ECE 445: Senior Design Project Laboratory
Final Report
Oxygen Delivery Robot

Team 27
Sayankar, Rutvik
rutviks2@illinois.edu

Dunican, Aidan
dunican2@illinois.edu

Kalyniouk, Nazar
nazark2@illinois.edu

Teaching Assistant
Subramaniam, Selva
ss170@illinois.edu

May 1, 2024

Abstract

Mobile oxygen delivery is a mostly untapped market for users both elderly and young. This report details a prototype design for a mobile oxygen delivery robot. We utilize a three wheeled omni-directional drivetrain, ultra wideband triangulation, computer vision, and a simple Roomba style bumper system. We remove ourselves from the context of dealing with oxygen related tubing as swivel solutions already exist and focus on the issue of creating a robot that can autonomously and efficiently move an oxygen tank around in an indoor space.

Contents

1. Introduction

1.1 Project Overview

Children's interstitial and diffuse lung disease (ChILD) is a collection of diseases or disorders. These diseases cause a thickening of the interstitium (the tissue that extends throughout the lungs) due to scarring, inflammation, or fluid buildup [1]. This eventually affects a patient's ability to breathe and distribute enough oxygen to the blood. Numerous children experience the impact of this situation, requiring supplemental oxygen for their daily activities. It hampers the mobility and freedom of young children, diminishing their growth and confidence. Moreover, parents face an increased burden, not only caring for their child but also having to be directly involved in managing the oxygen tank as their child moves around.

Given the absence of relevant solutions in the current market, our project aims to ease the challenges faced by parents and provide young children the freedom to explore their surroundings without worry. As a proof of concept for an affordable solution, we propose a three wheeled omni-directional mobile robot capable of supporting a filled M-6 oxygen tank. We plan to implement two localization subsystems to ensure redundancy and enhance child safety. The first subsystem utilizes Ultra-Wideband (UWB) transceivers for triangulating the child's location relative to the robot in indoor environments, this is similar to how Apple AirTags triangulate their location relative to an iPhone [2] (although AirTags use a combination of UWB and Bluetooth triangulation [3]). The second subsystem makes use of a desktop web camera which streams video to a Raspberry Pi where it will leverage open-source object tracking libraries to improve our directional accuracy in tracking a child. The final subsystems will focus on close range object detection.

1.2 Block Diagram

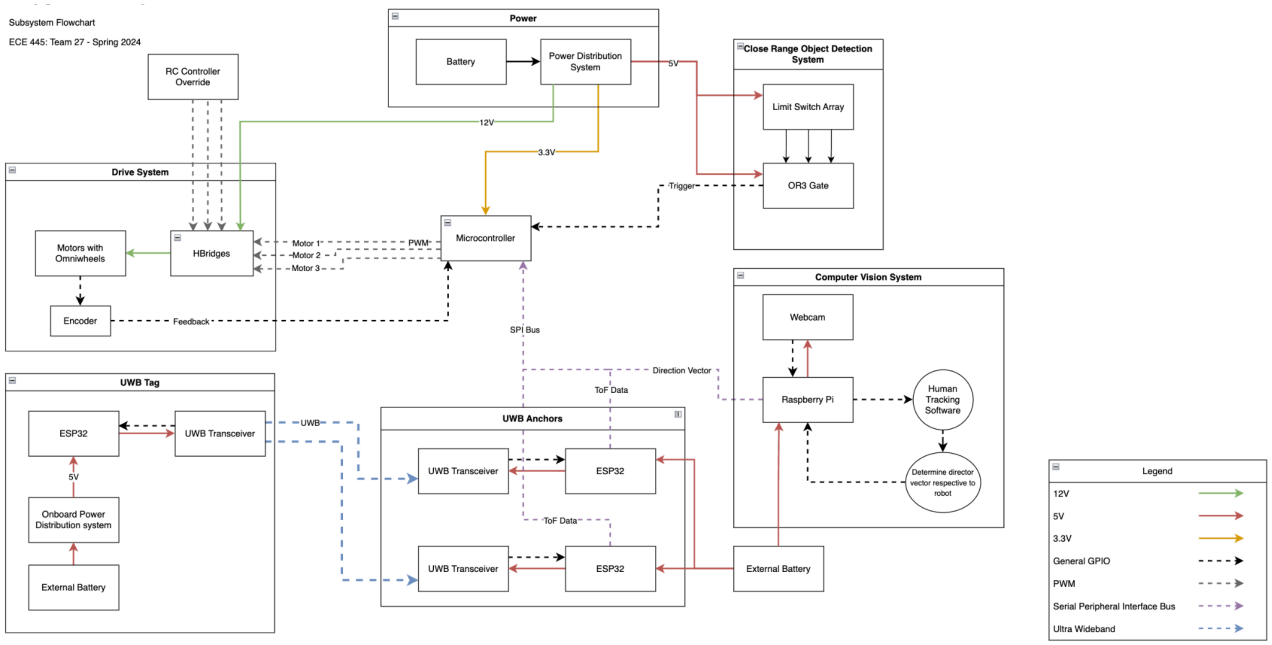

Figure 1.1: High Level Block Diagram

1.2.1 Omniwheel Drivetrain System

We utilize a three wheeled omni-directional wheel drive train. In software we make use of simple matrix multiplication (detailed in section 2.2.1) to derive individual wheel speeds given that we know where we want to go in Cartesian space.

1.2.2 Ultra-Wideband Triangulation System

This system makes use of UWB transceivers in order to triangulate location in an indoor space with high accuracy. The goal was to be able to locate the user in an indoor environment where there maybe be interference with other bluetooth or wifi signals.

1.2.3 Close Range Object Detection System

The Close Range Object Detection subsystem makes use of limit switches with bumpers attached to them arranged in a ring around the robot that will detect when it encounters an object in its path. The goal of this system is to halt the robot and any further movement when a limit switch is pressed. This enhances the safety of the robot, particularly in environments with children. With the expectation that the robot will be utilized inside spaces such as a living room which might have toys scattered around due to younger children being present. To limit the scope of the project we want the robot to simply stop movement upon

collision with any object in it's path and have either the user or a someone supervising come in and clear the area before the robot is to be used again.

1.2.4 Computer Vision System

The Computer Vision subsystem was incorporated into our design to ensure redundancy and robustness. By having a sensor fusion between this and the UWB subsystem, we can ensure that the robot follows the child as desired and add an extra layer of redundancy and safety to our design. The performance requirements of this subsystem are that this subsystem will initiate a corrective measure to the STM32 microcontroller if the child deviates too far from the center of the input frame. Specifically, this system focuses on the degrees away from the center the user currently is. Between the start of this design and the end, nothing was changed in the block level of this subsystem.

1.2.5 Control System

This system is purely regarding how the STM32 microcontroller interacts and controls all our peripheral devices.

1.2.6 Power System

The power system of our robot was designed to take a 12V input and regulate it down to the 5V and 3.3V rails that will be used by all devices on our PCB. Specifically, for our 3.3V rail we needed to ensure a clean supply as it would be powering our STM32 microcontroller, which will be generating all of our PWM signals and reading a lot of timer related signals for which a clean 3.3V is preferred.

2. Design

2.1 Design Procedure

2.1.1 Omniwheel Drivetrain System

We utilize a three wheeled omni-directional wheel drive train. Chosen mainly due to the cost benefit over a similar four wheeled drive train. This type of system is susceptible to drift in the form of yaw, therefore we made sure to include feedback from encoders back into our STM32 microcontroller to allow for drift compensation.

2.1.2 Ultra-Wideband Triangulation System

Our initial design discussion was regarding the type of wireless indoor localization we want to design. Immediately, two types of communication came to mind: UWB and BLE. Ultimately, we chose UWB over BLE due to the following benefits:

- Data transmission: UWB can transmit more data than Bluetooth Low Energy
- Accuracy: UWB can more accurately measure distance and determine position because it uses wideband radio waves and higher frequency bands
- Interference resistance: UWB is less likely to be affected by interference and obstructions

Localization Algorithms

To break it down simply, our goal was to find out the location of a user relative to our robot. We assume the user and robot will be in a valid location in real world space, and therefore can just focus on the user's location relative to our robot.

To localize the user, we need to create a triangle of known lengths with vertices defined by the UWB transceivers. To obtain the lengths of each side we need to perform some ranging between the tag transceiver (the transceiver which the user will have) and the anchor transceivers (which are mounted onto the robot). With the anchors being a set 18" apart (the minimum as required by the DW3000 ICs we were using), we only needed the remaining two sides of our triangle.

There are two ways to range between UWB transceivers. The first is to use Time of Flight (ToF). ToF is a positioning method based on two-way ranging. That means the tag needs to send and receive signals from the anchor several times and then the flight time of the signal between the anchor and the tag can be measured, because radio waves travel at the speed of light, we can calculate the distance between the tag and each anchor given that we know how long it took for a signal to travel between them. Secondly, Time Difference of Arrival (TDoA) is localization based on comparing the time difference between signals and

each anchor and this technique requires an accurate time synchronization function. When using the TDoA method, the UWB tag will send out a poll message, and all the nearby UWB anchors will receive it and record the arrival time. Because the location of anchors is different, they will not receive the message at the same time. We can use these time differences to determine the tag's location.

Implementing an accurate time synchronization function between all of our transceivers increases the complexity of the project ten-fold therefore we decided to use a ToF based ranging method to calculate the distance between each anchor and the tag device.

2.1.3 Close Range Object Detection System

This system is the simplest but definitely required some debate in terms of how we wanted it implemented. We debated between two main types of detection: simple limit switches or ultrasonic sensors. Limit switches don't require explanation, you activate a switch and a signal goes high. The only struggle here would be finding a limit switch that would be suitable for our needs. The alternative was ultrasonic detection which would have required us to use a much larger number of sensors and which would have been much more sensitive. The reason we decided to use limit switches was that they would only stop the robot when an object collides with it. In addition, limit switches allow us to make a 360-degree bumper for the robot with only 3 switches, which would fill in the gap between each wheel. To design the subsystem we decided to go with the 3 Honeywell 101SN11 limit switches as they were the best available limit switch in the ECE Self-service center (put simply, it was the only limit switch that could take any sort of impact without damage). One of the major issues we had when designing this subsystem was how to get the Honeywell limit switches to work. Seeing as these limit switches have since been discontinued, finding proper documentation on them proved to be difficult. We discovered that the output to the limit switches needed a 10k pull-up resistor in order to drive the output high when the limit switch is inactive. This was all thanks to a helpful ebay seller who provided a simple explanation on the function of some old limit switches he was selling (www.ebay.com/itm/333999755279).

2.1.4 Computer Vision System

Alternate approaches to incorporating person detection besides using a Haar cascade which we ultimately chose are as follows.

- Histogram of Oriented Gradients (HOG)
- You Only Look Once (YOLO)
- Single Shot MultiBox Detector (SSD)

Haar cascade was chosen for our final implementation as it is simpler to implement than conventional deep-learning models. It contains pre-trained full body models, a large stepping stone which would be required for our project. For the Haar cascade algorithm, a pre-trained model that detects full bodies is used. Once the body is detected, a bounding box is drawn around the body. The center point of this box and the center point of the screen are compared and this determines how far off the center the body is.

2.1.5 Control System

The details of the control system during design were fairly simple. It mainly consisted of choosing a microcontroller and deciding what pins to assign where. We were limited by what was provided by the ECE supply center, we decided to go with the F103 line of STM32 chips as we believed it had the perfect amount of GPIO and timer pins required for our robot. Our final pin assignments are shown in figure 6.23.

We required individual timers for the following:

- PWM output to motors (can all be synchronized therefore only need one timer)
- Motor 1 Encoder feedback
- Motor 2 Encoder feedback
- Motor 3 Encoder feedback

We required GPIO for the following:

- Limit switch bumper input
- USB lines
- SPI bus
- Chip select for 3 peripherals on the SPI bus
- High Speed External Crystal Oscillator input and output.
- Programmer pins

2.1.6 Power System

In our initial power design we had four requirements:

- Supply a max of 9 A at 12 V per motor therefore 27 A at 12 V
- Supply 5 V to supply around 3.5 A line to power items like a Raspberry Pi running computer vision software, and two ESP32 boards that power the UWB transceiver anchors on the robot, and some other miscellaneous 5V powered ICs throughout our PCB.
- Supply 3.3 V at < 80 mA for our STM32.

We initially began with separate buck converters for each of our power rails. This original design took up lots of space on our PCB and was unnecessary as we only needed one 5 V line rated for higher current. After realizing that our selection of ESP32 boards and Raspberry Pi's do not have the option of power delivery through GPIO pins, we decided to use an external battery pack to supply them power. But we opted to leave our higher current buck converter in the design to leave headroom for the possibility of delivering power to our Raspberry Pi and two ESP32 boards in the future. This led us to designing our final power system which consisted of a 12 V to 5 V buck converter rated for up to 6 A. This was followed with a 5 V to 3.3 V LDO to supply the power to our 3.3 V line. The reason we opted

to go with an LDO for the 3.3v was because it was supplying power to the microcontroller which due to its sensitivity needed to be at a stable 3.3 V ideally without much noise. By opting to go with an LDO we were able to lower the noise floor and improve the line regulation of the supply going to the STM32 and hopefully improve the analog performance of our microcontroller.

To further ensure the performance of the our analog pins in our microcontroller we decided to make use a ferrite bead which suppresses high-frequency electronic noise in electronic circuits. Ferrite beads employ high-frequency current dissipation in a ferrite ceramic to build high-frequency noise suppression devices.

2.2 Design Details

2.2.1 Omniwheel Drivetrain System

Inverse Kinematics of the OWB

In a three-wheeled omni-wheel robot, the three wheels are arranged in a triangular pattern, with one wheel at the front and two wheels at the back. Each wheel has several small omni-directional wheels arranged in a circular pattern, which allows the robot to move in any direction.

To move the robot, each wheel is driven independently using a motor. By varying the speed and direction of each wheel, the robot can move in any direction and rotate around its center point.

The kinematics of a three-wheeled omni-wheel robot can be modeled mathematically the below equations [4] which relate the robot's linear and angular velocities to the speed and direction of each wheel.

We first define for each wheel a direction of rotation V_i, and the resulting movement effected in the x and y axes, V_ix and V_iy as shown below.

Knowing the angle of each wheel from the vertical axis, we can compute the x and y components created by the rotation of each wheel. By summing these components together, we can get the effective motion of the robot in the x, y and yaw axes.

$$\begin{pmatrix} V_x \\ V_y \\ \omega \end{pmatrix} = \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & -\frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}$$

V_x and V_y are the desired linear velocities in the x and y directions, respectively, and omega is the desired angular velocity. V_1, V_2, and V_3 are the velocities of the three wheels, and L is the distance between the robot's center of mass and the wheel axes.

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix} = \begin{pmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & -\frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \\ 1 & 1 & 1 \end{pmatrix}^{-1} \begin{pmatrix} V_x \\ V_y \\ \omega \end{pmatrix}$$

By taking the inverse of our 3x3 component coefficient matrix above, we can solve for our individual motor speeds V_1, V_2 and V_3, given the desired motion in terms of V_x, V_y, and w.

2.2.2 RC Control

For drivetrain testing and part of our demo we used an RC controller. We made use of a Flysky FS-i6 6CH 2.4GHz Radio System RC Transmitter Controller with FS-iA6 Receiver (6.22). While the receiver supports up to six channels, the OWB only needs three channels.

The OWB can move forward and backward like a normal RC car. This motion is represented by V_y, and is mapped to CH1 on the transmitter. While a normal RC car needs a V_y component to turn left or right, the OWB can "yaw" on the spot. This is represented by w, and is mapped to CH3 on the transmitter. The final unique motion the OWB can execute is a "drift" toward the left or right. This motion is represented by V_x and is mapped to CH2 on the transmitter.

Thus, using these three channels, the operator can define the robot's desired V_x, V_y, and w. These desired motion vectors are then fed to our inverse kinematics algorithm, which breaks down our desired motion into an individual wheel speed and direction for each motor.

2.2.3 Ultra-Wideband Triangulation System

The UWB triangulation utilizing double sided two way ranging between the tag device which will be attached to the user. This tag will performing this ranging calculation with each tag device, therefore deriving the lengths of each side of our localization triangle.

Symmetrical double-sided two-way ranging

Symmetrical double-sided two-way ranging (SDS-TWR) is a ranging method that uses two delays that naturally occur in signal transmission to determine the range between two stations [5].

Signal Propagation Delay

When a test packet is transmitted from station A to station B, the propagation delay can be calculated. As the packet travels through space, the difference in time is calculated from when it's sent from the transmitter to the receiver. This time delay is known as the signal propagation delay.

Processing Delay

Following the transmission between station A and station B, station A waits for an acknowledgement from station B. Once station A takes a known amount of time to process the test packet, it generates an acknowledgement and prepares it for transmission. The time it requires to process the acknowledged message is referred to as the processing delay.

Range Calculation

The signal propagation delay and the processing delay are encapsulated in the acknowledgment sent back to station A. On this acknowledgment, station A computes another signal propagation delay. These three values are then used by the algorithm to calculate the range between station A and station B.

Verifying Range Calculation

To verify the accuracy of the previous range calculation, station B sends a test packet to station A and station A sends an acknowledgement to station B once it receives the signal. By repeating this procedure, the two range values are determined and an average can be calculated to verify the accuracy of the distance measurement between these two stations

2.2.4 Close Range Object Detection System

Once the Honeywell 101SN11 limit switch is pressed it drives the output voltage low. This signal is pulled high with a 10 $k\Omega$ resistor (refer to figure 6.4). Then fed directly to the CD74HC00N four channel two input NAND gate in order to invert the signal to high when the switch is pressed. From there the 3 inverted limit switch signals are passed on to a 3 input OR gate which combines the inputs into one single output, which is high when any limit switch is pressed. Next the output is passed to the one of the STM32's FT (5 V tolerant) pins and which triggers an interrupt that will stop the PWM signals that drive the motors from being generated. This in turn halts the motors placing the robot in an idle state until the limit switch is unpressed and the robot is reset.

2.2.5 Computer Vision System

For the software portion of this subsystem, the OpenCV code was written. Figure 6.6 provides an overview of how the algorithm works.

To test the pre-trained Haar cascade classifier, the code was tested with an input image of a child walking down a hallway.

When the body is detected, the distance is found between the center of the body and the center of the image. A distance is calculated by subtracting the two center points and dividing that number by the number of pixels on the screen. A message will be printed to inform the user if the off-center variable is large enough to where a correction will be needed. If no person is detected, a message will be printed. If this value is larger than 9%, a correction will be made. This initial calculator can be seen below. Later testing with the webcam will confirm whether this value is reasonable or will need to be adjusted. Some helpful design parameters regarding this calculation are as follows:

- Resolution of Camera = $1600 * 1200$ pixels
- FOV of Camera = 75 degrees
- Average Shoulder Width of Child = 9 inches = 0.23 m

We expect the child will be a distance of 1 $m \pm 25\%$ (0.75 m to 1.25 m) away from the robot. Figure 6.8 is an image of the minimum and maximum ranges. The red boxes are used to represent the width of the child. To calculate how much of the screen the child will take up, we must first calculate the horizontal width of the FOV 6.9. This is done by using simple trigonometry.

Calculating horizontal FOV width:

$$\text{Half of FOV} = (\frac{75^\circ}{2} = 37.5^\circ)$$

$$x = (1 \text{ meter}) \times \tan(37.5^\circ) = 0.767 \text{ meters}$$

Now, multiply this distance by 2, and the result is the horizontal FOV width.
Horizontal FOV Width = 0.767 meters × 2 = 1.534 meters
Now to calculate how much of the screen the target takes up with the current values.

$$\text{Percent of screen} = \frac{\text{Average Shoulder Width of Child}}{\text{Horizontal FOV Width}} = 15\%$$

Since we want to stay within this range, the tolerance for the Percent of Screen taken up by the child will be $15\% \pm 3\%$. The center of the child and the center of the screen will be compared to determine if a correction will be made. Therefore, we divide 15% by 2% to get 7.5% and add 1.5% to that value, so if the difference between the two midpoints is more than 9%, then a correction will be made.

With input images working, we tested the code with a video input from the web camera. Using similar code as before, it was modified to capture frames from a live video stream. First, we used a MacBook Pro web camera as we tested our code in a jupyter notebook, then went on to use the Logic Pro 9000 web camera. Similar to before, when a full body is detected, a bounding box is drawn around the person. The offset was calculated and printed on the screen.

This code was modified further to ensure that there are no error detections. This was done by checking for the largest body in the frame so that no accidental object in the background would be flagged as a body. This greatly increased the accuracy of the detection and helped to only draw a bounding box around the body.

Originally, our intention was to flash an SD card with an operating system to download OpenCV on a Raspberry Pi 3 Model B. However, we encountered difficulties when attempting to boot it. Despite multiple troubleshooting attempts, this issue persisted so we opted to utilize a spare Raspberry Pi 3 Model A+. This model was compatible with the flashed SD card and allowed us to continue. While this issue was resolved, issues arose when trying to download OpenCV. Since this was a slightly older model than the previous Raspberry Pi, the limited RAM struggled to download OpenCV since it is very resource intensive. Due to time constraints and the fact that this wasn't one of the high-level requirements, efforts were focused on ensuring our high-level requirements were met. In the future, a stronger Raspberry Pi, such as the Raspberry Pi 4, would be a better option.

If OpenCV were to work on the Raspberry Pi, the next step would've been to set up communication between the Raspberry Pi and STM32 over SPI. Once communication was established, a sensor fusion of the computer vision and ultra-wideband sensors. This would ensure the desired child-tracking that we would want our omni wheel robot to exhibit.

2.2.6 Control System

The pinout of our STM32 was detailed in figure 6.23. We implemented system interrupts as shown in figure 6.24.

In order to setup our SPI bus we took a register setting approach. Therefore, we write and access registers directly to read SPI bus data, enable and write to the bus.

See figures 6.25, 6.26, 6.27, 6.28 and 6.29 for a brief overview of the setup within the STMCubeIDE. Comments are also included within the code which help describe the process. You can also refer to figure 6.30 for detail on how the motor control through PWM was setup.

2.2.7 Power System

Our system is powered from a 2200mAh battery (which we later acknowledge was a limiting factor in our final performance). To prevent excessive current draw from our motors and ensure that we have a method in hardware that prevents the motors running at full power, we integrated 9 A PTC Resettable Fuses on each power line connected to a motor. PTC fuses are made of a material that increases its internal resistance as the temperature increases, therefore if we are drawing too much current we increase the resistance on the line and reduce the current being supplied to the motors. Once the current or high temperature is removed, the polymer cools and returns to its original state, restoring the fuse to a low-resistance state.

From there the 12 V from the battery is stepped down to 5 V by our buck converter (6.10). We opted to go for a 6 A buck converter in order to allow for the possibility of directly powering the Raspberry Pi and UWB subsystem from the board without needed a redesign of the power system.

After the battery voltage is dropped down to 5 V it is dropped down again to 3.3 V through an LDO (6.11). This 3.3 V is used to power the STM32 which is the main processor of our robot. We chose to go with an LDO due to its lowered noise floor compared to a buck converter and better line regulation which will ensure that the voltage of the STM32 is kept constant and will not be affected by any supply noise that could result negatively impact our sensitive analog processing.

Another helpful debugging feature of our power subsystem is the addition of a jumper connector that allows for us to choose how to supply 5 V to our board. This was added in order to allow us to be debug our microcontroller and any peripherals without needed to power the motors. We provided the option to select to supply 5 V from the 12 V to 5 V buck converter, an external 5 V source, or through the USB Micro B port that was included on the board.

3. Verification

3.1 Omniwheel Drivetrain System

For our OWB drivetrain our goal was to reliably set the speed and control RPM of motors. Following that, we wanted to maintain speed of 3 MPH under full load.

We could tell from our encoder feedback that we were able to control with RPM, and unfortunately without a bigger battery we couldn't test the max speed of the robot. But without a bigger battery we were able to clearly move the robot in a desired direction but not move over 3 MPH.

3.2 Ultra-Wideband Triangulation System

For the UWB triangulation our goal was to utilize ToF to create a triangle of known lengths with vertices defined by each UWB transceiver. We tested creating a large triangle and comparing the serial read out to the read world distances. We also tested closer distances with a ruler. For all of our verification we were within tolerance.

3.3 Close Range Object Detection System

In order to verify our close range object detection subsystem we included a test point on the output of the OR gate (6.5) which allowed us to be able to check the voltage level going into the STM32 when the limit switches were pressed.

3.4 Computer Vision System

A requirement of the Computer Vision subsystem was to utilize OpenCV or OpenPose to determine the angle of the child respective of the robot. To verify this requirement, the Raspberry Pi must be plugged into the monitor and display the video feed with an overlaid box over the body. In addition to this overlay, a value would be shown to represent how far off the body is from the center of the screen. As seen in figure 6.17, this verification is fulfilled therefore confirming that this requirement was successfully implemented.

3.5 Control System

When connecting all of our peripherals together while our STM timers are active and trying to control the motors, we run into some issues with our SPI bus. We occasionally skip bits, refer to figure 6.18.

3.6 Power System

In order to be able to appropriately test our power system we included test points attached at the output of each voltage line to make it easier for us to validate. Starting off with our 12 V line we receive a voltage that is around 12 $V \pm 5\%$ which is clearly showing in the figure 6.19.

Next with our 5 V line we wanted to achieve a similar margin of $\pm 5\%$ which we were able to achieve as shown in figure 6.20.

Lastly, we also had a test point for our 3.3 V line which similarly we wanted to have a margin of $\pm 5\%$ which was met with a significantly lower margin as seen in figure 6.21.

4. Costs

Labor costs for each team member is calculated with the following equation

$$\text{Estimated hourly rate} * \text{actual hours spent} = \text{individual labor cost} \tag{4.1}$$

Given current ECE graduate pay, this cost comes out to

$$\$44/hr * 137\ hrs = \$6028/person$$

$$\$6028/person * 3 \text{ team members} = \$18084$$

Labor costs for each the machine shop is calculated using equation 4.1. Estimating the average pay for the ECE machine shop worker, this cost comes out to

$$\$60/hr * 40\ hrs = \$2400$$

With that in mind, the robot cost is roughly as follows.

Item	**Quantity**	**Cost**
PCB BOM	1	$62.03
PCB Manufacturing	1	$40.38
3D Printed Parts	1	$5
Omni-Wheels	3	$78
Modern Robotics 12VDC Motor	3	$44.97
4-Pos JST PH to 4-Pos JST XH Adaptor (Encoders)	4	$11.56
Victor 888 Speed Controller	3	$149.97
QuickCam Pro 9000 Webcam	1	$74.99
ESP32UWB DW3000 (Ultra-Wideband transceiver)	3	$131.40
Raspberry Pi 4 1GB Model B	1	$35
Machine Shop Parts	-	$74
Robot Subtotal	-	**$707.30**
Machine Shop Labor	-	$2400
Team Member Labor	-	$18084
Total		**$21191.30**

Figure 4.1: Cost Breakdown

5. Conclusions

This project has effectively fulfilled the high-level requirements we proposed at the beginning of the semester. Key successes include the drive train, the UWB localization, and close-range object detection. These objectives have been successfully achieved as outlined in the previous section's R & V tables. Some challenges that we faced were the small battery size preventing a sufficient current supply for extended use, SPI bus timing was sometimes skipping bits and prevented reliable communication between peripherals, we would have liked to print a full 360-degree bumper if we have a larger 3D printer. Further work would include allocating more time for SPI bus debugging and choosing a larger battery.

5.1 Ethics and Safety

5.1.1 Ethical Considerations

As we continue through the development of our project, we are unwavering in our dedication to abide by the ethical and safety principles outlined by the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE). As we embark on this project, we pledge our commitment to adhere to these standards, ensuring that our actions and choices uphold the highest level of professionalism and integrity.

As outlined in Section I of the IEEE Code of Ethics, we pledge to "uphold the highest standards of integrity, responsible behavior, and ethical conduct in professional activities" [6]. We will prioritize safety in our design and adhere to ethical design practices. Educating the parents and caregivers about the robot's use and limitations will allow for informed decision-making. Following relevant laws and regulations regarding this technology will also be a high priority.

In the same Code of Ethics, outlined in Section III, we pledge to "strive to ensure this code is upheld by colleagues and co-workers" [6]. We will support each other in ethical conduct and foster a culture of ethical behavior. Open communication will be established and encouraged to raise concerns and provide guidance to team members.

5.1.2 Safety Considerations

This project aligns with the safety principles outlined in the ACM Code of Ethics and Professional Conduct. Safety remains our number one priority and as outlined in Section 1.2 [7], we will avoid negative consequences, especially when those consequences are significant and unjust. We will take careful consideration of potential impacts and minimize harm. In the context of this project, we will ensure that the robot's design and operation prioritizes safety, especially to the children this product aims to assist. We will work to analyze potential risk and consider the robot's mobility and interaction with its environment.

As well as promoting safety, privacy is also a very important guideline that will be followed, which is outlined in Section 1.6. As professionals, we must safeguard the personal information of our users, especially if it involves children. As it relates to this project, we will ensure that no data will be collected and stored in an external location. Data collection will be minimized to only what is necessary for the robot to operate.

One aspect of our project where safety must be considered is regarding the use of lithium batteries. We acknowledge the potential risks associated with the misuse of lithium batteries. We are committed to following the safety guidelines associated with the batteries we plan on using. More specifically, maintaining the battery's temperature within the recommended range. Also, we are dedicated to the responsible disposal of batteries to ensure sustainability.

Since we are incorporating motors into our design, we will deploy essential control systems to mitigate potential hazards such as collisions with the environment. Safe operation will be ensured with the use of sensors, vision systems, and warning systems.

6. Figures and Tables

6.1 Inverse Kinematics

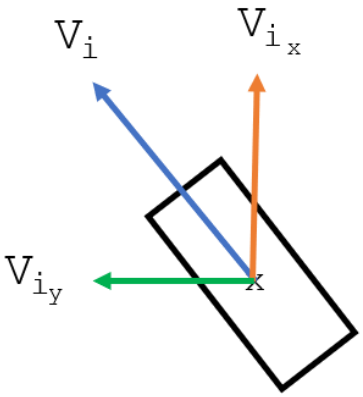

Figure 6.1: Vector Definitions

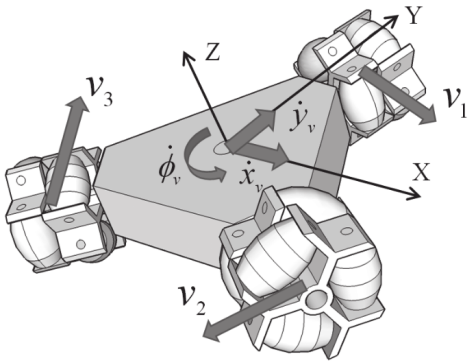

Figure 6.2: OWB drive vectors [8]

6.2 Double-sided Two Way Ranging

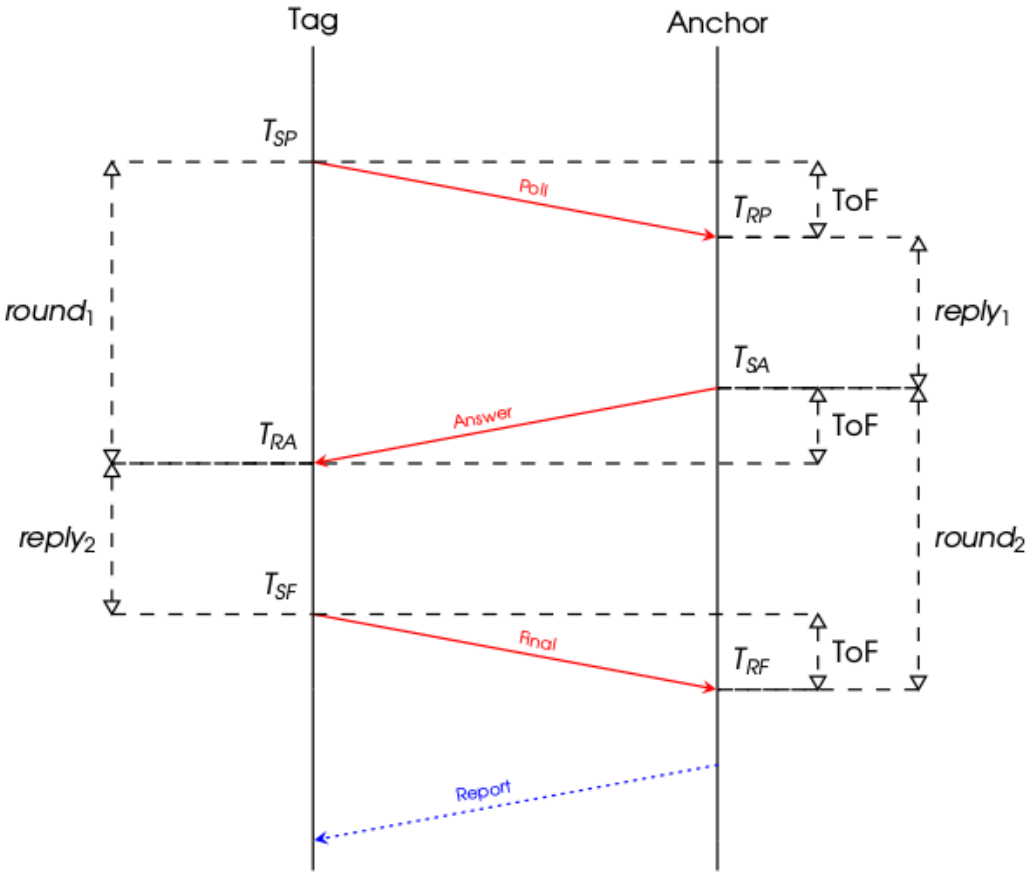

Figure 6.3: Symmetrical Double-sided Two-way Ranging

6.3 Limit Switch Circuit

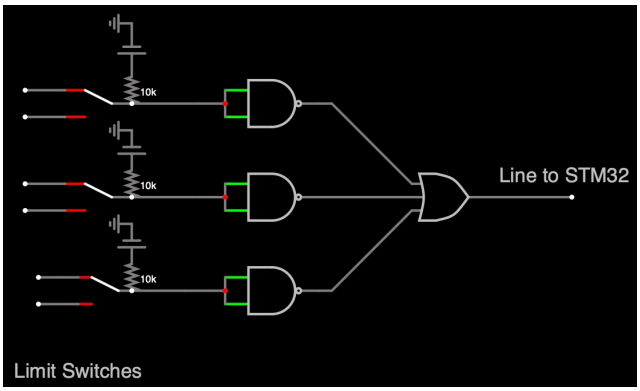

Figure 6.4: Limit Switch Circuit

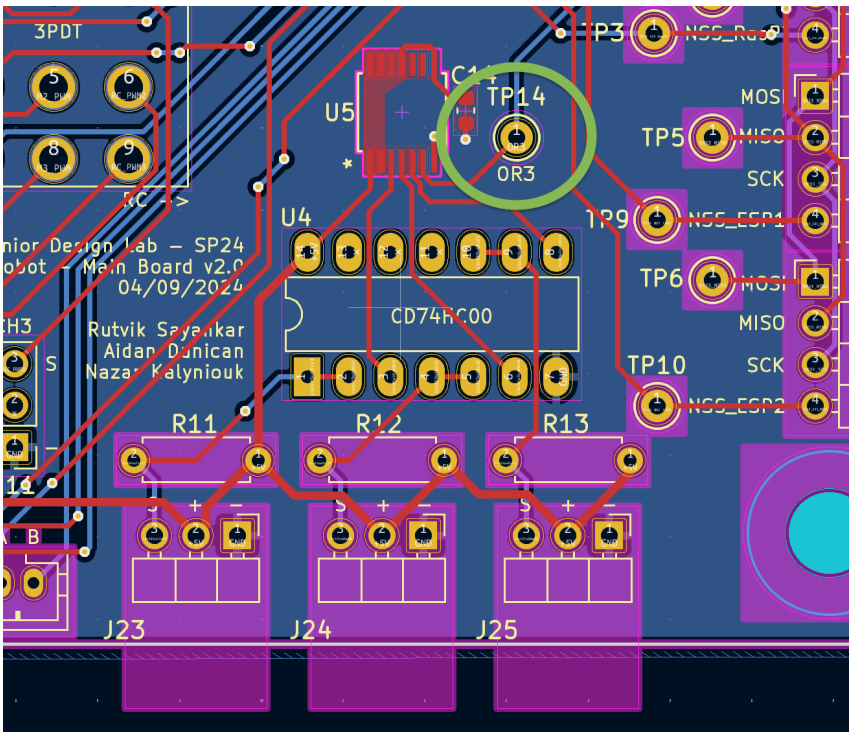

Figure 6.5: PCB OR3 Test Point

6.4 Open CV Images

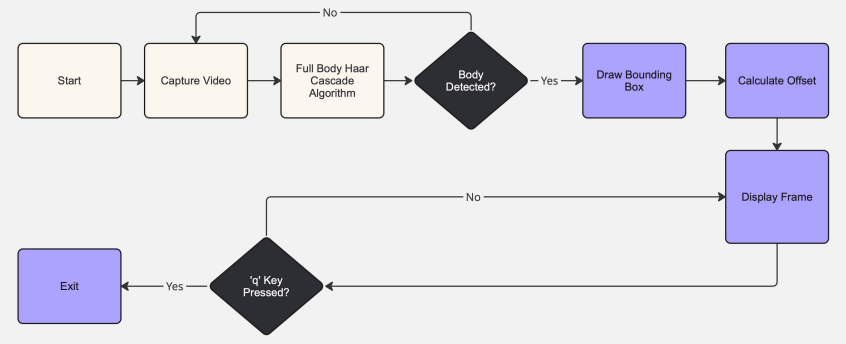

Figure 6.6: OpenCV Algorithm Flowchart

Figure 6.7: OpenCV Performance Test

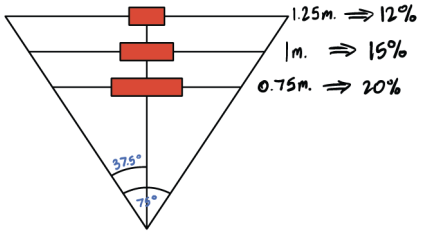

Figure 6.8: Percent of screen calculations for OpenCV algorithm

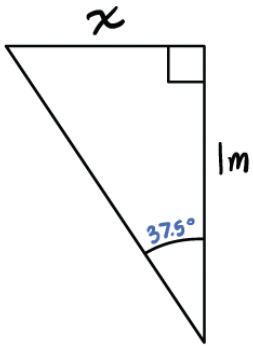

Figure 6.9: Calculation of Horizonatal FOV Width

6.5 Power Delivery

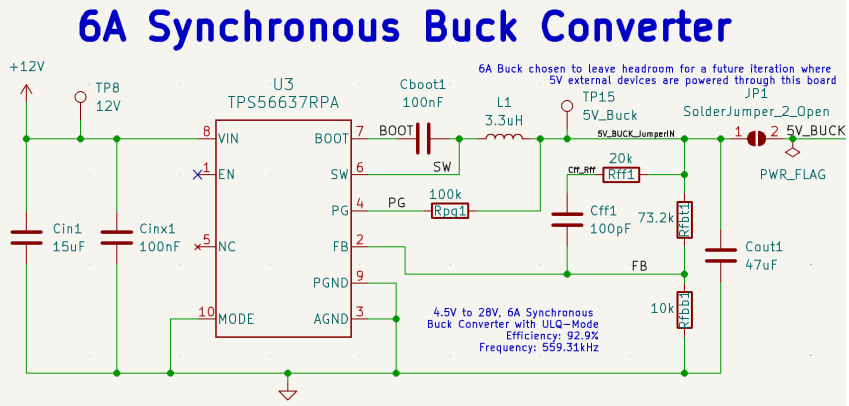

Figure 6.10: 5 V 6 A Buck Converter

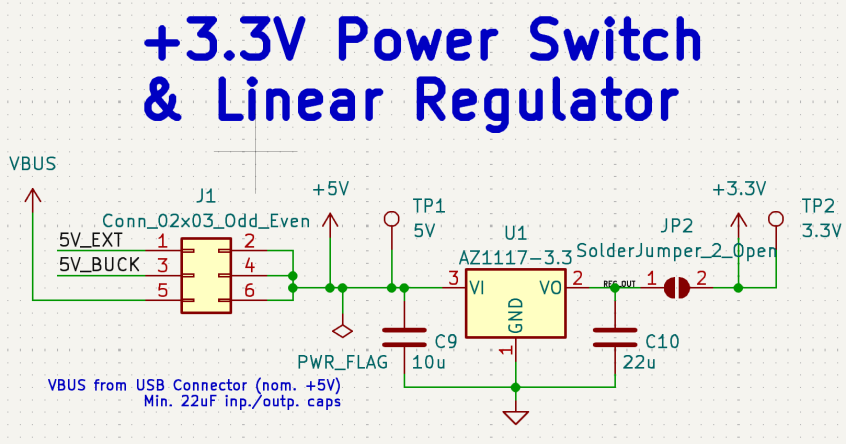

Figure 6.11: 3.3 V Low Dropout Regulator

6.6 UWB Triangulation

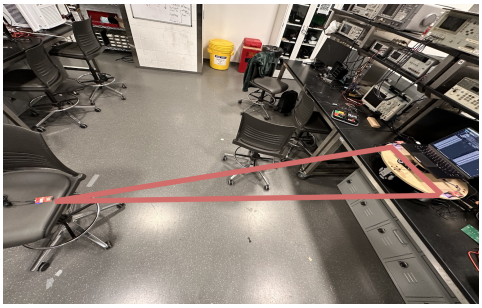

Figure 6.12: UWB Triangle Laid out

Figure 6.13: UWB Serial Read Out

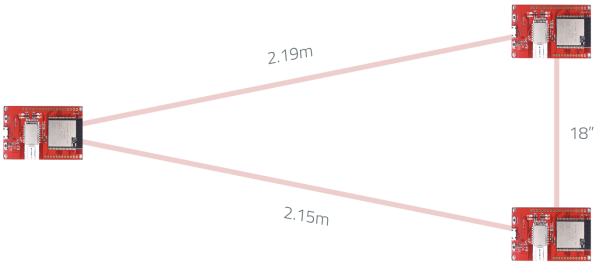

Figure 6.14: UWB Triangle with Distances

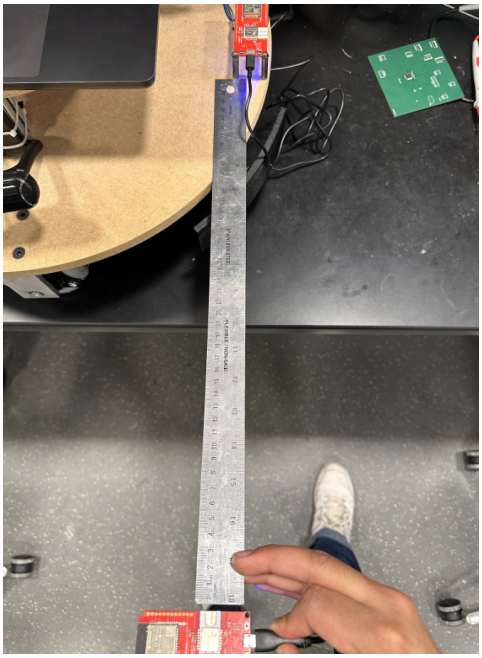

Figure 6.15: 45” Ruler between UWB Transceiver

```
11:52:33.415 -> DIST: 0.42 m
11:52:39.447 -> DIST: 0.44 m
11:52:41.423 -> DIST: 0.43 m
11:52:42.447 -> DIST: 0.44 m
11:52:44.456 -> DIST: 0.41 m
```

Figure 6.16: UWB Serial Read Out

6.7 Computer Vision Verification

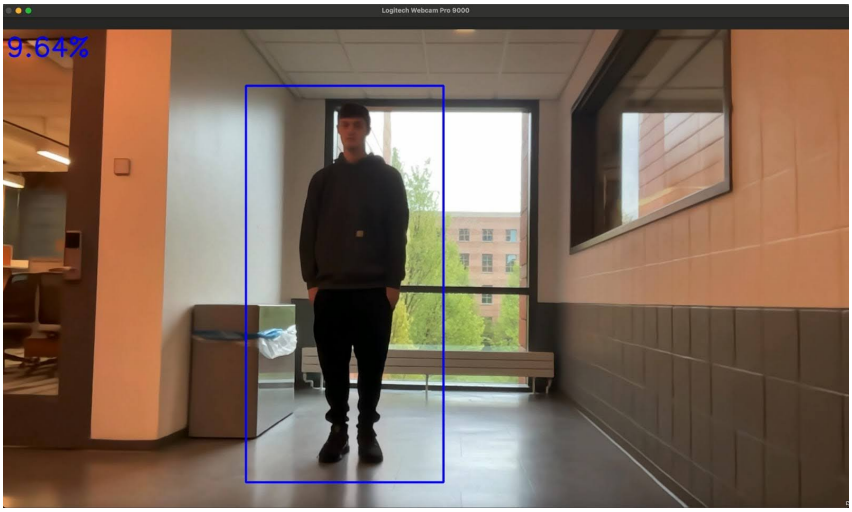

Figure 6.17: Computer Vision Verification

6.8 Control System Verification

Requirement

- Using direction and angle from UWB system, and angle from computer vision system we should be able to compute a direction vector for the robot to move.

Main Issue

SPI Bus skipping bits

- 0xCE = 11001110
- 0x67 = 01100111
- 0xE0 = 11100000
- 0xC = 1100

```
12:55:19.456 -> Received Data: E0
12:55:19.456 -> Received Data: E0
12:55:19.947 -> Received Data: E0
12:55:20.474 -> Received Data: C
12:55:20.969 -> Received Data: CE
12:55:20.969 -> Received Data: CE
12:55:21.464 -> Received Data: CE
12:55:21.464 -> Received Data: CE
12:55:21.959 -> Received Data: CE
12:55:21.959 -> Received Data: CE
12:55:22.450 -> Received Data: CE
```

```
uint8_t data[1];
uint8_t address = 0x67;
data[0] = address;

CS_Enable ();  // pull the pin low
SPI_Transmit (data, 1);  // send test
SPI_Receive (RxDataTest, 1);  // receive 1 byte of  test data
CS_Disable ();  // pull the pin high
HAL_Delay(500);
```

Figure 6.18: SPI Bus Skipping Bits

6.9 Power System Verification

Figure 6.19: 12 V Power Line

Figure 6.20: 5 V Power Line

Figure 6.21: 3.3 V Power Line

6.10 RC Controller

Figure 6.22: Channel Distribution

6.11 STM32 Pinout

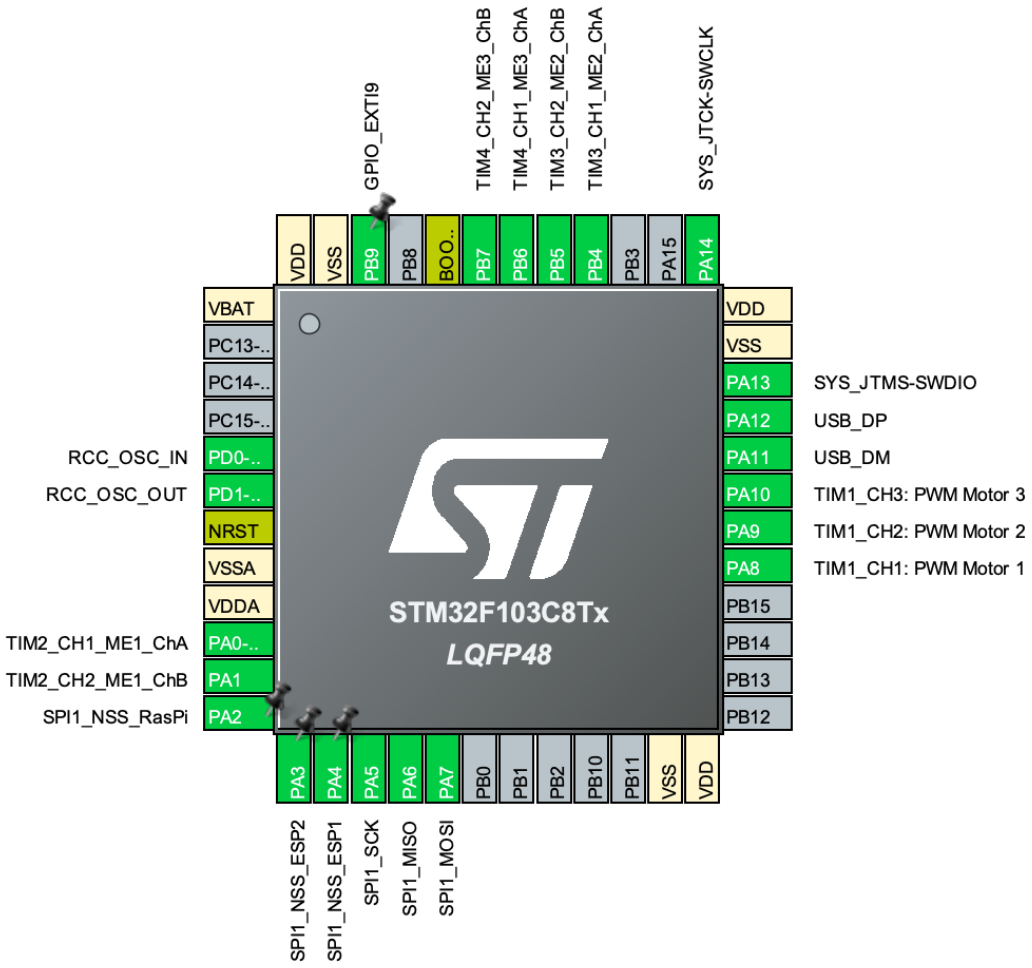

Figure 6.23: STM32F103C8T6 Pinout

```
1         void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) {
2           if((GPIO_Pin == GPIO_PIN_9)) {
3             if((HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_9) == GPIO_PIN_SET)){
4                 HAL_TIM_PWM_Stop(&htim1, TIM_CHANNEL_1);
5                 HAL_TIM_PWM_Stop(&htim1, TIM_CHANNEL_2);
6                 HAL_TIM_PWM_Stop(&htim1, TIM_CHANNEL_3);
7                 }
8           }
9           else{
10            __NOP();
11          }
12        }
```

Figure 6.24: Limit Switch Interrupt Functionality

```
1         void SPIConfig (void){
2           RCC->APB2ENR |= (1<<12);  // Enable SPI1 CLock
3           SPI1->CR1 |= (1<<0)|(1<<1);   // CPOL=1, CPHA=1
4           SPI1->CR1 |= (1<<2);  // Master Mode
5           SPI1->CR1 |= (3<<3);  // BR[2:0] = 011: fPCLK/16, PCLK2 = 80MHz,
                SPI clk = 5MHz
6           SPI1->CR1 &= ~(1<<7);  // LSBFIRST = 0, MSB first
7           SPI1->CR1 |= (1<<8) | (1<<9);  // SSM=1, SSi=1 -> Software Slave
                Management
8           SPI1->CR1 &= ~(1<<10);  // RXONLY = 0, full-duplex
9           SPI1->CR1 &= ~(1<<11);  // DFF=0, 8 bit data
10          SPI1->CR2 = 0;
11        }
```

Figure 6.25: SPI Config Setup

6.12 STM32 Code

```
1         void SPI_Transmit (uint8_t *data, int size){
2
3           /************** STEPS *****************
4           1. Wait for the TXE bit to set in the Status Register
5           2. Write the data to the Data Register
6           3. After the data has been transmitted, wait for the BSY bit to
                reset in Status Register
7           4. Clear the Overrun flag by reading DR and SR
8           ************************************************/
9
10          int i=0;
11          while (i<size){
12            while (!((SPI1->SR)&(1<<1))) {};  // wait for TXE bit to set ->
                  This will indicate that the buffer is empty
13            SPI1->DR = data[i];  // load the data into the Data Register
14            i++;
15          }
16
17        /*During discontinuous communications, there is a 2 APB clock period
              delay between the
18        write operation to the SPI_DR register and BSY bit setting. As a
              consequence it is
19        mandatory to wait first until TXE is set and then until BSY is
              cleared after writing the last
20        data.
21        */
22          while (!((SPI1->SR)&(1<<1))) {};  // wait for TXE bit to set ->
                This will indicate that the buffer is empty
23          while (((SPI1->SR)&(1<<7))) {};  // wait for BSY bit to Reset ->
                This will indicate that SPI is not busy in communication
24
25          //  Clear the Overrun flag by reading DR and SR
26          uint8_t temp = SPI1->DR;
27                  temp = SPI1->SR;
28        }
```

Figure 6.26: SPI Transmit Setup

```
1         void SPI_Receive (uint8_t *data, int size){
2           /************** STEPS *****************
3           1. Wait for the BSY bit to reset in Status Register
4           2. Send some Dummy data before reading the DATA
5           3. Wait for the RXNE bit to Set in the status Register
6           4. Read data from Data Register
7           ******************************************/
8
9           while (size){
10            while (((SPI1->SR)&(1<<7))) {};  // wait for BSY bit to Reset ->
                  This will indicate that SPI is not busy in communication
11            SPI1->DR = 0;  // send dummy data
12            while (!((SPI1->SR) &(1<<0))){};  // Wait for RXNE to set -> This
                   will indicate that the Rx buffer is not empty
13            *data++ = (SPI1->DR);
14            size--;
15          }
16        }
```

Figure 6.27: SPI Receive Setup

```
1         void GPIOConfig (void){
2           RCC->APB2ENR |=  (1<<2);  // Enable GPIOA clock
3           GPIOA->CRL = 0;
4           GPIOA->CRL |= (11U<<20);   // PA5 (SCK) AF output Push Pull
5           GPIOA->CRL |= (11U<<28);   // PA7 (MOSI) AF output Push Pull
6           GPIOA->CRL |= (1<<26);    // PA6 (MISO) Input mode (floating)
7           GPIOA->CRL |= (3<<16);    // PA4 used for CS, GPIO Output
8         }
```

Figure 6.28: SPI GPIO Chip Select Setup

```
1         void SPI_Enable (void){
2           SPI1->CR1 |= (1<<6);   // SPE=1, Peripheral enabled
3         }
4
5         void SPI_Disable (void){
6           SPI1->CR1 &= ~(1<<6);   // SPE=0, Peripheral Disabled
7         }
8
9         void CS_Enable (void){
10          GPIOA->BSRR |= (1<<9)<<16;
11        }
12
13        void CS_Disable (void){
14          GPIOA->BSRR |= (1<<9);
15        }
16
17        uint8_t RxData[3]; // receive SPI data bytes
```

Figure 6.29: SPI Enable and Disable Setup

```
1         // Maps PWM speed between a range
2         uint32_t MAP(uint32_t au32_IN, uint32_t au32_INmin, uint32_t
              au32_INmax, uint32_t au32_OUTmin, uint32_t au32_OUTmax){
3             return ((((au32_IN - au32_INmin)*(au32_OUTmax - au32_OUTmin))/(
                  au32_INmax - au32_INmin)) + au32_OUTmin);
4         }
5
6         // Maps speed between a limit
7         int mapChannel(int speed, int minLimit, int maxLimit){
8           return MAP(speed, -20, 20, minLimit, maxLimit);
9         }
10
11        // PWM Timer Setup
12        MX_TIM1_Init(); // Initialize all configured peripherals
13
14        HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1); // Setting up PWM Timer
15        HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_2); // Setting up PWM Timer
16        HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_3); // Setting up PWM Timer
17
18        HAL_TIM_Encoder_Start_IT(&htim2, TIM_CHANNEL_ALL); // Setting up
              Encoders
19        HAL_TIM_Encoder_Start_IT(&htim3, TIM_CHANNEL_ALL);
20        HAL_TIM_Encoder_Start_IT(&htim4, TIM_CHANNEL_ALL);
21
22        while(1){
23          htim1.Instance->CCR1 = mapChannel(M1_Speed, 1100, 1884); // current
                , minimum, maximum, default
24          htim1.Instance->CCR2 = mapChannel(M2_Speed, 1100, 1884);
25          htim1.Instance->CCR3 = mapChannel(M3_Speed, 1100, 1884);
26
27          HAL_Delay(3000);
28        }
```

Figure 6.30: PWM Setup and Motor Control

A. Appendix

A.1 Terms and Keywords

- ChILD - Children's Interstitial and Diffuse Lung Disease
- PCB – Printed Circuit Board
- OWB - Three-Wheeled Omni-Wheel Bot
- UWB - Ultra-wideband is a radio technology that can use a very low energy level for short-range, high-bandwidth communications over a large portion of the radio spectrum.
- ESP32 - A series of low-cost, low-power system-on-a-chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth.
- RPM - Revolutions per minute
- GPIO - General-purpose input/output
- SPI - Serial Peripheral Interface
- PWM - Pulse-width modulation
- MPH - Miles per hour
- OpenCV - (Open Source Computer Vision Library) is an open source computer vision and machine learning software library.
- TDoA - Time Difference of Arrival
- ToF - Time of Flight
- BOM - Bill of materials
- BLE - Bluetooth Low Energy
- LDO - Low Dropout Voltage Regulator
- PTC - Positive Temperature Coefficient, PTC fuses are made of a material that increases its internal resistance as the temperature increases.

Bibliography

[1] "Childhood interstitial lung disease," Mar 2022. [Online]. Available: https://www.nhlbi.nih.gov/health/childhood-interstitial-lung-diseases

[2] "Ultra wideband," Mar 2023. [Online]. Available: https://discussions.apple.com/thread/254750118?sortBy=best

[3] "Find your keys, backpack, and more with airtag and find my," Feb 2024. [Online]. Available: https://support.apple.com/en-us/109021

[4] R. T. Yunardi, D. Arifianto, F. Bachtiar, and J. I. Prananingrum, "Holonomic implementation of three wheels omnidirectional mobile robot using dc motors," vol. 2, no. 2, 2021, pp. 65–71.

[5] I. C. Society, "Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (lr-wpans)," *IEEE Standard for Information technologyTelecommunications and information exchange between systems*, vol. 802, no. 15.4a, 2006.

[6] "Ieee code of ethics," Jun 2020. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html

[7] "Acm code of ethics and professional conduct," Jun 2018. [Online]. Available: https://www.acm.org/code-of-ethics

[8] M. Wada and T. Hirama, "Research for an omnidirectional mobile robot using acrobat, the active-caster with a dual-ball transmission," *Transactions of the Society of Instrument and Control Engineers*, vol. 51, pp. 400–408, 2015.