of microcontrollers, including the Arduino Mega.

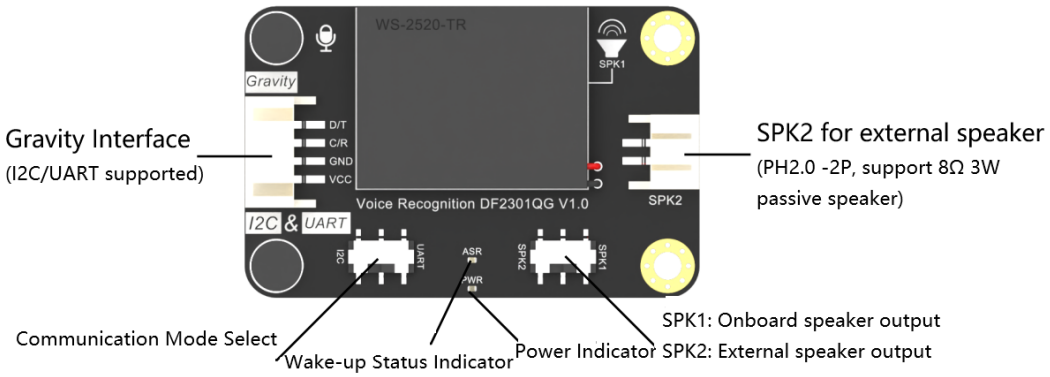

Figure 22: DF Gravity Voice Recognition System.

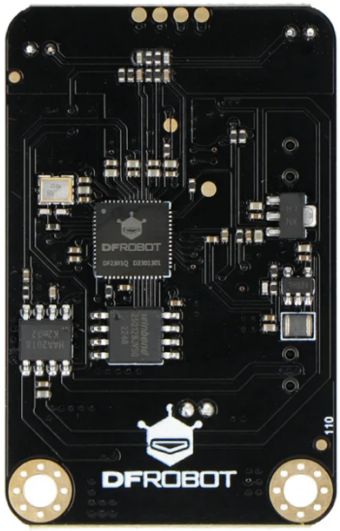

Figure 23: DF Gravity Voice Recognition System Hardware.

Gravity Voice Recognition Module Key Features:

1. Dual Communication Modes: Supports I2C and UART interfaces for flexible connectivity.
2. High Recognition Accuracy: Equipped with sophisticated algorithms to achieve high accuracy in voice recognition.
3. Speaker-Dependent Recognition: Trains to recognize specific voices, enhancing command accuracy.
4. Command Customization: Users can program up to 22 custom commands.
5. Noise Filtering: Built-in noise filtering to improve recognition in various environments.

Gravity Voice Recognition Technical Specifications:

- Operating Voltage: 3.3V - 5V

- Current Consumption: ≤ 50mA
- Recognition Distance: 1m to 2m
- Dimensions: 30mm x 47mm / 1.18in x 1.85in
- Operating Temperature: -10°C to 55°C
- Built-in speaker
- Built-in microphone
- Power indicator
- Status indicator
- Communication mode select

Incorporating the Gravity Voice Recognition Module into a smart wheelchair can revolutionize the way users interact with their mobility device. Here are some practical applications:

1. Navigation Commands: Users can issue voice commands such as "go forward," "turn left," "stop," etc., to control the wheelchair's movement.
2. Entertainment Control: Commands like "play music" allow users to have entertainment during their ride.

The Gravity Voice Recognition Module operates by capturing voice inputs through its built-in microphone. It then processes these inputs using its onboard DSP (Digital Signal Processor) to recognize predefined commands. Here’s a step-by-step breakdown of how the module works:

1. Voice Capture: The user speaks a command, which is captured by the microphone.
2. Preprocessing: The captured audio signal is preprocessed to filter out noise and enhance the voice signal.
3. Feature Extraction: The module extracts features from the voice signal that are essential for recognition.
4. Pattern Matching: The extracted features are compared against the stored voice command patterns.
5. Command Execution: Upon recognizing a command, the module sends the corresponding data through the I2C or UART interface to the microcontroller (e.g., Arduino Mega) for execution.

2.5.6.2 I2C communication

For our design we decide to use the I2C communication mode of the module. I2C (Inter-Integrated Circuit) is a widely used communication protocol that allows multiple devices to communicate with each other over two wires: Serial Data Line (SDA) and Serial Clock Line (SCL). Developed by Philips Semiconductor (now NXP Semiconductors), it is used for communication between microcontrollers and peripherals such as sensors, displays, and other integrated circuits. The key features of I2C include:

1. Two-Wire Interface: Only two wires are needed for communication, making it ideal for applications with limited pin availability.
2. Addressable Devices: Each device on the I2C bus has a unique address, allowing multiple devices to share the same bus without conflict.

3. Master-Slave Configuration: One device (the master) controls the clock and initiates communication, while the other devices (slaves) respond.
4. Data Transfer Rates: Supports multiple speed modes, including Standard Mode (100 kbps), Fast Mode (400 kbps), Fast Mode Plus (1 Mbps), and High-Speed Mode (3.4 Mbps).

2.5.6.3 Gravity Voice Recognition with Arduino Mega

The Arduino Mega has dedicated pins for I2C communication:

- SDA (Serial Data Line) on pin 20
- SCL (Serial Clock Line) on pin 21

To use Gravity Voice Recognition module with I2C communication on the Arduino Mega, follow these steps:

1. Hardware Setup

Connect the SDA and SCL pins of your I2C device(s) to the corresponding pins on the Arduino Mega. Connect the Vcc and GND pin of Gravity voice recognition module to the Arduino 5V and GND pins.

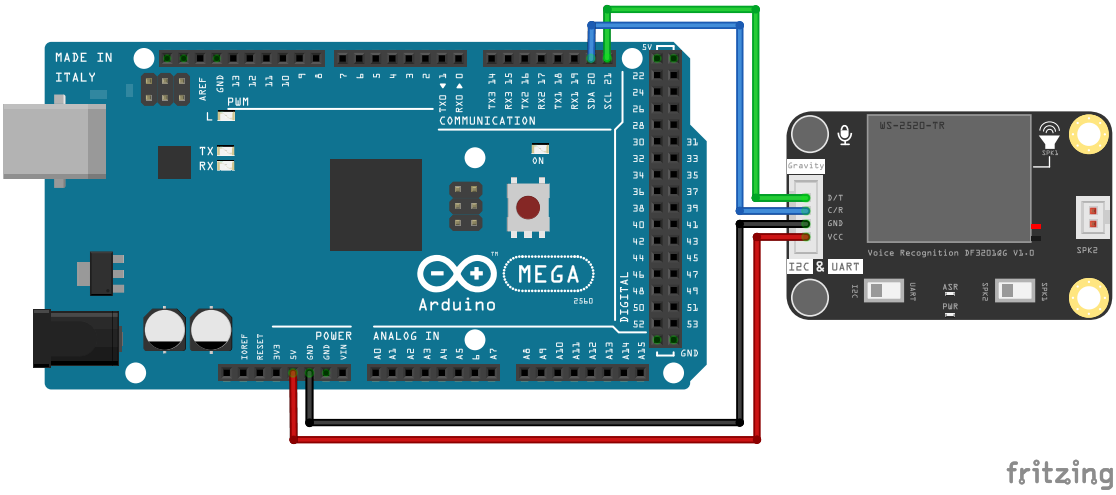

Figure 24: Voice recognition system circuit diagram.

2. Library Installation

You need the DFRobot_DF2301Q_I2C library to interact with the module. Install the library via the Arduino Library Manager

3. Example Code

Use the example code given by the manufacturer on their official website

For our specific our goal was to set up the code in the way that when the commands are triggered the other parts of other circuit perform the actions needed. For example, when the user says the voice command “Go forward”, the controller receives that it is a command number 22 and, if the joystick is not active at the moment it sends the command to the motor drivers to actuate the motors. [10] [12]

CHAPTER 3: DESIGN AND IMPLEMENTATION

3.1 Actual wheelchair

Our smart wheelchair project kicks off with a regular, strong wheelchair. We picked this simple one because it's tough and people already like using it. Regular wheelchairs are made to be comfy and easy to use. We want to keep that comfort and simplicity while adding our cool features. Using this common wheelchair makes it easy for us to add our smart stuff without messing up how it already works. These wheelchairs are everywhere, so getting one is not going to be a problem. And because they're not super advanced, they keep costs down. This means we can focus on making our smart additions without spending too much money.

Figure 25: Wheelchair purchased.

In addition, people already know how to use regular wheelchairs. Starting with something they're familiar with makes it easier for them to use our smart wheelchair when we're done. In short, we're starting with a regular wheelchair because it's strong, comfortable, easy to find, and doesn't cost too much. We want to keep what's already good about it while making it even better with our smart touches.

3.2 Software tool used

The project utilized the Arduino Integrated Development Environment (IDE) for its software development. This software tool is tailored for programming Arduino microcontrollers and offers a user-friendly interface for code creation, compilation, and uploading. What sets the Arduino IDE apart is its simplicity, making it accessible even to beginners in electronics and programming. It boasts features like a built-in text editor, serial monitor for debugging, and a comprehensive library of functions and examples for development assistance. Supporting the Arduino programming language, which is a simplified version of C/C++ with additional libraries for Arduino hardware, the IDE was an excellent fit for the blind stick project. Its user-friendly nature, extensive documentation, and active community support proved invaluable in streamlining the software development process, facilitating efficient and successful implementation of required functionalities.

3.3 Circuit diagram

The circuit diagram depicts the interconnected components of the smart wheelchair system, which comprises a joystick, a voice recognition module, multiple ultrasonic sensors, motor drivers (IBT-2), a DFPlayer Mini MP3 module, a DROK buck converter, a horn button, a speaker, a 12V 9 Ah battery, and an Arduino Mega 2560 board.

The joystick allows manual control of the wheelchair, enabling users to navigate their environment intuitively. Concurrently, the voice recognition module offers hands-free operation by interpreting user commands, enhancing accessibility and ease of use.

The ultrasonic sensors are employed for obstacle detection, continuously monitoring the surroundings to identify potential obstructions in the wheelchair's path. When an obstacle is detected, the system provides audio feedback through the speaker, alerting the user with messages like "obstacle detected." This proactive measure ensures safe navigation by preventing collisions. The DFPlayer Mini MP3 module, in conjunction with the speaker, also provides entertainment by playing music, improving the overall user experience.

The IBT-2 motor drivers control the wheelchair motors, receiving signals from the Arduino Mega to adjust the speed and direction based on input from the joystick or voice commands. The DROK buck converter manages the power supply, ensuring stable voltage and current to the system components. The 12V lead-acid battery serves as the primary power source, supplying the necessary energy to the entire system.

The horn button allows the user to activate the horn, alerting pedestrians and others in the vicinity, and enhancing safety. The Arduino Mega 2560 board acts as the central processing unit, orchestrating the operations of all connected components.

This circuit diagram, created using Fritzing software, visually represents the layout and connections of the various components in the smart wheelchair system. Fritzing is a popular open-source tool for designing and documenting electronic circuits, featuring an intuitive interface suitable for both beginners and experienced electronics enthusiasts. It offers a vast library of components and symbols, allowing users to easily

build and visualize their circuits. The software also provides features for naming components, making notes, and customizing the circuit design, with the capability to export the schematic in various formats.

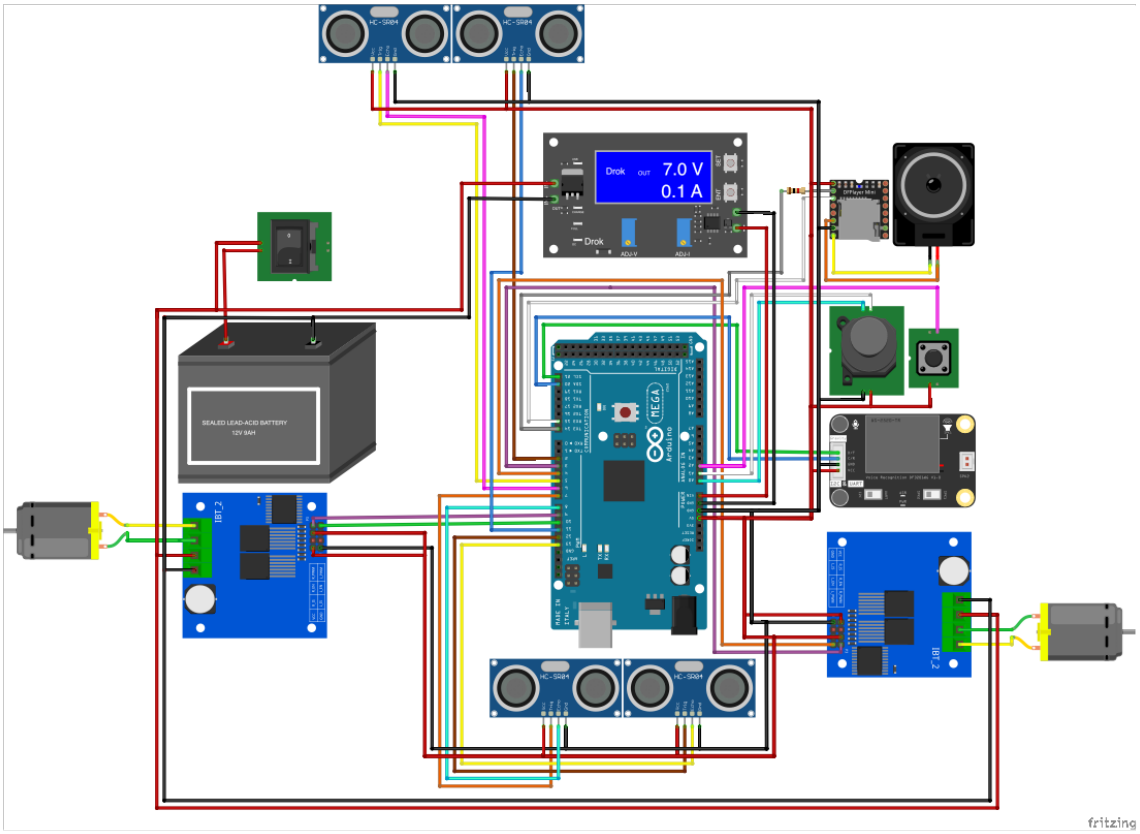

Figure 26: Complete circuit diagram of Smart Wheelchair.

By depicting the overall electrical structure and component linkages, the circuit diagram serves as a valuable visual reference, enhancing understanding and communication of the project's design.

3.4 Implementation of the Smart Wheelchair

In the development of the smart wheelchair, several enhancements were incorporated to improve usability and provide essential functionality for the user. A

standard wheelchair was used as the base, upon which an MDF platform was installed to support the motors and battery. Additionally, two wooden platforms were integrated to house the circuitry, ensuring a robust and organized setup.

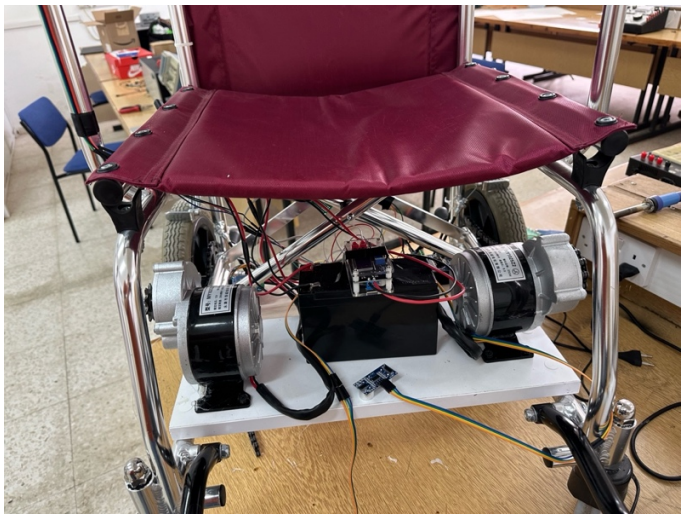

Figure 27: MDF platform with motors and battery.

A power switch was strategically placed on the right side of the wheelchair for easy access, allowing the user to control the power effortlessly.

Figure 28: Power switch.

To enhance obstacle detection, two rods were installed to mount the front sonar sensors. This thoughtful placement ensures that the sensors accurately detect obstacles ahead, rather than the user's legs.

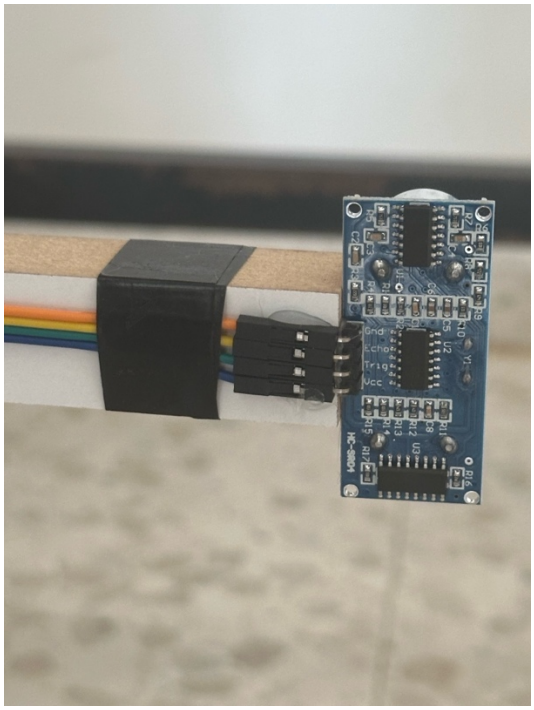

Figure 29: Front sonar sensor installed.

Mechanically, gears were attached to the wheels, and spacers were utilized to position the wheels slightly outward, creating space for the chain mechanism. This careful arrangement ensures smooth and reliable operation of the wheelchair. The chain, essential for transmitting power, was meticulously integrated into the design.

Figure 30: Gears and chains.

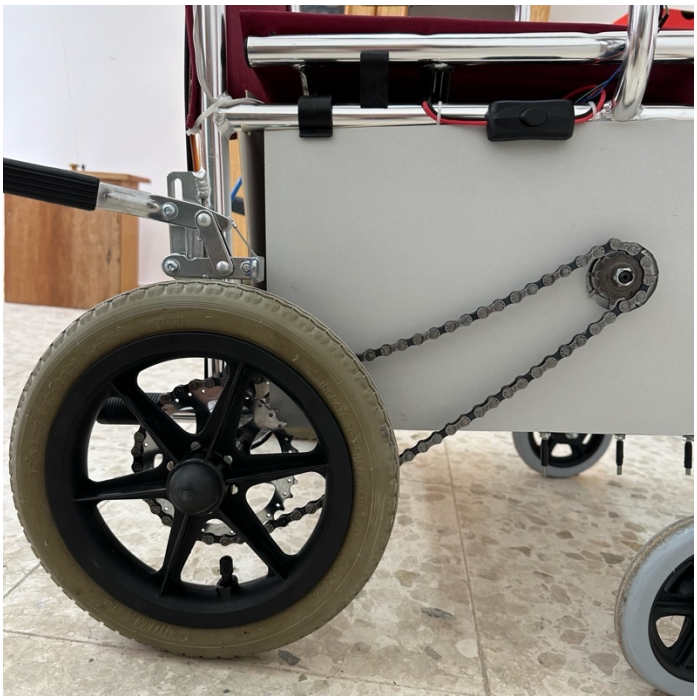

Figure 31: Mechanical system of Smart Wheelchair.

Extensive wire management was undertaken to maintain a clean and organized appearance. All electronic components were concealed from the user's view, enhancing the aesthetic appeal and preventing any potential distractions. For maintenance, the seat, which is fabric-based, can be easily removed by unscrewing it, allowing engineers to access the electronic components for repairs.

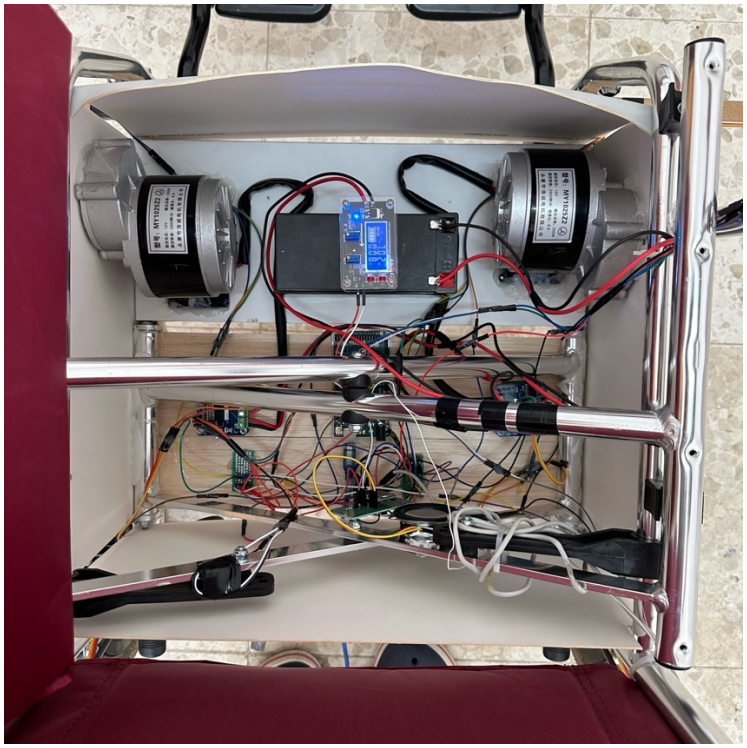

Figure 32: Smart Wheelchair inside.

The joystick, which serves as the primary control interface, was mounted on the right armrest of the wheelchair. A custom-made box was designed to house the joystick, providing a neat and ergonomic solution. Additionally, the voice recognition module was attached to this box, ensuring that it can accurately pick up the user's commands without any obstructions.

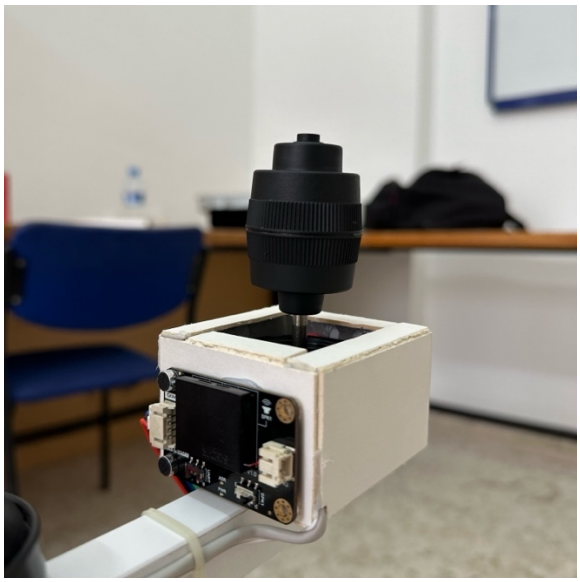

Figure 33: Joystick and voice recognition installed.

The careful integration of these components into the smart wheelchair, along with thoughtful placement and design considerations, significantly enhances the overall functionality and user experience. The design emphasizes ease of use, safety, and reliability, ensuring that the smart wheelchair meets the needs of its users effectively.

CHAPTER 4: STANDARDIZATION

4.1 Ethics and Limitations

As responsible engineers, it is essential to recognize and address the potential limitations and ethical implications associated with the development of our smart wheelchairs. Conducting thorough risk assessments and evaluating any potential hazards or risks associated with the use of the smart wheelchair is crucial. This includes assessing the reliability and accuracy of the sensors, ensuring that the system functions reliably in various environmental conditions, and addressing any potential concerns related to privacy and data security. Additionally, we must consider the safety and comfort of the user, ensuring that the wheelchair can handle different terrains and obstacles without compromising user safety. We also understand the importance of user feedback and involvement throughout the development process. Engaging with individuals who use wheelchairs, their caregivers, and relevant stakeholders allows us to gain valuable insights, validate the effectiveness of our solution, and make necessary improvements based on their experiences and suggestions. This participatory approach ensures that the final product meets the actual needs and preferences of the users.

Our commitment to ethical engineering standards extends beyond the initial implementation phase. Regular maintenance, calibration, and updates are necessary to ensure the continued reliability and performance of the smart wheelchair. This involves periodic checks and testing of components, monitoring the accuracy of sensor readings, and promptly addressing any issues or malfunctions that may arise. By adhering to these ethical considerations and limitations, we aim to create a safe, reliable, and effective

solution that significantly enhances the lives of individuals with mobility impairments, promoting independence and improving their overall well-being. [14]

4.2 Engineering Standards

To ensure that the smart wheelchair adheres to engineering design principles and satisfies the necessary criteria, it is imperative to follow specific standards and regulations. These standards cover the requirements and quality guidelines set by various organizations. One of the key sources of engineering standard design guidelines is provided by the Institute of Electrical and Electronics Engineers (IEEE). The International Organization for Standardization (ISO) ensures that ethical considerations are addressed and that the necessary criteria for projects are met in accordance with ethical norms. The smart wheelchair project involves numerous electrical components, various types of sensors, and circuit elements. It is crucial to ensure that these components meet the required standards without causing any harm to the user. The design and analysis should prevent any risk of electric shock or other failures. For example, sensitive data generated by sensors must be safeguarded and transmitted securely, and all electrical connections and installations should be carried out safely to prevent any harm to the user.

In addition, adherence to the standards set by regulatory bodies ensures that all project-related items are manufactured to the necessary specifications prior to construction. This compliance not only guarantees the safety and reliability of the smart wheelchair but also fosters trust and confidence among users and stakeholders. By following these engineering standards and ethical guidelines, we can develop a smart

wheelchair that is not only functional and efficient but also safe and user-friendly, thereby improving the quality of life for individuals with mobility impairments. [15]

4.3 Cost analysis

Item	Price
Motor driver (2 pcs.)	$30
Joystick	$28
Ultrasonic sensor pack (5 pcs.)	$15
Jumper wires (240 pcs.)	$20
PCB board prototype pack	$17
SD card reader	$13
DC motor (2 pcs.)	$190
Arduino Mega 2560	$65
SD card (32 GB)	$16
Audio speaker	$8
Voice recognition module with a microphone	$45
12V 9ah battery	$25
Wheelchair	$140
Mechanical materials	$80
Other materials	$20
Total	$712

Table 1: Cost analysis.

Our project is special because it's way cheaper than the motorized wheelchairs you find in stores. We figure our spending is around $712, including $140 for the actual wheelchair, $472 for our additional smart system, $80 for mechanical parts and $20 for some other expenses.

Now, if you look at regular motorized wheelchairs, they start at around $2000 after adding delivery and taxes. Ours is a way more affordable option. What makes our project stand out is that we're turning a regular wheelchair into a smart one with great features, and we're doing it for much less money. The big, fancy motorized wheelchairs don't offer these features at such a reasonable price. [2]

4.4 Project planning

Month	February		March				April				May				June		
Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Plan	Registration and Brainstorming		Preparing Project Equipment				Project Construction and Design				Modifying and Finalizing Report				Presentation Preparation		Presentation
Done																	

Table 2: Project planning

CHAPTER 5: CONCLUSION

In conclusion, our project aimed to address the challenges faced by individuals with mobility impairments in their daily lives and provide them with a smart wheelchair that enhances their safety, independence, and overall quality of life. Through the integration of advanced sensor technologies, including ultrasonic sensors for obstacle detection, we created a comprehensive system that alerts the user in real time to potential hazards in their environment.

The inclusion of a joystick and voice control system allows users to navigate the wheelchair with ease, providing them with multiple options for control based on their preferences and physical capabilities. This adaptability ensures that the wheelchair can be operated comfortably and efficiently by a wide range of users.

To enhance the user experience further, we incorporated a horn and music system. The horn serves as a safety feature, allowing users to alert others to their presence and the music system, utilizing the DFPlayer Mini MP3 module and a speaker, offers auditory entertainment and instructions. This feature allows users to play a variety of audio cues and alarms tailored to different scenarios and needs.

We focused on the user experience and accessibility by building a user-friendly interface with audible feedback, tactile signals, and easily accessible controls. The lightweight and ergonomic design of the smart wheelchair sought to optimize comfort and usability during long periods of usage. We worked hard to verify the smart wheelchair's dependability, precision, and usefulness in real-world circumstances

through rigorous testing and development. We aimed to foster inclusion and respond to individual preferences by building the system to adapt to varied user demands.

APPENDIX

A.1 Smart wheelchair code

```
//include the libraries for Arduino, voice recognition and mp3 player
#include "DFRobot_DF2301Q.h"
#include "Arduino.h"
#include "DFRobotDFPlayerMini.h"

//define FPSerial as Serial 3 ports
#define FPSerial Serial3

DFRobotDFPlayerMini myDFPlayer;
void printDetail(uint8_t type, int value);

int max_speed_def = 80; // Maximum speed

// Motor Left
int leftForw = 9; // Enable Forward pin for Motor Left
int leftBack = 10; // Enable Backward pin for Motor Left

// Motor Right
int rightForw = 3; // Enable Forward pin for Motor Right
int rightBack = 4; // Enable Backward pin for Motor Right
```

```
// Joystick Input
int joyVert = A0; // Vertical joystick pin
int joyHorz = A1; // Horizontal joystick pin
int joyButt = A2; // Button joystick pin

// Motor Speed Values - Start at zero
int MotorSpeedLeft = 0; // Speed value for Left Motor
int MotorSpeedRight = 0; // Speed value for Right Motor

// Joystick Values - Start at 512 (middle position)
int joyposVert = 512; // Current vertical joystick position
int joyposHorz = 512; // Current horizontal joystick position

// Sonar sensors pins
int TriggerFrontLeft = 5;
int EchoFrontLeft = 6;
int TriggerBackLeft = 7;
int EchoBackLeft = 8;

int TriggerFrontRight = 2;
int EchoFrontRight = 11;
int TriggerBackRight = 12;
int EchoBackRight = 13;
```

```
// Sonar sensor data
const int obstacleThreshold = 50; // Distance threshold for obstacle detection
bool obstacleDetectedFront = false;    // Flag to keep track of obstacle detection
bool obstacleDetectedBack = false;    // Flag to keep track of obstacle detection

long durationFrontLeft;
int distanceFrontLeft;

long durationBackLeft;
int distanceBackLeft;

long durationFrontRight;
int distanceFrontRight;

long durationBackRight;
int distanceBackRight;

// Voice command flags
bool voiceCommandForward = false;
bool voiceCommandBackward = false;
bool voiceCommandStop = false;
bool voiceCommandTurnLeft = false;
```

```
bool voiceCommandTurnRight = false;

// Flag to indicate whether a voice command is currently active
bool voiceCommandActive = false;

// State variable to track if the horn was played
bool hornPlayed = false;

//Time values for obstacle warning
bool actionPerformedFront = false;
unsigned long lastActionTimeFront = 0;
const unsigned long actionIntervalFront = 7000; // 7 seconds in milliseconds
const unsigned long minIntervalAfterActionFront = 1250; // 1.25 second in milliseconds

bool actionPerformedBack = false;
unsigned long lastActionTimeBack = 0;
const unsigned long actionIntervalBack = 7000; // 7 seconds in milliseconds
const unsigned long minIntervalAfterActionBack = 1250; // 1.25 second in milliseconds

//I2C communication
DFRobot_DF2301Q_I2C asr;

void setup() {
```

```
  //Serial communication with the DF Player Mini
  FPSerial.begin(9600);
  // Initialize serial communication
  Serial.begin(115200);

  myDFPlayer.setTimeOut(1000); //Set serial communictaion time out 1000 ms

  // Retry mechanism for DFPlayer initialization
  bool dfPlayerInitialized = false;
  for (int attempts = 0; attempts < 5; attempts++) {
    if (myDFPlayer.begin(FPSerial, /*isACK = */true, /*doReset = */true)) {  // Use
serial to communicate with mp3.
      dfPlayerInitialized = true;
      break;
    }
    Serial.println(F("Initialization failed, retrying..."));
    delay(1000);  // Wait 1 second before retrying
  }

  if (!dfPlayerInitialized) {
    Serial.println(F("Unable to begin:"));
    Serial.println(F("1. Please recheck the connection!"));
```

```
  Serial.println(F("2. Please insert the SD card!"));
  while (true) {
    delay(0);  // Code to compatible with ESP8266 watch dog.
  }
}

Serial.println(F("DFPlayer Mini online."));
myDFPlayer.volume(25);  // Set volume value. From 0 to 30

// Set all the motor control pins to outputs
pinMode(leftForw, OUTPUT);
pinMode(leftBack, OUTPUT);
pinMode(rightForw, OUTPUT);
pinMode(rightBack, OUTPUT);

// Set the sonar sensor pins
pinMode(TriggerFrontLeft, OUTPUT);
pinMode(EchoFrontLeft, INPUT);
pinMode(TriggerBackLeft, OUTPUT);
pinMode(EchoBackLeft, INPUT);
pinMode(TriggerFrontRight, OUTPUT);
pinMode(EchoFrontRight, INPUT);
pinMode(TriggerBackRight, OUTPUT);
```

```
  pinMode(EchoBackRight, INPUT);

  // Init the voice recognition module
  while (!(asr.begin())) {
    Serial.println("Communication with device failed, please check connection");
    delay(3000);
  }
  Serial.println("Begin ok!");

  asr.setMuteMode(0); // Mute mode to 0
  asr.setWakeTime(60); // Be awake for 60 seconds since the wake up

  uint8_t wakeTime = asr.getWakeTime();
  Serial.print("wakeTime = ");
  Serial.println(wakeTime);
}

void loop() {

  // Define a timeout for pulseIn (in microseconds)
  const long timeout = 30000; // 30 ms

  //Obstacle Front Left
```

```
  distanceFrontLeft = getDistance(TriggerFrontLeft, EchoFrontLeft, timeout);

  //Obstacle Back Left
  distanceBackLeft = getDistance(TriggerBackLeft, EchoBackLeft, timeout);

  //Obstacle Front Right
  distanceFrontRight = getDistance(TriggerFrontRight, EchoFrontRight, timeout);

  //Obstacle Back Right
  distanceBackRight = getDistance(TriggerBackRight, EchoBackRight, timeout);

  // Check for obstacles
  obstacleDetectedFront = ((distanceFrontLeft <= obstacleThreshold &&
distanceFrontLeft != -1) || (distanceFrontRight <= obstacleThreshold &&
distanceFrontRight != -1));
  obstacleDetectedBack = ((distanceBackLeft <= obstacleThreshold &&
distanceBackLeft != -1) || (distanceBackRight <= obstacleThreshold &&
distanceBackRight != -1));

  unsigned long currentTimeFront = millis();
  unsigned long currentTimeBack = millis();

  if (obstacleDetectedBack) {
```

```
    // Check if the action has been performed and the interval has passed
    if (!actionPerformedBack || (currentTimeBack - lastActionTimeBack >=
actionIntervalBack)) {
      performActionBack(currentTimeBack);
    }
  }
  else{
    // If the action has been performed and less than 1 second has passed since the last
action, do nothing
    if (actionPerformedBack && (currentTimeBack - lastActionTimeBack <
minIntervalAfterActionBack)){
      return;
    }
    lastActionTimeBack = currentTimeBack;
    actionPerformedBack = false;
  }

    if (obstacleDetectedFront) {
    // Check if the action has been performed and the interval has passed
    if (!actionPerformedFront || (currentTimeFront - lastActionTimeFront >=
actionIntervalFront)) {
      performActionFront(currentTimeFront);
    }
```

```
  }
  else{
    // If the action has been performed and less than 1 second has passed since the last
action, do nothing
    if (actionPerformedFront && (currentTimeFront - lastActionTimeFront <
minIntervalAfterActionFront)){
      return;
    }
    lastActionTimeFront = currentTimeFront;
    actionPerformedFront = false;
  }

  //maximum speed for motors
  int max_speed = max_speed_def;

  uint8_t CMDID = asr.getCMDID();
  if (CMDID != 0) {
    // Reset all voice command flags
    voiceCommandForward = false;
    voiceCommandBackward = false;
    voiceCommandStop = false;
    voiceCommandTurnLeft = false;
    voiceCommandTurnRight = false;
```

```
switch (CMDID) {
  case 22:
    // Go forward
    Serial.println("received 'Go forward', command flag '22'");
    voiceCommandForward = true;
    voiceCommandActive = true;
    break;

  case 23:
    // Go backward 8
    Serial.println("received 'Go backward', command flag '23'");
    voiceCommandBackward = true;
    voiceCommandActive = true;
    break;

  case 5:
    // Stop
    Serial.println("received 'Stop', command flag '5'");
    voiceCommandStop = true;
    voiceCommandActive = false;
    break;
```

```
case 6:
  // Turn left
  Serial.println("received 'Turn left', command flag '6'");
  voiceCommandTurnLeft = true;
  voiceCommandActive = true;
  break;

case 7:
  // Turn right
  Serial.println("received 'Turn right', command flag '7'");
  voiceCommandTurnRight = true;
  voiceCommandActive = true;
  break;

case 92:
  //Play music
  Serial.println("received 'Play Music', command flag '92'");
  myDFPlayer.play(7);
  delay(100);
  break;

case 93:
  //Stop playing
```

```
      Serial.println("received 'Stop playing', command flag '93'");
      myDFPlayer.pause();
      break;

    default:
      Serial.print("CMDID = ");
      Serial.println(CMDID);
      break;
  }
}

// Read the joystick X and Y positions
joyposVert = analogRead(joyVert);
joyposHorz = analogRead(joyHorz);

// Read the Joystick button state
int buttonHorn = analogRead(joyButt);

  if (buttonHorn > 1010 && joyposHorz < 767) {
  if (!hornPlayed) {
    myDFPlayer.play(3);  // Play the horn sound
    delay(10);
    hornPlayed = true;   // Set the state variable to true
```

```
    }
  } else {
    hornPlayed = false;  // Reset the state variable when the button is released
  }

  // Check if the joystick is moved from the rest position
  bool joystickMoved = !(joyposVert >= 460 && joyposVert <= 564 && joyposHorz
>= 460 && joyposHorz <= 564);

  // If joystick is moved, override voice commands
  if (joystickMoved) {
    voiceCommandActive = false;
    voiceCommandForward = false;
    voiceCommandBackward = false;
    voiceCommandTurnLeft = false;
    voiceCommandTurnRight = false;
  }

  if (voiceCommandActive) {
    if (voiceCommandForward) {
        if (!obstacleDetectedFront) {
            analogWrite(leftBack, max_speed);
            analogWrite(rightForw, max_speed+5);
```

```
      analogWrite(leftForw, 0);
      analogWrite(rightBack, 0);
    } else {
      analogWrite(leftForw, 0);
      analogWrite(rightBack, 0);
      analogWrite(leftBack, 0);
      analogWrite(rightForw, 0);
      voiceCommandForward = false;  // Reset the command flag
    }
  } else if (voiceCommandBackward) {
    if (!obstacleDetectedBack) {
      analogWrite(leftForw, max_speed);
      analogWrite(rightBack, max_speed+5);
      analogWrite(leftBack, 0);
      analogWrite(rightForw, 0);
    } else {
      // Stop the motors if an obstacle is detected
      analogWrite(leftForw, 0);
      analogWrite(rightBack, 0);
      analogWrite(leftBack, 0);
      analogWrite(rightForw, 0);
      voiceCommandBackward = false;  // Reset the command flag
    }
```

```
} else if (voiceCommandStop) {
    analogWrite(leftBack, 0);
    analogWrite(rightForw, 0);
    analogWrite(leftForw, 0);
    analogWrite(rightBack, 0);
} else if (voiceCommandTurnLeft) {
    analogWrite(leftBack, 0);
    analogWrite(rightBack, 0);
    analogWrite(leftForw, max_speed/2);
    analogWrite(rightForw, max_speed/2);
    delay(2000);
    analogWrite(leftBack, 0);
    analogWrite(rightBack, 0);
    analogWrite(leftForw, 0);
    analogWrite(rightForw, 0);
    voiceCommandTurnLeft = false;  // Reset the flag after execution
} else if (voiceCommandTurnRight) {
    analogWrite(leftForw, 0);
    analogWrite(rightForw, 0);
    analogWrite(leftBack, max_speed/2);
    analogWrite(rightBack, max_speed/2);
    delay(2000);
    analogWrite(leftBack, 0);
```

```
        analogWrite(rightBack, 0);
        analogWrite(leftForw, 0);
        analogWrite(rightForw, 0);
        voiceCommandTurnRight = false;  // Reset the flag after execution
      }
    }
    else {
      // If no active voice command, use joystick to control the motors
      if (joyposVert < 460 && joyposHorz >= 255 && joyposHorz <= 767 &&
!obstacleDetectedBack) {
        // This is backward
        joyposVert = joyposVert - 460; // This produces a negative number
        joyposVert = joyposVert * -1;  // Make the number positive

        MotorSpeedLeft = map(joyposVert, 0, 460, 0, max_speed);
        MotorSpeedRight = MotorSpeedLeft;

        analogWrite(leftForw, MotorSpeedLeft);
        analogWrite(rightBack, MotorSpeedRight);
        analogWrite(leftBack, 0);
        analogWrite(rightForw, 0);
```

```
    } else if (joyposVert > 564 && joyposHorz >= 255 && joyposHorz <= 767 &&
!obstacleDetectedFront) {
      // This is forward
      joyposVert = joyposVert - 564; // This produces a positive number

      MotorSpeedLeft = map(joyposVert, 0, 460, 0, max_speed);
      MotorSpeedRight = MotorSpeedLeft;

      analogWrite(leftBack, MotorSpeedLeft);
      analogWrite(rightForw, MotorSpeedRight);
      analogWrite(leftForw, 0);
      analogWrite(rightBack, 0);

    } else if (joyposHorz < 255 && joyposVert >= 255 && joyposVert <= 767) {
      // This is left
      joyposHorz = joyposHorz - 255; // This produces a negative number
      joyposHorz = joyposHorz * -1;  // Make the number positive

      MotorSpeedLeft = map(joyposHorz, 0, 255, 0, max_speed);
      MotorSpeedRight = MotorSpeedLeft;

      analogWrite(leftForw, MotorSpeedLeft/2);
      analogWrite(rightForw, MotorSpeedRight/2);
```

```
    analogWrite(leftBack, 0);
    analogWrite(rightBack, 0);

  } else if (joyposHorz > 767 && joyposVert >= 255 && joyposVert <= 767) {
    // This is right
    joyposHorz = joyposHorz - 767; // This produces a positive number

    MotorSpeedLeft = map(joyposHorz, 0, 255, 0, max_speed);
    MotorSpeedRight = MotorSpeedLeft;

    analogWrite(leftBack, MotorSpeedLeft/2);
    analogWrite(rightBack, MotorSpeedRight/2);
    analogWrite(leftForw, 0);
    analogWrite(rightForw, 0);

  } else {
    // If joystick is in neutral position, stop motors
    analogWrite(leftForw, 0);
    analogWrite(rightForw, 0);
    analogWrite(leftBack, 0);
    analogWrite(rightBack, 0);
  }
}
```

```
}

int getDistance(int triggerPin, int echoPin, long timeout) {
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);

  long duration = pulseIn(echoPin, HIGH, timeout);

  if (duration == 0) {
    // If no echo is received within the timeout, return -1
    return -1;
  }

  // Calculate the distance in cm
  int distance = duration * 0.034 / 2;

  return distance;
}

void printDetail(uint8_t type, int value) {
```

```
switch (type) {
  case TimeOut:
    Serial.println(F("Time Out!"));
    break;
  case WrongStack:
    Serial.println(F("Stack Wrong!"));
    break;
  case DFPlayerCardInserted:
    Serial.println(F("Card Inserted!"));
    break;
  case DFPlayerCardRemoved:
    Serial.println(F("Card Removed!"));
    break;
  case DFPlayerCardOnline:
    Serial.println(F("Card Online!"));
    break;
  case DFPlayerUSBInserted:
    Serial.println("USB Inserted!");
    break;
  case DFPlayerUSBRemoved:
    Serial.println("USB Removed!");
    break;
  case DFPlayerPlayFinished:
```

```
  Serial.print(F("Number:"));
  Serial.print(value);
  Serial.println(F(" Play Finished!"));
  break;
case DFPlayerError:
  Serial.print(F("DFPlayerError:"));
  switch (value) {
    case Busy:
      Serial.println(F("Card not found"));
      break;
    case Sleeping:
      Serial.println(F("Sleeping"));
      break;
    case SerialWrongStack:
      Serial.println(F("Get Wrong Stack"));
      break;
    case CheckSumNotMatch:
      Serial.println(F("Check Sum Not Match"));
      break;
    case FileIndexOut:
      Serial.println(F("File Index Out of Bound"));
      break;
    case FileMismatch:
```

```
        Serial.println(F("Cannot Find File"));
        break;
      case Advertise:
        Serial.println(F("In Advertise"));
        break;
      default:
        break;
    }
    break;
  default:
    break;
 }
}

void performActionFront(unsigned long currentTimeFront) {
 // Perform the desired action
 Serial.println("Obstacle Front");
 myDFPlayer.play(9);

 delay(100);

 // Indicate the action has been performed
 actionPerformedFront = true;
```

```
  // Record the time the action was performed
  lastActionTimeFront = currentTimeFront;
}

void performActionBack(unsigned long currentTimeBack) {
  // Perform the desired action
  Serial.println("Obstacle Back");
  myDFPlayer.play(1);

  delay(100);

  // Indicate the action has been performed
  actionPerformedBack = true;

  // Record the time the action was performed
  lastActionTimeBack = currentTimeBack;

}
```

REFERENCES

[1] *All About DC Motor Controllers - What They Are and How They Work. (n.d.)*. (n.d.). All About DC Motor Controllers - What They Are and How They Work.. https://www.thomasnet.com/articles/instruments-controls/dc-motor-controllers/

[2] (n.d.). Amazon.com. https://www.amazon.com

[3] (n.d.). Arduino - Home. https://www.arduino.cc

[4] *Arduino - Joystick*. (n.d.). Arduino Getting Started. https://arduinogetstarted.com/tutorials/arduino-joystick?utm_content=cmp-true

[5] *DC motor: What is it? How does it work? Types, uses*. (n.d.). OEM Manufacturers | OEM Manufacturing Companies | IQS Directory. https://www.iqsdirectory.com/articles/electric-motor/dc-motors.html

[6] *DFPlayer mini Mp3 player - DFRobot Wiki*. (n.d.). wiki_EN_content_page-DFRobot. Retrieved June 10, 2024, from https://wiki.dfrobot.com/DFPlayer_Mini_SKU_DFR0299

[7] (n.d.). Digikey. https://www.digikey.com

[8] (n.d.). DROK. https://www.droking.com

[9] (2024, 9). DroneBot Workshop. https://dronebotworkshop.com

[10] *Gravity: Offline language learning voice recognition sensor for micro:bit / Arduino / ESP32 - I2C & UART*. (n.d.). DFRobot Open-Source Hardware Electronics and Kits. https://www.dfrobot.com/product-2665.html

[12] I2C communication pins in Arduino boards. (n.d.). Linux Hint. https://linuxhint.com/arduino-i2c-communication-pins/#b3

[13] *IBT-2 H-bridge with Arduino*. (2013, December 28). Dr. Rainer Hessmer. https://www.hessmer.org/blog/2013/12/28/ibt-2-h-bridge-with-arduino/

[14] (n.d.). IEEE - The world's largest technical professional organization dedicated to advancing technology for the benefit of humanity. https://www.ieee.org/

[15] *International Organization for Standardization*. (n.d.). ISO. https://www.iso.org/home.html

[16] Kim, H., & Ryu, D. (2006). Smart wheelchair based on ultrasonic positioning system. *Lecture Notes in Computer Science*, 1014-1020. https://doi.org/10.1007/11788713_148

[17] *A literature review on the smart wheelchair systems*. (n.d.). arXiv.org e-Print archive. https://arxiv.org/html/2312.01285v1

[18] *MP3 player with Arduino*. (2018, April 3). Instructables. https://www.instructables.com/MP3-Player-With-Arduino/

[19] *Obstacle avoiding bot using ultrasonic sensor and Arduino*. (2019, December 8). Instructables. https://www.instructables.com/Obstacle-Avoiding-Bot-Using-Ultrasonic-Sensor-and-/

[20] Paslıoğlu, E. (2021, October 15). *How to use Proteus? Guide for beginners*. Medium. https://medium.com/@erenpaslioglu/how-to-use-proteus-guide-for-beginners-31165afd78b9

[21] Sahoo, S. K., & Choudhury, B. B. (2024). Autonomous navigation and obstacle avoidance in smart robotic wheelchairs. *Journal of Decision Analytics and Intelligent Computing*, *4*(1), 47-66. https://doi.org/10.31181/jdaic10019022024s

[22] Schwesinger, D., Shariati, A., Montella, C., & Spletzer, J. (2016). A smart wheelchair ecosystem for autonomous navigation in urban environments. *Autonomous Robots*, *41*(3), 519-538. https://doi.org/10.1007/s10514-016-9549-1

[23] (n.d.). SparkFun Electronics. https://www.sparkfun.com

[24] Storr, W. (2022, August 3). *Pulse width modulation used for motor control*. Basic Electronics Tutorials. https://www.electronics-tutorials.ws/blog/pulse-width-modulation.html

[25] Whitaker, J. C. (2018). *The electronics handbook*. CRC Press.

[26] Y, R. (2021, September 15). *Buck converter*. Electronics Coach. https://electronicscoach.com/buck-converter.html