TagAlong Accessibility Robot

By:

Deniz Caglar
(dcaglar2@illinois.edu)

Okan Kocabalkanli
(okan2@illinois.edu)

Jirawatchara Tanthiptham
(jt20@illinois.edu)

Final Report for ECE 445, Senior Design, Spring 2023

TA: Raman Singh

Professor: Viktor Gruev

1 May, 2023

Project No. 36

Abstract

TagAlong is a cargo carrying robot with user-following capabilities designed to meet the needs of the elderly and differently abled demographics. TagAlong is successfully capable of carrying more than 3 kgs of cargo while tracking and following a user-specific QR-code at distances of upwards of 1 m, aiming to maintain a distance of 0.5 m at speeds of up to 0.8 m/s. TagAlong reliably operates in artificially lit indoor environments and across level ground.

Contents

1 Introduction

1.1 Motivation

The ability to carry objects is one that provides a great degree of convenience in our lives and is even crucial in many activities designed around their use. However, many experience difficulty or are entirely unable to carry items by themselves with the key demographics studied for this project being the elderly and the differently abled.

Both key demographics are projected to grow substantially in the coming years, indicating a crucial and growing demand for innovations aimed towards improving their standards. With regards to the elderly demographic, the World Health Organization predicts a global proportion of population over 60 to rise to 22% by 2050. With regards to the differently abled demographic, within the United States alone, studies estimate 1.6 million individuals living with limb loss as of 2005 with the number projected to double to 3.6 million individuals by 2050.

These difficulties reduce the ability of affected individuals to live an independent and active lifestyle, requiring caretakers or mobility scooters to assist in mundane tasks such as commuting or shopping. This project aims to provide a solution which allows for active movement while remaining more portable and affordable than aforementioned alternatives.

1.2 Solution

Our solution is to create a path-finding robot that will follow the individual. A static mounted Xbox 360 camera will be used to estimate the distance and bearing of the person of interest by tracking an article of clothing printed with QR code via OpenCV. Pathfinding to the user

The finalized structure of the robot is depicted below in figure 1.

Figure 1a) Right profile of robot

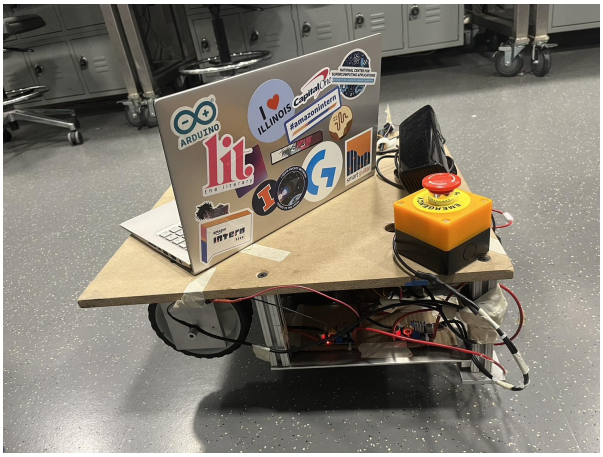

Figure 1b) Front profile of robot

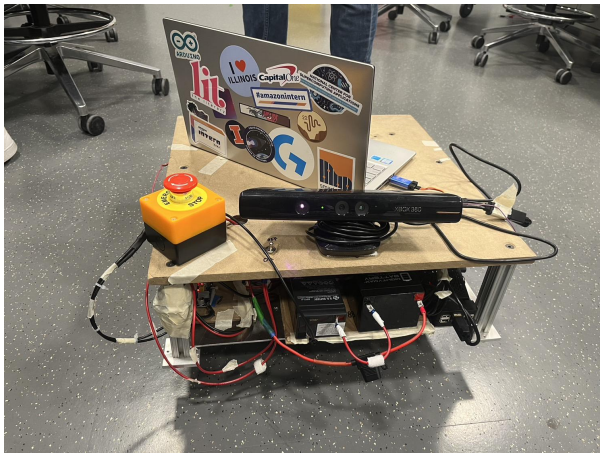

Figure 1c) Left profile of robot

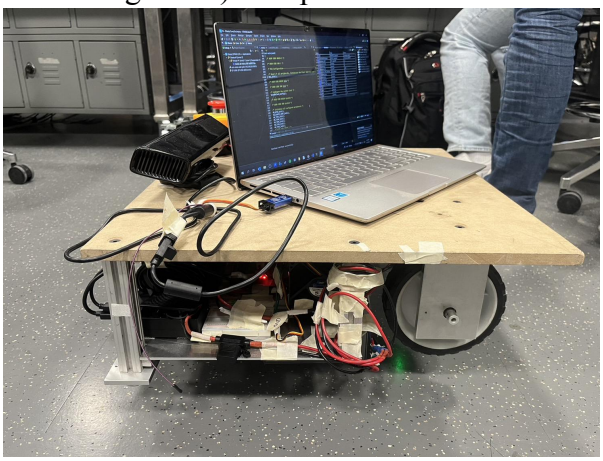

Figure 1d) Back profile of robot

Figure 1) Profile of robot from four orientations

1.3 High Level Requirements

To succeed, the robot must achieve the following criteria:

- The robot should be able to consistently follow the person of interest, designated by a printed QR code, over level ground in artificially lit environments
- The robot will be able to locate the person of interest from distances of at least 1 meter
- The robot will be able to maintain a distance of at least 0.5 meters from the person of interest
- Under constant change in speed, the robot should be able to follow in a straight line.
- The robot should also be able to carry a load of 3 kg over level ground
- The robot should be capable of moving at speeds of up to 0.5 miles per hour (0.8 km/h, 0.22 m/s)
- The robot is able to stop promptly and safely in emergency situations
- The robot is able to run at full speed for a minimum of 1.5 hours

2 Design

2.1 Design Procedure

As a whole, the different subsystems were designed in a manner which allowed for each subsystem to be worked separately and independently with the greatest ease. Given the requirements of the course, the design was built around the MCU subsystem as the base. Our MCU was first selected based on an estimation of the required performance to achieve our high level goals, after which other components were selected according to the capabilities of the MCU. As a whole much of the present design originated as a result of features being removed from a vastly more ambitious initial design. An alternative and likely preferable approach would be to iteratively scale up from a core design to minimize the number of possible causes when debugging and reduce wasted expenditures on parts which proved to be unnecessary in later stages.

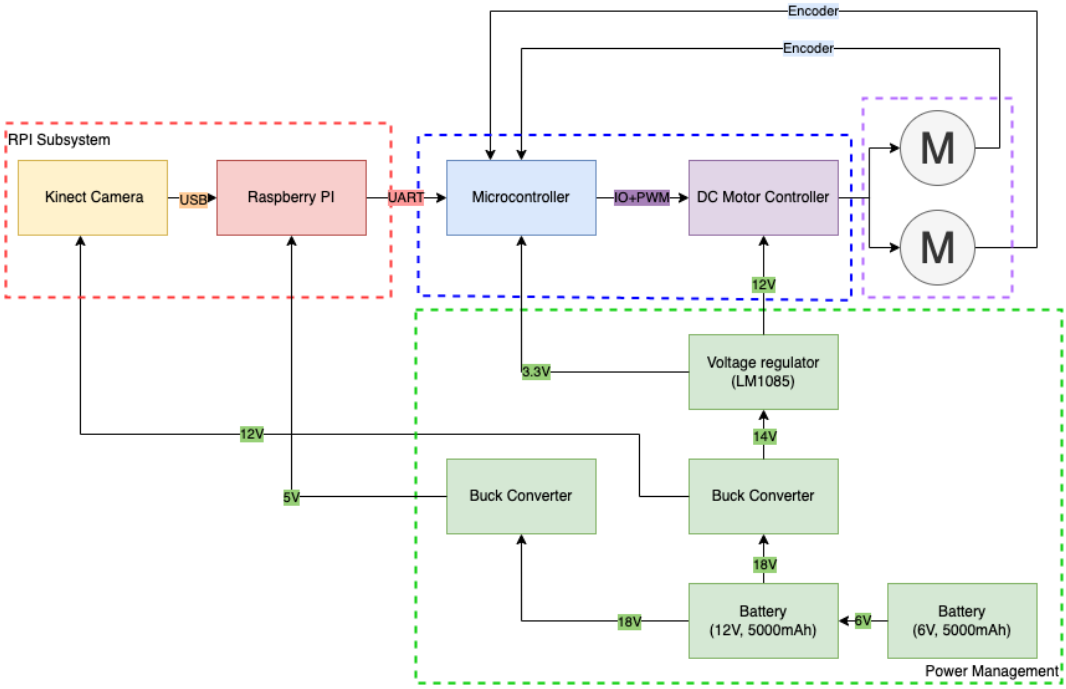

Figure 2) Finalized High Level Block Diagram

2.2 RPI Subsystem

2.2.1 Subsystem Description

The Raspberry Pi subsystem's main objective is to provide the MCU subsystem with the bearing and distance of the user. This subsystem has two main underlying components: a camera interface and the RPI itself. The outputs of the subsystem are stored in a packet and serialized, to be sent to the microcontroller over serial communication.

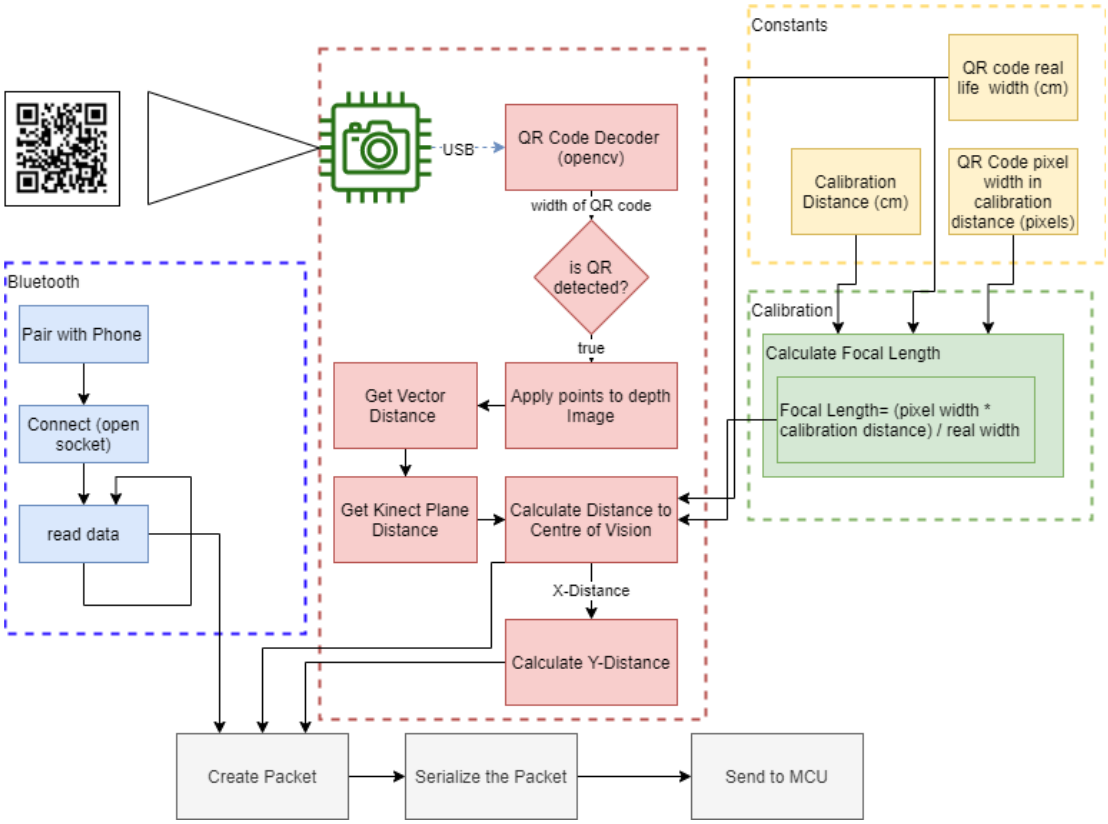

Figure 3) Detailed block diagram of RPI subsystem

The Camera component consists of Xbox 360 Kinect, and uses QR code detection and decoding to determine the distance between the robot and the user. It first searches the camera input for a QR code using OpenCV. The QR code contains a unique sha256 hash for each user, ensuring that the robot follows the correct user. Upon finding the correct QR code, the coordinates of this QR code on the RGB image is applied to the depth image as a crop.

The cropped depth image is flattened and the average is taken, giving us an average distance estimate of the QR code. This average distance is named as Vector Distance. The Vector Distance is converted to the same plane as the Kinect by taking the cosine of the tilt angle; this distance is named Kinect Plane Distance. Next, using the focal length of the camera and the pixel width of the found QR code, the distance between the QR Code and center of vision is calculated, named X-Distance. Using X-Distance, Kinect Plane Distance, and pythagorean theorem we calculate the Y-Distance. Y-Distance is the perpendicular distance between user and robot. The X-Distance and Y-Distance are sent to packet generation.

2.2.2 Verification

Requirement	Verification
Measure the distance to the user with a 5cm error rate in artificial lighting.	All verification tests successful 1. In a controlled environment with artificial lighting conditions, have a user walk in a straight line towards, away from, left of, and right of the Kinect sensor. 2. Measure the distance from the user to the sensor manually using a tape measure and record the value. Simultaneously, use the Kinect sensor to measure the distance and record the value. 3. Repeat step 3 at least 5 times with different users and distances. 4. Calculate the average error rate by subtracting the manual measurement from the Kinect measurement for each trial, taking the absolute value, and dividing by the actual distance. The error rate should be less than or equal to 5 cm.
Serialize measured and collected data, and transfer it to the microcontroller quickly with a Baud Rate of 9600.	All verification tests successful 1. Connect the RPI to the RPI connector. 2. Load a script onto the RPI that writes “Hello” and reads in an infinite loop. 3. Load a program onto the MCU that echoes the input text with a “Got:” prefix. 4. Power on the RPI and MCU using the RPI connector. 5. Verify that the RPI is successfully running the script and the MCU is successfully echoing the input text.

Table 1) RPI Subsystem Requirements & Verification Table

2.3 MCU Subsystem

2.3.1 Subsystem Description

The Microcontroller Subsystem serves as the central processing unit of our project. It acquires data from a range of sources, including kinect depth calculations and encoder feedback speed measurements. The subsystem is made up of four PID controllers, each of which is responsible for a different aspect of the robot. The Y-Controller and X-Controller, two of these controllers, are accountable for trajectory following. By obtaining data from kinect measurements, these two controllers determine the necessary change in distance for QRCode tracking. In the case of the Y-Controller, the steady state of this PID controller represents the desired distance that the user should be followed, which we have set at 90cm. Meanwhile, the second controller manages the robot's lateral movement along the x-axis, adjusting the robot's tilt according to the ratio of motor speeds. Its steady state is set to 0, indicating the center of the robot's vision. The primary goal of this controller is to keep the person in the center of the robot's vision at all times by controlling the robot's tilt. By coordinating the actions of these two controllers, the Microcontroller Subsystem guarantees that the robot remains on track and follows its intended course while keeping a safe distance from the person it is following. To optimize the subsystem's accuracy, we manually tune the PID parameters to minimize overshoot and rise time. As an added safety measure, the subsystem is programmed to halt if it fails to acquire data from the RPI subsystem, as shown in Figures 4 and 5. The system is further elucidated in Figure 6.

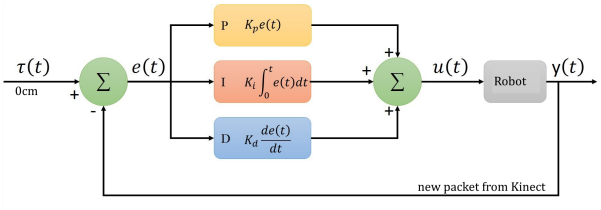

Figure 4) X-Controller

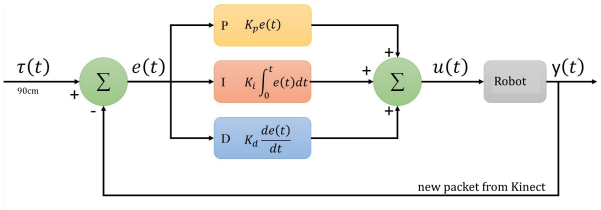

Figure 5) Y-Controller

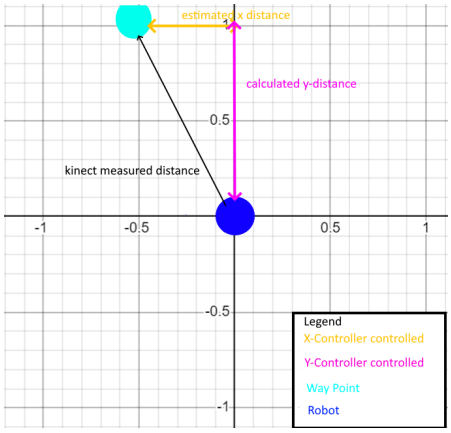

Figure 6) Camera Calibration Visualization

The controllers in our system provide us with the most effective change in y(y) and x distance(x) required for trajectory tracking. To translate this distance into velocity, we utilize a kinematics calculator. This calculator computes the velocity of each wheel using a fixed rotation radius and rotation angle. The rotation angle(θ) is determined by taking the arctangent of x and y. Once the rotation angle is obtained, the ratio between the two wheels can be easily calculated using the equation:(r, is turning radius, l is distance between two wheels which are a constant,)

$$\theta = \arctan(x/y)$$

$$Dist_{leftwheel} = \theta \cdot r$$

$$Dist_{rightwheel} = \theta \cdot (r + l)$$

$$\frac{Dist_{leftwheel}}{Dist_{rightwheel}} = \frac{\theta \cdot r}{\theta \cdot (r + l)} = \frac{v_{leftwheel}}{v_{rightwheel}}$$

Equation 1) Ratio between wheels based on turning,

Initially, we need to identify the wheel that would turn at a slower speed. In our tank drive model, this is straightforward - the left wheel for a left turn and the right wheel for a right turn. Based on this, we compute the velocity of the system in three possible scenarios:

Turning Left	Going straight (Δy=0)	Turning Right
$V_{rightwheel} = \frac{\theta r}{T_{controller\ sample}}$ $V_{leftwheel} = \frac{\theta(r+l)}{T_{controller\ sample}}$	$V_{overall} = \frac{\Delta y}{T_{controller\ sample}}$ $V_{rightwheel} = V_{overall}$ $V_{leftwheel} = V_{overall}$	$V_{leftwheel} = \frac{\theta r}{T_{controller\ sample}}$ $V_{rightwheel} = \frac{\theta(r+l)}{T_{controller\ sample}}$

Table 2) Equations for how the velocities of each wheel affected by direction

Once the wheel velocities are calculated, they are used as inputs for two additional PID controllers. These controllers are responsible for ensuring that the motors are running at the specified speed. This step is essential because setting the same duty cycle does not guarantee that the motors will run at the same speed, as there may be differences in the motors' manufacturing or mechanical characteristics. To address this, the two controllers use encoders to measure the actual speed of the motors and adjust their output accordingly to maintain the desired speed as the steady state. The operation of these two PID controllers is illustrated in Figure #.

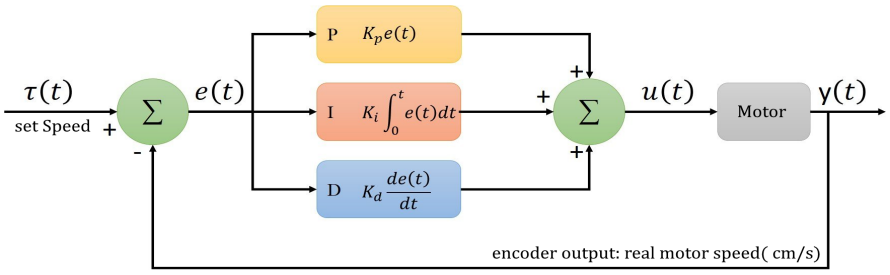

Figure 7) PID Speed Controllers

The Encoder data structures are equipped with individual timers that keep track of the PWM feedback signals generated by the motor's encoder. These signals increment a counter that represents the amount of rotation of the wheel. To measure the distance traveled by the wheel between two samples, our encoder data structure computes the difference between the counter values at the two sampling points. This distance is then used to calculate the motor speed during that interval. Both encoders utilize a shared timer interrupt that triggers the sampling process. Each Encoder structure has an encoder timer that counts the PWM input and is used to compute the motor speed when the shared interrupt timer triggers an interrupt. In other words, the encoder timers count the PWM input and the interrupt timer triggers the calculation of the motor speed.

```
typedef struct{
        int lastCounterValue;
        float distance;
        int counterValue;
        float lastMeasurement;
        int sampleRate;
        int sampleCount;
        int sampleSum;
        double *realSpeed;
        double * desiredSpeed;
        double * motorSpeed;
        PID_TypeDef *speedController;
        TIM_HandleTypeDef * INT_Timer;
        TIM_HandleTypeDef * encoderInputTimer;
//      VirtualTimer * vt;
} Encoder;
```

Figure 8) Encoder Data Structure

Finally, the encoder PID controller utilizes a special timer that we have defined as the Virtual Timer. This timer takes in three inputs - an interrupt timer, a GPIO output, and a duty cycle. The Virtual Timer has an internal timer that keeps track of the number of interrupts triggered by the interrupt timer. When the timer reaches a count of 0 or the value designated by the duty cycle, the GPIO output is toggled(Figure 11). When a change in speed is required, the Virtual Timer uses a function called setDutyCycle(Figure 10). This function uses an updateFlag to ensure that the current duty cycle is not altered before it finishes, which results in a smoother transition.

Figure 9) Virtual Timer Structure

```
typedef struct {
        volatile int timerCount;
        int timerPeriod; //driverSpeed
        float dutyCycle;
        int updateFlag;
        int timerStepDownCount;
        uint16_t outputChannelPin;
        GPIO_TypeDef * outputChannelPort;
        TIM_HandleTypeDef * htim;
} VirtualTimer;
```

Figure 10) Virtual Timer setDutyCycle() function

```
void setDutyCycle(VirtualTimer *vt, float amount) {
        vt->dutyCycle = amount;
        __disable_irq();
        vt->updateFlag = 1;
        __enable_irq();
}
```

Figure 11) Virtual Timer Interrupt Handler

```
void VirtualPWM_IT_Handler(VirtualTimer *vt) {
        __disable_irq();
        if (vt->updateFlag) {
                vt->timerStepDownCount = (int) (vt->dutyCycle * vt->timerPeriod);
                vt->updateFlag = 0;
                __enable_irq();
                return;
        }
        if (vt->timerCount <= vt->timerStepDownCount) {
                HAL_GPIO_WritePin(vt->outputChannelPort, vt->outputChannelPin,
                                GPIO_PIN_SET);
        }
        if (vt->timerCount > vt->timerStepDownCount) {
                HAL_GPIO_WritePin(vt->outputChannelPort, vt->outputChannelPin,
                                GPIO_PIN_RESET);
        }
        if (vt->timerCount == vt->timerPeriod) {
                vt->timerCount = 0;
                __enable_irq();
                return;
        }
        vt->timerCount++;
        __enable_irq();
}
```

The Microcontroller Subsystem is the brain of the project and uses four PID controllers, four timers, and various sensors to control the robot. The Y-Controller and X-Controller calculate the optimal change in distance to follow the QR code and the kinematics calculator converts it into wheel velocity. Two additional PID controllers with encoders provide feedback on the actual motor speed. The encoder data structure samples changes in the counter to calculate distance travelled and the speed of the motor. The Virtual Timer is a unique component that is used by the encoder PID controller to allow for smooth transitions when a change in speed is required. Overall, this complex system requires careful coordination and synchronization of multiple components in order to achieve its intended function. The block diagram in Figure # provides a simplified overview of the various components and their relationships within the system.

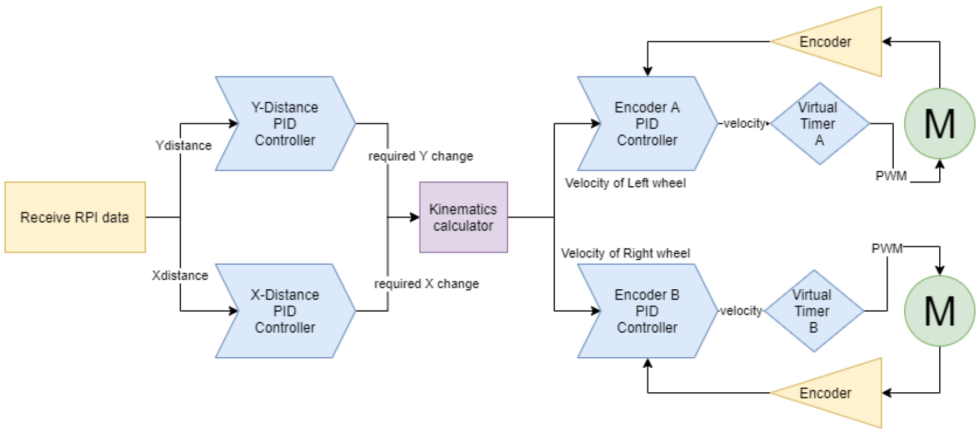

Figure 12) Finalized Block Diagram of Microcontroller Subsystem

2.3.2 Verification

Requirements	Verification
PID Speed controllers can reach steady state	All verification tests successful 1. Place robot on top of box to suspend the wheels 2. Set each encoders desiredSpeed variable to a valid speed between 0 and 1.20cm/s. 3. Observe the Encoder structs real speed value to see if the value is equal to desiredSpeed
PID Y Controller can adjust the motor speed based on distance	All verification tests successful 1. Place robot on top of box to suspend the wheels 2. Run the rpi (main.py) script 3. Start at 90cm and move backwards. 4. Observe the motor speed. 5. Next, move closer to the robot 6. Observe the motor speed. 7. You should observe that the motors become faster as you move further away from the robot and slower as you move closer to the robot.
PID Controller can follow you.	1. Place robot on the floor 2. Run rpi (main.py) script 3. Start moving around with the QRCode 4. Observe if the robot follows you.

Table 3) MCU Subsystem Requirements & Verification Table

2.4 Power Subsystem

2.4.1 Subsystem Description

The power subsystem provides 12, 5 and 3.3V outputs from an 18V input to all components and peripherals of the device. The 12V output supplies power to the drive motors, cooling fans and Xbox 360 Kinect camera, 5V output supplies power to the Raspberry Pi subsystem and the 3.3V supplies power to the microcontroller subsystem. All powered components feature LEDs to provide visual

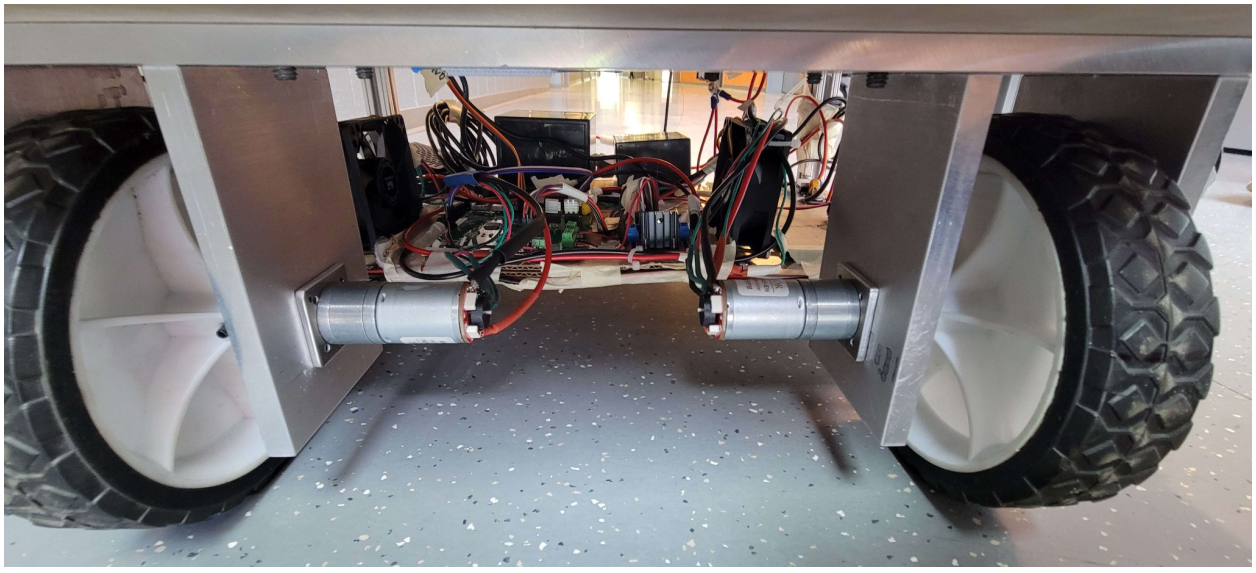

Figure 13) Back view of robot: motors, main circuit, cooling fans and batteries

To power the entire system, 18V of power is supplied by two sealed lead acid (SLA) batteries (12V ML5-12, 6V BP5-6) connected in series. This larger input is necessary for two reasons. Firstly, as the power supply system uses LM1085 linear regulators to ensure stable output voltages, input voltages must be greater than each desired output due to the LM1085's 1.5V voltage drop. Secondly, significant voltage drops occur during use as battery charge is expended, eventually resulting in system components browning out. Given that the device is designed to run for an extended period of time, a substantially higher voltage than required was selected in order to grant a longer operational lifetime.

In order to obtain the desired output voltages, the 18V battery output is passed through two parallel Buck Converter ICs in order to obtain an intermediate step down to 14V and 5V. These intermediate voltages are then passed through an additional LM1085 post-regulator with the 14V being converted to 12V and 5V output being converted to 5V and 3.3V outputs. While the LM1085 is capable of stepping down to the required voltages without the intermediate Buck Converter, testing has shown that such drops result in excessive heating of the LM1085 leading to erratic behavior with operation beyond 5-10 minutes. As such, the intermediate buck converter step down is necessitated for safety reasons. To ensure reliable operation even in hot environments, additional heat sinks and cooling fans are also added to the device. As an added benefit, the voltage step down provided by the Buck converters is notably more power efficient than a LM1085 centric design, allowing for significantly increased battery life.

With regards to the system maintenance, the batteries must be regularly charged and periodically equalized using an external power supply. To charge, the 12V ML5-12 battery is equipped with a separate charger which automatically shifts to a float charge once battery capacity is reached. However, as this charger proved to be inadequate for the 6V BP5-6 battery, it must instead be charged separately using an external power supply. The following settings are applied to the power supply unit with alligator clips attached to the corresponding terminals of the BP5-6 battery: a 6.9V voltage with a maximum current of 1A, charged until power supply current reads 0.15A or lower. These settings are applied in accordance to the following formulae with values drawn from the battery datasheet.

$$Power\ Supply\ Voltage\ =\ (Number\ of\ Cells)\ x\ (Cell\ Voltage\ Limit)$$
$$Max\ Current\ =\ (20\%)\ x\ (Rated\ MAh\ Capacity)$$

2.4.2 Verification

<table>
<tr><th>Requirement</th><th>Verification</th></tr>
<tr><td>System is able to provide sufficient voltage and current for all components</td><td rowspan="3">All verification tests successful
1. Charge both batteries to 6V and 12 V respectively.
2. Measure the individual batter output to confirm that they are 6V and 12Vs.
3. Measure the total battery output from the battery connector to make sure it is above 18V.
4. Check each PCB output pin to make ensure they are within ±10% 3.3V, 5V, 12V
5. Run the system for a period of 60 minutes and probe power output pins again</td></tr>
<tr><td>System is able to step down voltage to safe levels for all components</td></tr>
<tr><td>System is able to provide stable output over extended periods of time</td></tr>
<tr><td>System is able to provide sufficient power for the required run-time</td><td>All verification tests successful
1. Charge the battery fully according to the manufacturer's instructions.
2. Connect the battery to the system and turn it on.
3. Run the system continuously for the required run-time, while measuring the voltage and current levels periodically.
4. Record the voltage and current measurements at regular intervals to ensure that the power level remains within the specified range for the entire run-time.
5. If the voltage and current levels are within the specified range for the entire run-time, then the test is considered successful.</td></tr>
<tr><td>Battery component of the subsystem is able to be removed and replaced in a safe and easy manner not requiring any specialized tools</td><td>No specific test case is mentioned, but this requirement can be verified by ensuring that the battery component can be easily removed and replaced without the need for specialized tools, and by testing the safety of the process.</td></tr>
</table>

Table 4) Power Subsystem Requirements & Verification Table

4 Costs

4.1 Parts

All costs below are in units of U.S. dollars from listed pricing at time of purchase for the quantity required to assemble a single robot. Due to unknown and varying bulk costs for many components, only retail costs at time of robot assembly are listed. Given the distributed purchasing over multiple accounts, both on my.ECE and out of pocket, the total parts cost of the project is difficult to calculate for certain but is estimated to be within $400.00 - 500.00. Due to the vast number of design changes made during the course of this project as a result of both identified errors and procurement difficulties, the below costs represent only the costs of a single unit of the finalized device as to not erroneously include parts not present in the final product.

Table 5)Total parts cost for a single unit

Description	Manufacturer	Total Retail Cost ($)	Quantity
Chassis	Machine Shop	-	-
DC Motor Cont. Module w/ L298N IC	HiLetgo	2.29	-
PCB	JLC	3.4	1
Assorted SMD Components	JLC	3.0	
Cooling Fans	Kingwin	8.01	2
Geared Brushed DC Motor	Robotshop.com	20.00	2
STM32F401RBT6TR	STMicroelectronics	5.68	1
Raspberry Pi 4 Model B	Raspberry Pi	45.00	1
XBox360 Kinect	Microsoft (Lab Checkout)	37.49	1
LM1085 Voltage Regulator	STMicroelectronics	6.88	4
Buck Converters	D-Planet	2.50	2
Lead Acid Battery 12V ML5-12	MightyMax (Lab Checkout)	16.99	1

Lead Acid Battery 6V BP5-6	BB Battery	22.00	1

4.2 Labor

As of 2021, the average annual starting salary (excluding bonuses) for a UIUC graduate is $80,296 for an electrical engineer and $105,352 for a computer engineer. Assuming a 40 hour work week and 52 work weeks per year, these work out to $38.60 and $50.65 per hour respectively. As a whole, we plan to work on the project for roughly 30 hours per week each for a total project period of 9 weeks. With two computer engineers and one electrical engineer, the total engineering labor costs for this project is $113,319.

$$Labor\ Cost = (Total\ Man\ Hours) \times (Total\ Hourly\ Wage)$$
$$Labor\ Cost = (3 \times 30 \times 9) \times (38.60 + 2 \times 50.65) = \$113,319$$

During the course of this project, much of the engineering costs involved can be categorized as research and development (R&D) tasks prior to the finalization of design. Actual assembly time for the finalized product by team members amounts to a total of roughly 24 total work hours over the course of 2 days.

In addition to this, the ECE Machine Shop has quoted 8 hours of labor for the final assembly of the robot including the Chassis-Rotating Tower assembly and motor mount. The UIUC Facilities and Services quotes a billable hourly rate of $90.22 for machinists, resulting in a total machine shop cost of $721.76.

Assuming the estimated labor costs and the estimated upper bound parts cost of $500.00, the total estimated cost of this project is $114,540.76 with the vast majority of the costs being in engineering wages and parts used in R&D.

$$Total\ Cost = Engineering\ Cost + Machining\ Cost + Part\ Cost$$

Under mass production conditions, project costs are anticipated to be significantly lower. For the purpose of this estimation, the following assumptions are made: Labor costs during the R&D

phase of the project are excluded, Illinois state minimum wage of $13.00 per hour assumed for robot assembly, robot assembly requires 8 hours, the materials unit cost remains $178.36. In total, the estimated mass production unit cost is $282.36.

5 Conclusions

5.1 Accomplishments

As a whole, our robot succeeded in achieving all of our high level requirements, frequently exceeding our expectations in many areas. Given the difficulties we experienced in developing the robot, the device provides a strong foundation from which to continue developing a more polished product.

5.2 Uncertainties

Due to the limited number of spare parts we had available in our inventory, we were unable to perform an exhaustive test of the robot's maximum ratings out of caution for causing irreversible damage. Should the project be continued with greater funding and time provisions, values such as maximum cargo load, maximum battery life and maximum operating temperature amongst others should be measured and specified for safe and reliable use by intended customers.

5.3 Ethical Considerations

Regarding the ethical considerations of this project, our team intends to hold ourselves to the highest ethical standards through adherence to the IEEE Code of Ethics. Outlined below are the relevant safety, ethical and regulatory concerns we have identified as well as the means through which we intend to alleviate these issues.

1.) As the power management subsystem only makes use of batteries in parallel with charging not intended to be performed during use, the subsystem does not include a battery management system (BMS) circuit. Having discussed the matter with a teaching assistant, the absence of BMS in our battery system should not pose any issues given the intended discharge-only use of the battery. Thus, to avoid the risk of battery failure or fires, we have designed the robot to require the battery to be charged separately from the rest of the system.

a.) Additionally we will follow best practices for battery safety by storing the battery in a cool well-ventilated area and insulating the battery terminals to prevent conductive materials from coming in contact.
b.) Given the inherent fire hazard of lithium-based batteries, we have elected to use a spill-resistant sealed lead-acid (SLA) battery to reduce the risk of fire and leaks.
c.) Each team member will be informed on how to use a battery spill kit and the Safety Data Sheet instructions in case of a spill.
d.) Each team member will become familiarized with the BP5-6 ML5-12 SLA battery's Material Safety Data Sheet (MSDS).

2.) Regarding the Bluetooth component of our RPI subsystem, we will ensure that only necessary user data is obtained and transmitted. The data, transferred via Bluetooth, shall only be transferred between the user's phone and the robot, shall not be used for any purpose other than the pathfinding of the robot and shall not be retained by any system of the project when not in active use.
3.) We will ensure that our project follows any relevant licensing terms and conditions for all software and parts used in the project.
4.) To prevent any harm to people or materials around the robot due to collisions we will implement PID controllers to ensure that the robot does not accelerate at an alarming rate. In addition to this, we included a physical kill-switch on the robot as well as a software cutoff to stop operation in case of an emergency.

5.4 Future Work

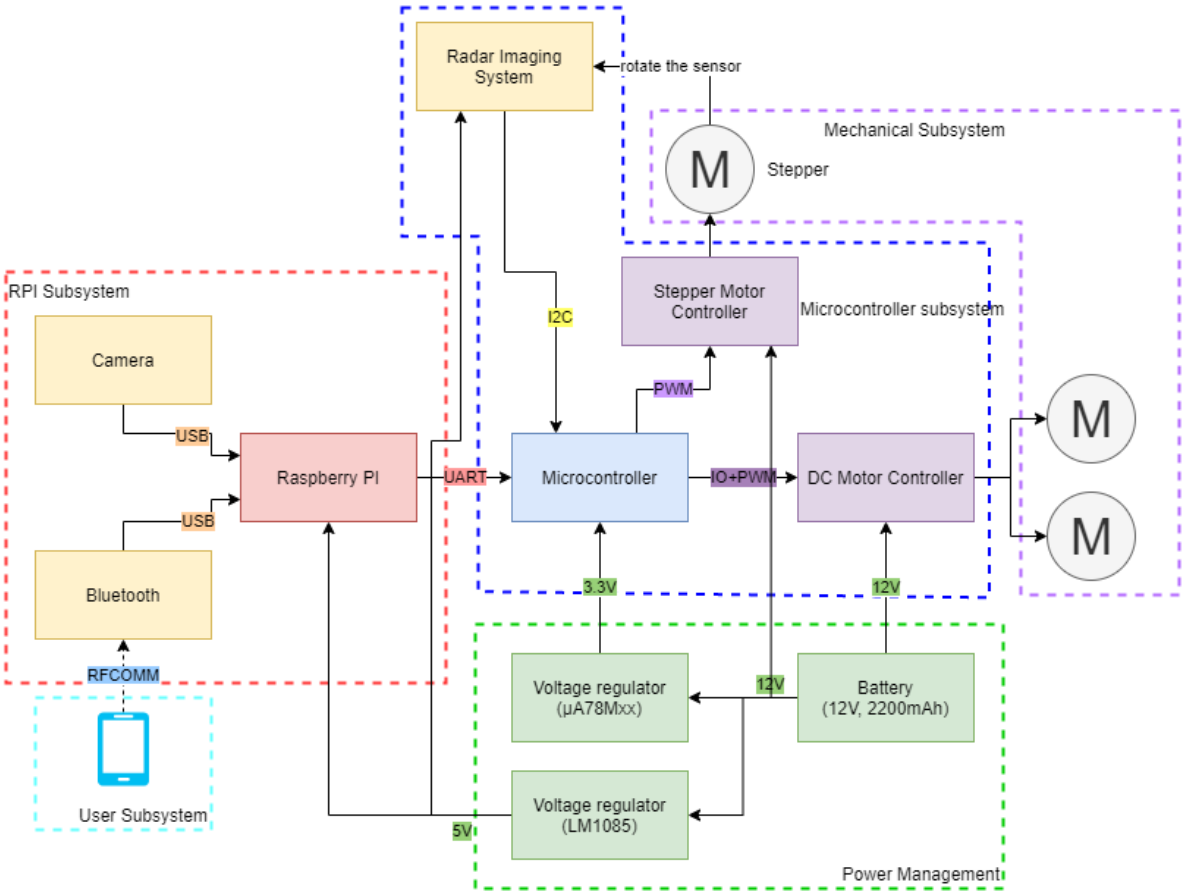

Figure 14) Initial Block Diagram

5.4.1 Unintegrated Features

Due to time constraints and logistical issues, a number of our features were completed but unable to be integrated into the finalized products. Despite completing the LIDAR and Bluetooth features on the devboard, we were unable to integrate them into the finalized product due to mechanical time constraints. The LIDAR was completed but could not be integrated into the robot due to the late delivery of motors and limited machine shop time. Similarly, the Bluetooth feature was fully functional on the devboard but could not be integrated into the finalized product due to time limitations. Given more time and budget, we would have liked to integrate these features into the robot as they would have improved its functionality and performance. However, despite these limitations, we were able to successfully complete the project and create a functioning robot that met most of our initial requirements.

QR Code Perspective correction:

To ensure accurate detection of QR codes, we utilized OpenCV's QR code detection algorithm. However, this method is limited in detecting square QR images and can struggle with tilted images. To overcome this, we implemented a perception warping technique, which corrects the perspective to make it look like there is no tilt.. This involved creating a rotation matrix that corrected for the tilt caused by the Kinect sensor. The inverse of this matrix was then saved and used as a filter to apply the rotation matrix to the image. The result was a more accurately detected, square QR code image with higher precision thanks to the filtering process. By employing this technique, we were able to overcome the limitations of the QR code detection algorithm and ensure accurate detection of QR codes regardless of their orientation.

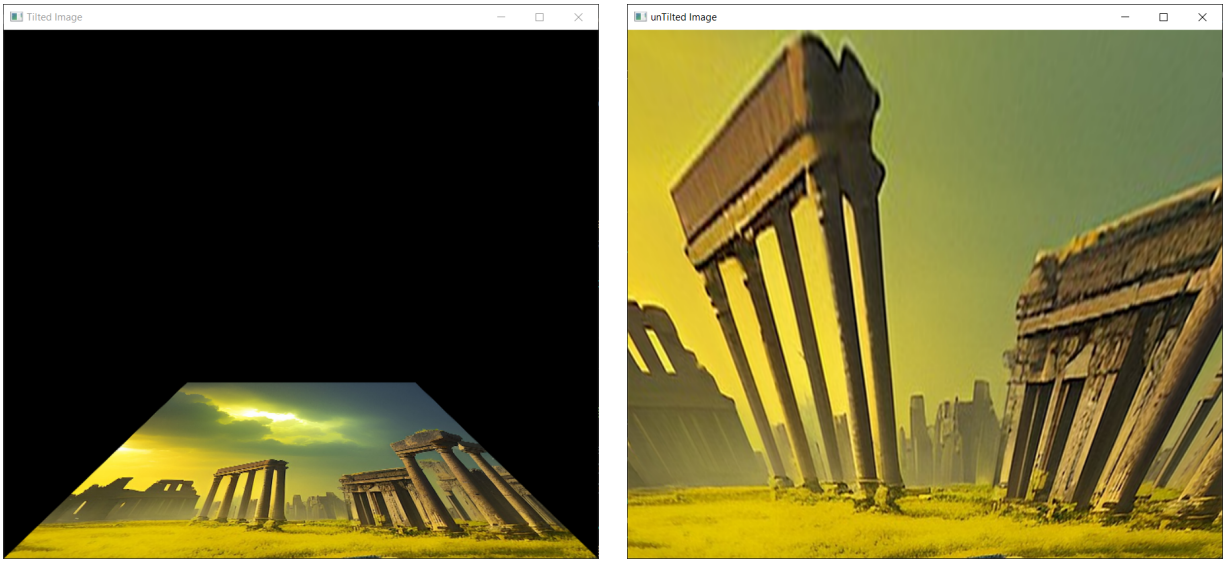

Figure 15) Sample output for Kinect perspective processing (left: before. right: after)

5.4.2 Further Plans

The testing of design limitations was limited due to a lack of spare parts. However, the design team has identified some areas for improvement. One such area is the unit enclosure, which can

be redesigned to improve ergonomics and user-friendliness. Another important consideration is power management. To make the battery charging process more convenient, a battery management subsystem can be implemented to allow charging without the need to remove the battery from the robot. This can be achieved through the use of buck converters integrated onto the PCB, or potentially with a separate power PCB. These improvements will not only enhance the functionality of the robot, but also improve the user experience. Lastly, improving the RPI subsystem by switching from a RPI to a product with GPU for better performance such as a Jetson,Cloud GPU, or a normal GPU.

6 References

[1] World Health Organization. (n.d.). *Ageing and health*. World Health Organization. Retrieved May 3, 2023, from https://www.who.int/news-room/fact-sheets/detail/ageing-and-health

[2] Ziegler-Graham K;MacKenzie EJ;Ephraim PL;Travison TG;Brookmeyer R; "Estimating the prevalence of limb loss in the United States: 2005 to 2050," *Archives of physical medicine and rehabilitation*. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/18295618/. [Accessed: 23-Feb-2023].

[3] Grainger Engineering Office of Marketing and Communications, "Salary averages," *Electrical & Computer Engineering | UIUC*. [Online]. Available: https://ece.illinois.edu/admissions/why-ece/salary-averages. [Accessed: 23-Feb-2023].

[4] "University of Illinois facilities and services," *Facilities and Services University of Illinois Urbana Champaign*. [Online]. Available: https://fs.illinois.edu/services/f-s-service-rates. [Accessed: 23-Feb-2023].

[5] "Mighty max battery ML5-12 [3/8] 8. exposure controls ... - mans.io." [Online]. Available: https://mans.io/files/viewer/1693682/3. [Accessed: 23-Feb-2023].