The Autonomous Cart 'FollowBot'

Adil Ali (EE), Carlos Gonzalez (EE),
Abhinav Sharma (CPE), David Falter (CPE)

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

Abstract -- This paper will discuss the design, and construction of an autonomous luggage cart. The cart utilizes a microcontroller that receives positional information to follow the user's path. The positional information will be wirelessly acquired from an application on the user's cellphone. Sensors will be used to avoid collisions with any obstructions or objects in the specified path. FollowBot's purpose is to simplify and automate the transportation of luggage and other materials.

Index Terms – Autonomous, Beacons, Bluetooth Low Energy, Triangulation, Ultrasonic Sensing

I. INTRODUCTION

FollowBot is an essence an autonomous vehicle that is purposed towards the transportation of items such as luggage, coolers, etc. which allows the user to simply walk while their goods safely follow them to their destination. The FollowBot's main goals are to be incredibly easy to use and have accurate tracking which will be accomplished through the utilizations of a plethora of sensors, microcontrollers, and external receivers.

This robotic cart will include a two-layer copper printed circuit board which houses the Atmega328p microcontroller that will serve as the brain of the system. The PCB will also contain the HM-10 Bluetooth 4.1 adapter which will be utilized to wirelessly broadcast the user's movement data to the microcontroller. An Android cellular device will be connected to nine Estimote Bluetooth low energy beacons which will triangulate the user's position based on a created algorithm. This positioning data will then be transmitted to the microcontroller for processing and if necessary trigger the LM298 motor controller. The motor controller is connected to two 12 volt Cytron motors that are used to turn and move the cart in a variety of directions. The front and sides of the cart will be fitted with HC-SR04 ultrasonic sensors that function as a collision detection system. This allows the cart to stop when there is an obstruction, allowing for the safety of this product to be maximized. The major subsystems of the FollowBot are the movement, obstacle avoidance, and communication systems which have been calibrated and tuned to minimize latency making the user experience as streamlined as possible.

II. SYSTEM OVERVIEW

When designing the FollowBot a specific set of criteria or specification requirements were determined by the group members. These requirements were meant to serve as an evaluation or guideline as to how successful our design approach was. Having guidelines allows us to tune and spec the components to meet these standards. The overall project requirement specifications that were set were:

- Vehicle Size: 16"x16"x6.5" (WxLxH)
- Collision Detection Range: 5 cm to 100 cm
- Speed: atleast 2 miles per hour
- Battery Life: greater than 2 hours
- Tracking Accuracy: accurate to 2 meters
- Carry Weight: >25 lbs
- Unit Weight: <20 lbs

It is important to note that these were not the only specfications set however, the ones listed above were the primary constraints for the design. These specifications are the minimum goals for the design and under the proper conditions should be exceeded as to provide maximum performance and efficiency.

III. SUBSYSTEMS

In this section each subsystem and its components will be further discussed in order to provide a deeper understanding into why certain designs were made and how these designs affect the designs performance. The analysis of each individual system will also provide a clearer picture into how these subsystems come together to form the backbone of the FollowBot.

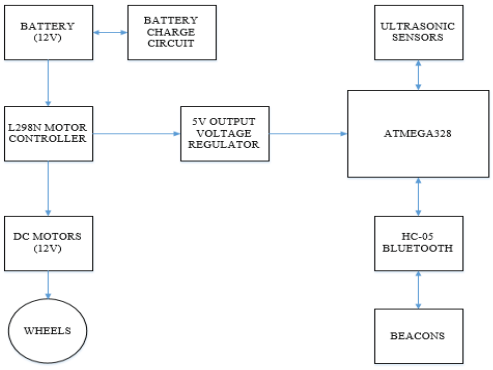

Figure 1: Overall System Block Diagram

A. Microcontroller

The microcontroller is a key part of this project: it drives the hardware and allows it to do all the functionality we have implemented. The microcontroller is used for quite a few tasks: motion controls of the motors because of software input, collision detection and avoidance, and Bluetooth pairing and functionality. The MCU (microcontroller unit) handled these tasks with ease and maintain high functionality in real time to not hinder the use of the product. Since the MCU is the backbone of combining our software and hardware, it was very important to understand the benefits and drawbacks of several types and brands. We discussed and juggled between several different types of microcontrollers and even development boards that were essentially mini-computers. We wanted a standout choice that was easy to program, easy to implement electronically, and powerful enough to handle all of our tasks.

The most popular MCU in our list of considerations is often considered a fan favorite for tinkerers and small project designers around the world. The ATmega328P is a very familiar microcontroller that supports a wide range of applications. The ATMega was a clear frontrunner in our choice of microcontroller due to the tremendous support and coding libraries it shares with Arduino, its biggest user. The biggest bonus of going with Arduino to begin with is the overwhelming amount of resources available on the internet to develop with the chip itself. People have been working on Arduino MCUs for years now and it has always had the upper hand for people looking to get into custom projects due to its ease of use. The ATMega328P also offered us a great price point at roughly $4 per microcontroller. We were able to get Arduino UNO boards for dev board testing before ordering our PCB and this allowed for rapid prototyping ahead of time. This MCU had very clear strengths and thus beat out the competition for us very early.

One of the biggest reasons this MCU was at the top of our list was due to its inseparable relationship with Java. The Arduino IDE is even written in Java, and everything on the board can be used to communicate in programs written in Java. The software engineers on our team feel very comfortable with the language and thus feel as though the ATMega chip could expedite the development process greatly. This microcontroller solution feels easy, but reliable, and thus makes it a very strong contender. Despite being a “user friendly” choice in the market, it is hard to underestimate the shadow that it casts. Arduino has been around for a long time taking the reigns as the leader of the public consumer market. This board may not see as much use in industry, but it certainly seems like a strong option for a project such as ours. We used the Arduino IDE for software development and testing, and though we used an Arduino UNO dev board for most of the early prototyping and development, swapping it out for a soldered PCB with the Arduino software flashed onto the board was simple enough. The microcontroller fit our needs by providing us with enough pins, support for all of our subsystems, and plenty of help with configuration online.

B. Collision Detection & Obstacle Avoidance (Ultrasonic Sensors)

While the focal point of this device is its movement abilities and ease of use, one of the most important considerations was the safety of both the user and other people in this device’s operating area. This meant that the FollowBot’s collision detection and obstacle avoidance systems had to function flawlessly without any hiccups. An HC-SR04 ultrasonic sensor was used to accomplish this task as it allows for the detection of an object in a range from 2 cm to 500 cm. Other options such as lidar which provides a much larger range were considered however, they deemed too expensive and provided little to no improvement over the ultrasonic sensors. The relatively large sensor range of the HC-SR04 allows the cart to have adequate time to stop and also provides it with enough space to turn if necessary. The ultrasonic sensor works by sending out a sound wave of a specific frequency every 10 microseconds. The trigger pulse is sent through the TRIG pin on the sensor after which eight 40 kHz sound waves are sent out. The ECHO pin of the sensor then receives the soundwave and from this the distance of the object can be calculated by the time it took for the pulse to leave and return to the ultrasonic sensor.

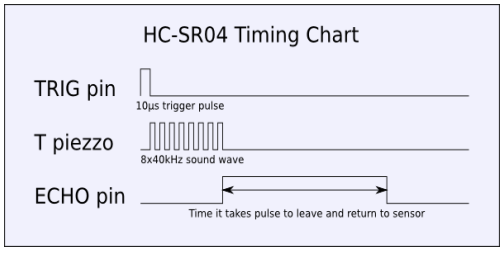

Figure 2: Ultrasonic Sensor Timing Chart [1]

There were two approaches to obstacle avoidance that were considered during the design process. The first is if the object is within a certain range then the cart should stop and then turn until the object is no longer present in the FOV. It could then proceed forward and continue following the specified path. As shown in the figure below the HC-SR04 works best when the object is within ±30º of the sensors field of view.

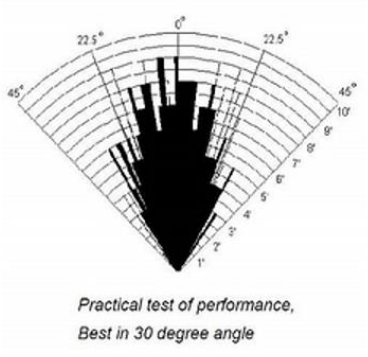

Figure 3: Range & Field of View of Ultrasonic Sensors [2]

The other simpler option is to simply stop the cart and wait to see if the object passes. Our method of managing object avoidance is to first observe our impending "collision" and allow the code to assess whether it may be an immediate issue. An example of such is a person walking by 2 meters ahead that will have probably cleared the space by the time our bot reaches the destination. In this case, we cannot react preemptively and stop our device. This implementation would be clunky and inefficient, and borderline unusable in a crowded environment. We chose to dynamically assess collisions in our code, and we are using the ultrasonic sensors to constantly update and send information to the microcontroller to allow it to determine whether it wants to stop or not.

A key variable here is "stopping distance". If the object is an imminent collision for the FollowBot, it first determines when it needs to stop based on its current stopping distance. This variable is effected by the weight on the device and the current speed of the vehicle. By analyzing this in real time, the device will stop before any collision occurs despite the user of the bot continuing expecting their product to follow them. After "handing the collision" – we consider this either the object in the path clearing the way or finding an alternate path – the device will resume its instruction and correctly calculate where it needs to go and whether it needs to catch up. By managing these directions and coordinating our locomotion accordingly, there should be no cause for collision unless forced by a foreign entity.

C. Power System

The power system of the FollowBot provided another challenge as one needs to ensure that the power draw is minimum to maximize the battery life of the cart. To power the entire system a variety of different battery options were tested including sealed lead acid batteries as well as lithium polymer batteries. Originally the FollowBot utilized a 7mAh 12-volt sealed lead acid battery. The capacity of this battery was very large and sealed lead acids are durable under heavy loads however, it came with the drawback of being incredibly heavy. This was detrimental to the design because if the weight of the cart increased this meant the carriable weight would decrease. Therefore, the battery was switched to a smaller 2200 mAh 11.1-volt lithium polymer battery. The Atmega328p and all of the peripheral devices such as the Bluetooth adapter and the Ultrasonic sensors all utilize 5 volt VCC connections. This meant that the voltage from the battery had to be regulated to 5 volts DC. This was accomplished by connecting the battery to the motor controller and then the motor controller to the Atmega328p and the secondary devices. The motor controller has a built in switching regulator which converts from 7 to 20 volts DC to 5 volts DC which is perfect for the FollowBot. Utilizing a lithium polymer battery as provides a few drawbacks with charging being one of them. Lithium polymer batteries must be charged using constant current/constant voltage charging. This form of charging essentially keeps the charge current constant until the battery reaches its fully charged state. The fully charged state of a lithium polymer battery can be defined when each cell of the battery pack has a voltage of 4.2 volts. Once each cell reaches this peak voltage then the charge rate is reduced. The voltage of each cell should also undergo a process known as balancing. This means that the voltage in each cell should equalized to ensure the cells discharge at the same rate. Lithium polymer batteries are also problematic in that they have a higher internal resistance as compared to other battery types and have a relatively low shelf life. This means that more heat will be generated when the battery is operating and also that they must be properly stored or else they could be damaged due to adverse environmental conditions (humidity, temperature, etc.)

D. Motor Controllers

The motor controller used for FollowBot was the L298N Dual H-Bridge DC Stepper Motor Driver

Controller. It can drive one 2-phase stepper motor, one 4-phase stepper motor, or 2-4 DC motors. It contains an H-Bridge circuit, as mentioned in the name, which creates higher working efficiency. It is more stable and reliable with a large capacity filter capacitance, and after flow protection diode. It has low heat, and an outstanding anti-interference performance. In addition, the motor controller offers pulse width modulation, which can independently manipulate the speed of each motor and the direction by driving a current in either polarity. This motor controller also offers a 5V input/output voltage regulator. This voltage regulator is channeled in our PCB to power up the microcontroller. Some important specifications to know about this motor controller are shown in the table below.

TABLE I
SPECIFICATIONS OF THE L298N

Specification	Value	Units
Drive Voltage	5-35	V
Max Stall Current	3	A
Max Power	25	W

The FollowBot uses one motor controller to power two DC motors. The motors used are Magnolora 12V DC 25MM 120RPM Powerful High Torque Motor. They have a nominal voltage of 12V, stall torque of 111oz-in, and a stall current of 1.8A. It is imperative to know these specifications on a motor. The stall torque determines how much weight the motors will be able to handle. The stall current is the amount of current the motor will draw at maximum torque conditions. The motor's power can be approximated by using the formula P = IV if it is not listed on the manufacturer's specifications. The motor's maximum power can by calculated by using the stall current and nominal voltage. Sometimes the gear ratio will be stated on the manufacturer's specifications. The gear down acts to increase torque and reduce the revolutions per minute. The ratio means the amount of revolutions that the driver gear must take to rotate the driven gear once.

Torque is known as a rotational force. In other words, it can be known as a twist to an object. It can be calculated by a force that is acting at a distance away from a pivot multiplied by the distance. DC motors rotate rapidly and for most cases, they have low torque. To increase the torque in a DC motor, a gear may be added. However, the trade-off of adding a gear to a DC motor is a decrease in the motor's speed. To have an accurate value for torque needed, the equation below can be used.

$$T = M(a + g\sin\theta)r$$

In this equation, **M** is the specified weight of the FollowBot, $\boldsymbol{\alpha}$ is the maximum speed reached in two seconds, **g** is gravitational force, $\boldsymbol{\theta}$ is maximum incline to climb, and **r** is radius of the drive wheels.

Process for DC Motor Control

Each type of motor has a specific way in which it can be controlled and this varies depending on the type of motor. Some motors have a more complex control method while others are relatively simple. These processes will be discussed and analyzed for DC motors in this section.

1. Nominal voltage. DC motor controllers usually have a range of voltage in their product description. The motor's nominal voltage meets within the range of voltage that the motor controller can supply.
2. Continuous current. The motor controller selected provides a current equal to or greater than the motor's continuous current. However, most motor controller manufacture companies do not specify the motor's continuous current but instead specify the stall current. If this happens, an easy way of estimating the motor's continuous current is to take 20% to 25% of the stall current given for the motor.
3. Control method. This includes PWM, R/C, UART, or analogue voltage. The pin types of the microcontroller selected have control methods that are needed.
4. Single versus dual. Dual DC motor controllers can operate the direction and speed of two identical dc motors. They only have one power output, so controlling motors at different voltages is not possible.

Pulse Width Modulation (PWM)

Pulse width modulation is an important factor of why the FollowBot can move and turn. It is a unique modulation technique used to encode a message into a signal. The signal is a square wave that is continuously switched on and off. This pattern of turning on and off can simulate a range of voltage by changing the amount of time the signal spends on against the time the signal spends off. The continuation of the time that the signal is on is called, the pulse width.

E. Motion System Design (Motor Configurations)

The image below shows the connection between the motors with the motor controller, and the motor controller with the microcontroller.

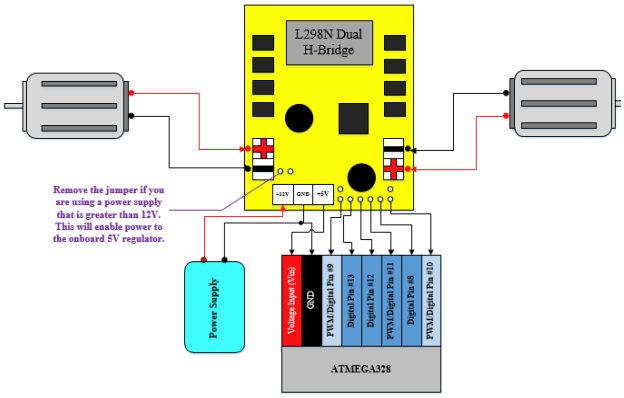

Figure 4: Motor Controller to Motor Configurations

The positive and negative wires of the motor on the left are screwed to the positive and negative terminals on the left of the motor controller. We will call these terminals OUT1 (+) and OUT2 (-). This motor will be controlling the left wheel of our design. The connection will be the same for the motor on the right side. We will call these terminals OUT3 (+) and OUT4 (-). This motor will be controlling the right wheel of our design. There will be one power supply. The positive wire of the power supply is connected to the +12V input terminal and the negative wire is connected to the GND terminal on the motor controller. The last hardware details are the connections between the pins on the motor controller and the pins on the ATMEGA 328.

IV. PROTOTYPE DESIGN

Wood is the material used to design the FollowBot. It consists of medium-density fiberwood (MDF) and common board. MDF is used as the base while the common board is used for the edges. The FollowBot has two sections, one section is used to hold the electronic components while the other is used to hold a specific object, such as a luggage. The section that holds the electronics is 8 x 8 inches while the section that holds the object is 16 x 16 inches. Lastly, the height of the vehicle is approximately 5.5 inches. The image gives a visual representation of the dimensions of the FollowBot.

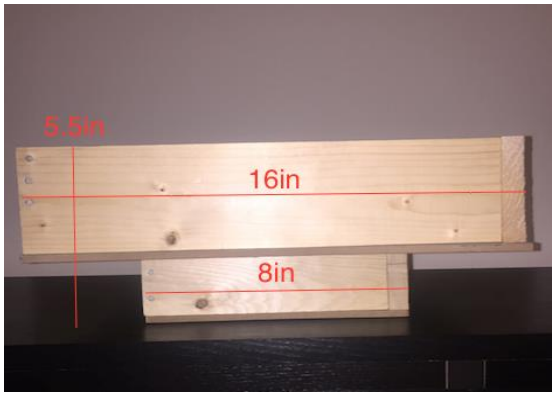

Figure 5: Prototype Base Construction

The vehicle design of FollowBot consist of three wheels. The front two wheels are 6 inches in diameter, semi-sold tire with polypropylene hub. They provide a maximum working load of 94 pounds. The back wheel is a 3 inch in diameter, hard rubber light duty swivel caster. The solid rubber on these wheels are great for durability and smooth movement. The caster has a 100-pound capacity weight, and can rotate 360 degrees to help maneuver heavier weight on the FollowBot.

There are three 3D printed components on the FollowBot which are used for important purposes. One is used to mount the motors to the edges of the 16-inch wood platform. Another one is used to mount the swivel wheel to the back of the vehicle. The last one is used to provide a connection between the motors, the shaft hubs, and the wheels.

V. BLUETOOTH LE HARDWARE

Localization and communication with the mobile application of it's user using Bluetooth LE technology. The three major components use to facilitate these functionalities are the FollowBot's Bluetooth LE module, the Beacons who's packets the module consumes, and the Bluetooth module on the users mobile device.

When choosing the beacons for the FollowBot, the primary points of discussion were cost, range, reusibility, and supporting APIs. At twenty-one dollars per beacon, 100m of range, an easy to use android and iOS indoor location sdk, and other added on features such as a fall detector; the Estimote location beacon was selected as the best option. While having the ability to transmit both iBeacon(Apple) and Eddystone(Google) adverisement packets, the Estimote also has a setting to send out a special Location packet that only contains

information needed to calculate the RSSI as well as the UUID that a receiving bluetooth module would need to identify the individual beacons. Aside from the unnatractive, yet completely water proofing casing, there is no better beacon for the purposes of this project.

For the Bluetooth Module on the Followbot itself, the HM-10 was selected for it's low cost, large quantity of available documentation, and compatibility with the ATmega328P. It used Bluetooth 4.0 LE technology which is necessary for consuming packets from LE beacons. The board comes with an onboard LED with different color and blinking combinations to assist the developer working with the board. This module is also extremely resistant to user error in voltage regulation as it comes with its own allowing it to run efficiently with souce from 3.3V to 6V. Similarly to other modules, the HM-10 needs to be connected to the Microcontroller using 4 wires: RXD, TXD, VCC, and Ground. The RXD receives serial UART and the TXD transmits it. This is what allows serial information on the location to be sent from the mobile phone, to the HM-10 and ultimately to the FollowBot so that it can use it for the locomotion portion of the code base. The HM-10 is both iOS and android compatible for serial communcation.

The last portion of the Bluetooth chain of communcation is the users mobile phone. The goal of the team was to make the app both on the iOS and the Android. This has been accomplished, however after testing, there is a clear winner between the two competing operating systems in regards to bluetooth localization. Due to the large variety of hardware configuration that can be found in the line of Android phones, and the amount of fine tuning needed to accurately find RSSI values between a beacon and a module(see section VI) the iPhone ends up being far superior for apps that are meant to be pushed out to multiple devices. While the app for the Android may be fine tuned for the Samsung Galaxy, there is a large chance that it be hopelessly inaccurate when opened on a Google Pixel. For these reasons the iPhone 5s+ will be the only devices fully supported by the FollowBot.

VI. LOCALIZATION

For the FollowBot two separate localization algorithms are deployed: one for the mobile application and they other for the FollowBot itself. Both algorithms at their base take advantage of the relative Received Signal Strength Indicator(RSSI) between a beacon and a module and use it to estimate the distance.

The most commonly used Formula and the one we use for correlating RSSI to is:

d = 10^((TxPower)/10n)

In this formula **d** is distance, **TxPower** is a constant and part of the packet sent by the beacon, and **n**, is an environmental factor that can be tuned for different modules. his value of **n,** needs to be heavily considered when writing applications that are meant to be widely used among different devices. With great variation, it is nearly impossible to use this equation. This is the reason, why FollowBot is currently only fully supported by iOS devices. Below in Figure 6, you can see a rough correlation between RSSI value and the distance.

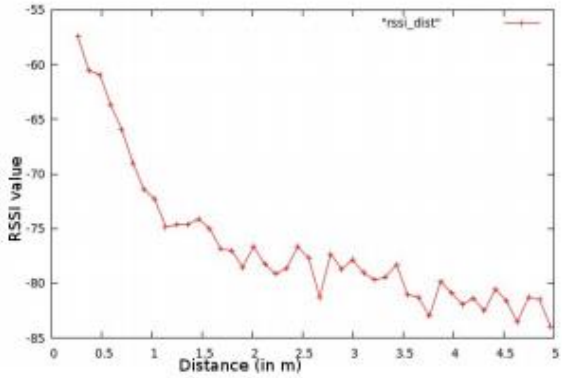

Figure 6: Distance to RSSI

Unfortunately, in addition to the issue of **n**, the formula has another issue it needs to deal with. The fluctuation of RSSI values even without disturbances in large. In Figure 7, you can see the heavy fluctation of RSSI values, and when left unfiltered, these RSSI values are just about useless for any meaningful localization algorithm.

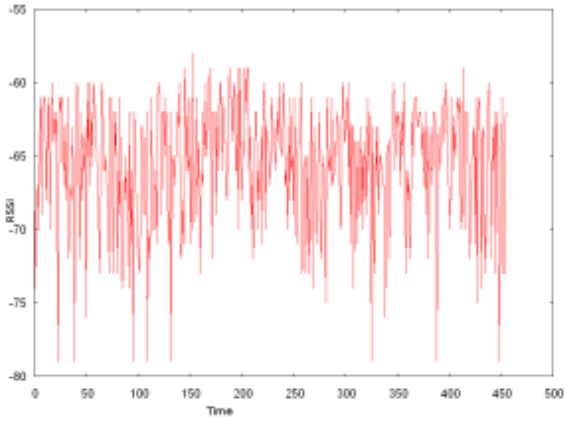

Figure 7: RSSI Over Time at 3 meters

In order to deal with the fluctuations in RSSI values, it is absolutely necessary to filter out the noise. In our algorithms we take advantage of two types of popular filters.

On the ATmega328P the popular Kalman Filter, used on the Apollo rocket and Tomahawk missiles, was selected cancel out the RSSI noise. The Kalman filter recursively estimates the current state of the distance by using the previous state and current measurement. The downside of the filter and any other particle filter is the lack of previous states at the beginning of the calculations, however this is not an issue with a robot meant to follow a user over a long period of time. Overtime, the FollowBot have more accurate idea of the distance between itself and the Estimote Beacons.

Using the filtered distances, the final step of the localization algorithm is to use those distances to find the location of the FollowBot. For this, an iterative Multilateration formula was deployed. While working on this, it is important to note while the more beacons, meaning more distances you have, the more accurate the multilateration formula becomes. This fact needs to be balanced with the amount of computation the ATmega328P needs to perform in order to filter multiple different streams of RSSI values and convert those into distances and the distances into a location.

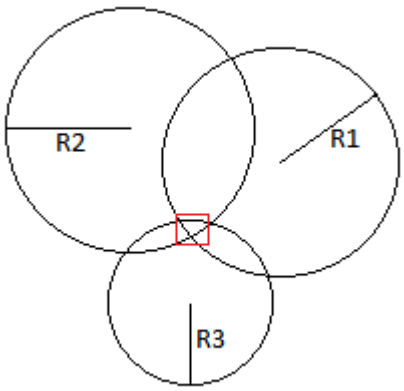

Figure 8: Trilateration

VII. LOCOMOTION

Locomotion of the FollowBot was one of our key challenges when discussing and implementing the project. Our design incorporates two wheels driven by DC motors at the front of the machine as well as a swivel wheel dependently turning at the rear middle. We chose this style of locomotion due to the ability to quickly and efficiently implement differential turning. We are driving both motors using our ATMega328p and allowing our mobile app to control the movements through various equations present within the code. The product can turn and adjust to variable movements based on the user and make changes on the fly. Our Bluetooth trilateration system allows for efficient tracking within its observation radius and thus allows fine turning by accounting for direction received by the host. The hardware design was specifically implemented to allow for the software to be able to translate movements into directions. Our early prototypes were focused on 4-wheel locomotion implementing motors in each wheel and steering each side separately. We quickly realized that this would cause more trouble than we needed to deal with and we made the design choice to go with the 2 forward wheels driven by motors and a swivel wheel at the back.

VIII. TURNING

Differential turning is a key principle that our code and locomotion subsystem relies on. By allowing each side to be independently controlled by software, we can adjust wheel and motor speeds to issue any degree of turn we may need. We can also completely stop and execute directions to turn each wheel in opposite directions if we need to turn on the spot, in the case of an imminent collision. Previously, our prototypes displayed very inconsistent and unreliable tests while turning. We had a 4-wheel system that also implemented differential turning but the stress on the wheels due to the variations caused very inefficient and slow turning. In addition to our other problems, we decided to scrap that prototype as a result and instantly implemented a new, functioning design. While a 4-wheel design is usually optimal for robotics and automotive applications, it relies heavily on using an axle with implemented steering. We chose to go without such subsystem, and our locomotion and turning is thus more reliable.

Our main focus around maneuvering and turning as key topics of study when researching our project was accuracy and feasibility. Installing an axle and proper steering system was far too mechanical for us and involved many other aspects of design that seemed unnecessary and past our abilities in the timeframe. Since a 4-wheel design was limiting us and we did make a transition to the 3-wheel differential turning idea, our prototyping and development became a lot simpler. We also switched from smaller hollow wheels filled with foam to semi-solid large rubber wheels. These provided a lot of stability and allowed our project to move way faster and even carry more weight. We were not using our motors to their full capacity and thus severely impacting our turn radius and ability. Changing strategies really helped us here.

VIIII. PROJECT OPERATION PROCEDURE

The FollowBot project aims to provide a user with a seamless carrying companion that can assist them in carrying burdensome cargo across various distances. User experience is a priority, and achieving seamless integration between host and device is both exciting and rewarding. To begin, a user should approach the idle FollowBot with their handheld Bluetooth capable device. The user must first download the FollowBot app from their desired platform application store and prepare for pairing. The user should now be able to sign in to their account on the mobile interface and begin the pairing process. An idle FollowBot will be constantly searching for new devices to pair with, and associate links based on information relayed by the Bluetooth connection through the app.

Once the pairing process is complete, the user is now on their way to burden-free travel. After the device and the user's mobile phone are paired, the FollowBot will search for beacons around the area to gauge location and initialize its following capabilities. The user will designate the follow mode and will then proceed to be closely shadowed. Travelling through crowds and obstacles, the FollowBot should keep up with haste and avoid any possible collisions along the way providing the user with an efficient and safe experience. The user should limit the use of FollowBot sessions to avoid draining too much battery and should always stay a safe distance away from the moving object while walking. Once the user has reached their destination, they will be able to disengage the device with a simple touch of a button on their handheld device and continue on with their travel. The bot will then reset and allow for continued use by anyone else who may wish to test.

X. CONCLUSION

The goal this idea was to make a creative way for more convenience in situations in which objects need to be transported from one location to the next such as an airport. FollowBot is an elegant solution to a popular problem. Not only are does it aim to improve the public market and general happiness across airports, we believe it can expedite foot traffic and relieve blockages in many of our country's biggest airports. By creating a mobile platform capable of carrying luggage and much more, we believe that FollowBot can even have applications beyond Senior Design. FollowBot prioritizes safety and efficiency – this has been repeated and reiterated – because it is a fundamental principal of our design. Areas with large foot traffic can easily be a huge challenge for an implementation such as this, and we want to show that with good object detection and avoidance, this can be surmounted. We believe that our choices in design have been consistent with the consumer market as well as strategically viable in real-world production. As this idea expands, we hope it will be applied to different fields due to its versatility. We faced plenty of challenges on our way to making this a success, and we believe that every design and software decision we made was crucial to that success.

BIOGRAPHY

Carlos Gonzalez is currently a senior at the University of Central Florida and will be graduating with a Bachelor of Science in Electrical Engineering in December 2017. After graduation, his goal is to pursue a career in the United States Air Force.

Adil Ali will be graduating from the University of Central Florida with a Bachelor's of Science in Electrical Engineering and plans to remain in the Orlando area to pursue a career after graduation.

Abhinav Sharma will be graduating from the University of Central Florida with a Bachelor's of Science in Computer Engineering and will be employed with Texas Instruments in Santa Clara upon graduation.

David Falter will be graduating from the University of Central Florida with a Bachelor's of Science in Computer Engineering and is currently employed as a Support Software Lead at MINT Media Interactive Software Systems LLC.

REFERENCES

[1] EngineersGarage. "Distance Finder Based on AT89S52 and Ultra-Sonic Sensors HC-SR04." EngineersGarage, 5 Nov. 2014, www.engineersgarage.com/contribution/experts/distance-finder-based-at89s52-and-ultra-sonic-sensors-hc-sr04

[2] .Sapkota, Sagar. "Sagar Sapkota." Build Circuit, 12 Dec. 2015, www.buildcircuit.com/simple-ultrasonic-range-finder-using-hc-sr04/.

[3] Raghavan N Aswin "Accurate Mobile Robot Localization in indoor environments using Bluetooth" Retrieved July 7, 2017. Web. http://www.cse.iitm.ac.in/~ravi/papers/Ashwin_ICRA10.pdf